

**FRAMWORK FOR PRODUCT ARGUMENTAL
ANALYSIS OF UNMANNED SYSTEMS AND
TECHNOLOGIES: FA²UST**

A Thesis
Presented to
The Academic Faculty

by

Seth Leon Libby

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Aerospace Engineering

School of Aerospace Engineering
Georgia Institute of Technology
May 2018

Copyright © 2018 by Seth Leon Libby

**FRAMEWORK FOR PRODUCT ARGUMENT ANALYSIS OF UNMANNED SYSTEMS AND
TECHNOLOGIES: FA²UST**

Approved by:

Professor Dimitri Mavris,
Committee Chair
School of Aerospace Engineering
Georgia Institute of Technology

Professor Dimitri Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Jean Charles Domercant
Electronic Systems Laboratory
Georgia Tech Research Institute

Professor Daniel P. Schrage
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Olivia J. Pinon
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Christopher Jouannet
Saab Aerosystems
Saab Group

Dr. Kristian Amadori
Saab Aerosystems
Saab Group

Date Approved: January 2018

DEDICATION

*“The bulk of mankind is as well equipped for flying as thinking.’
[Jonathon Swift (1711)] Which is now a more hopeful statement than
Swift intended it to be.”*

- Will Durant

ACKNOWLEDGEMENTS

I want to thank my committee - Dr. Dimitri Mavris, Dr. Daniel Schrage, Dr. Jean Charles Domercant, Dr. Olivia Pinon-Fischer, Dr. Jouannet Christopher, and Dr. Kristian Amadori. They offered their knowledge, experience, and time, for supporting me financially and academically throughout my dissertation to ensure I reached my goals. They provided the unwavering patience as I fine tuned my talents to become a better student and engineer. I also would like to thank Dr. Kelly Griendling who advised me on my research throughout my graduate studies.

Also, I want to thank my family, whose support provided the foundation necessary for me to pursue a higher education. Without my grandfather's and grandmother's unwillingness to accept failure or inferior performance, I would not have had the drive within myself to see the completion of this degree. I am grateful to my parents whose constant availability and love provided an unwavering foundation, allowing me to strive for higher education. Finally, I thank my sisters, Alysia and Eden, who provided the competitive spirit and heart I required to become the well-rounded person I am today.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF TABLES	xi
LIST OF FIGURES	xiv
LIST OF SYMBOLS OR ABBREVIATIONS	xix
GLOSSARY	xix
SUMMARY	xix
I INTRODUCTION	1
1.1 Reconfigurability and Commonality Selection in the Systems Engineering Process	3
1.2 Defining Design, Configurations, System Architectures, and Product Architectures	7
1.3 Expansion of Product Architectures	12
1.3.1 Defining Types of Product Architectures	12
1.3.2 Defining Types of Hybrid Product Architectures	15
1.3.3 Summary of the Product Architecture Space	18
1.4 Importance of Product Architecture Selection in the Design Process	20
1.5 Challenges of Selecting a Product Architecture	21
1.5.1 Review of Past Product Architecture Selections	21
1.5.2 Challenges Selecting a Product Architecture Concerning Un- manned Aerial Vehicles	25
1.5.3 Requirements of an Informed Product Architecture Selection	27
1.6 Problem Definition	27
1.7 Overview of Dissertation	30
II BACKGROUND RESEARCH	33
2.1 Current Systems Engineering Process	33

2.1.1	Process Input	33
2.1.2	Requirements Analysis	35
2.1.3	Functional Analysis and Allocation	38
2.1.4	Design Synthesis	43
2.1.5	System Analysis and Control	50
2.1.6	Process Output	53
2.1.7	Requirements Loop	55
2.1.8	Design Loop	55
2.1.9	Verification	56
2.1.10	Systems Engineering Process Summary	59
2.2	Existing Product Architecture Selection Methods	59
2.2.1	Quality Function Deployment (QFD) or Relational-Oriented Systems Engineering and Technology Trade- off Analysis (ROSETTA)	61
2.2.2	Unified Trade-off Environment	66
2.2.3	Customer Demands	67
2.2.4	Robust Concept Exploration Method	70
2.2.5	Variation-Based Platform Design Methodology	72
2.2.6	Multi-Disciplinary Analysis and Optimization (MDAO)	73
2.2.7	Contact and Channel Model	77
2.2.8	Architectural Enumeration and Evaluation	80
2.2.9	Architecture Selection under multiple Criteria and Evolving Needs for improved Decision-making (ASCEND)	81
2.2.10	A Product Family Design Methodology Employing Pattern Recognition	84
2.2.11	Evaluation of Existing Product Architecture Selection Methods	86
2.3	Additionally Required Concepts	88
2.3.1	Past Commonality and Reconfigurability Studies	89
2.3.2	Complex System Definition	104

2.3.3	Difference between Robust Design, Flexibility, and Complexity	106
2.3.4	Software's Impact on the Product Architecture	109
2.3.5	Additional Required Concepts Conclusion	122
2.4	Research Objective	122
2.5	Formation of Research Questions	124
2.5.1	Research Question 1: Establish the Need for a New Product	126
2.5.2	Research Question 2: Define the Problem or Requirements of the New Product	127
2.5.3	Research Question 3: Establish Value of the Product Archi- tecture	128
2.5.4	Research Question 4: Generating Alternative Product Architectures	129
2.5.5	Research Question 5: Analysis of Alternative Product Architectures	129
2.5.6	Research Question 6: Determining Areas of Interest in the Product Architecture Space	131
2.6	Summary of Background Research	131

III FORMULATION OF FRAMEWORK FOR PRODUCT ARCHI- TECTURE ANALYSIS OF UNMANNED SYSTEMS AND TECHNO- LOGIES: FA²UST 133

3.1	Analyzing Customer Needs and Formulating a Product-Based/Customer-Oriented Business Strategy	135
3.1.1	PESTEL Analysis	138
3.1.2	The Five-Forces Model	139
3.1.3	VRIO Framework	142
3.1.4	Value Chain Analysis	143
3.1.5	Selecting Business Strategy	145
3.1.6	Determining Product Needs	149
3.1.7	Checking the Formulated Product-Based, Customer-Oriented Business Strategy	153

3.1.8	Summary of Analyzing Customer Needs and Formulating a Business Strategy	154
3.2	Identification of Product Architecture Selection Drivers and their Impact	154
3.2.1	Investigation of Past Industries	158
3.2.2	Observing the Design Drivers' Impact on the Product Architecture Selection	182
3.2.3	Observing the Product Architecture Selection's Sensitivity to Design Drivers	187
3.2.4	Conclusions from Test Case Determining Drivers Influence on the Product Architecture	188
3.2.5	Formation of Hypothesis 1	189
3.3	Establishing a Valuable Product Architecture	189
3.4	Identification of Methods to Facilitate Generating Alternative Product Architectures	193
3.4.1	Commonality Index	195
3.4.2	Online Reconfigurability Index	195
3.4.3	Offline Reconfigurability Index	196
3.4.4	Example F-14 Tomcat Product Family Indices Breakdown	198
3.5	Evaluating Alternative Product Architectures by Quantifying Desirability, Flexibility, and Complexity	202
3.5.1	Formation of Design Problem	203
3.5.2	Creation of Desirability Metric	204
3.5.3	Creation of Requirement Flexibility Metric	206
3.5.4	Creation of Complexity Metric	208
3.5.5	Formation of Hypothesis 3	212
3.6	Formulation of Method to Identify Architectures of Interest	213
3.6.1	Definition of Pareto Identification	213
3.6.2	Utilization of Impact Mappings to Form Flexible Pareto Frontier	214
3.6.3	Categorization and Properties of Architecture Drivers	215
3.7	Experimental Plan	217

3.7.1	Experiment 1: Testing Predisposed Assumptions in Product Architecture Selection	220
3.7.2	Experiment 2: Testing the Validity of using Qualitative Weightings for various Metrics used to Evaluate the Product Architectures	222
3.7.3	Experiment 3: Validation of Product Architecture Drivers . .	224
3.7.4	Validation of Framework	225
3.8	Formulation of Framework Conclusion	227
IV	DEVELOPMENT OF THE ANALYTICAL MODULES REQUIRED BY FA²UST MODULE	228
4.1	FA ² UST Module Outer Layer Inputs	233
4.1.1	Inputing the Commonality of a Product Architecture	234
4.1.2	Inputing the Online Reconfigurability of a Product Architecture	234
4.1.3	Inputing the Offline Reconfigurability of a Product Architecture	235
4.2	FA ² UST Module Inner Layer	235
4.2.1	UAV Sizing and Synthesis Models	235
4.2.2	Automobile Sizing and Synthesis Models	290
4.3	FA ² UST Module Outer Layer Outputs	294
4.3.1	Calculating a Product Architecture's Desirability	294
4.3.2	Calculating a Product Architecture's Requirement Flexibility	296
4.3.3	Calculating a Product Architecture's Design Complexity . . .	297
4.4	Development of FA ² UST Module Summary	297
V	UNMANNED AERIAL VEHICLE CASE STUDY	300
5.1	Establishing the Need for a New UAV Product	300
5.1.1	UAV Industry External Analysis	301
5.1.2	UAV Manufacturer Internal Analysis	305
5.1.3	Selecting UAV Industry Business Strategy	308
5.1.4	Extracting Customer Needs for New UAV	310
5.1.5	Final UAV-Based, Customer-Oriented Business Strategy . . .	313
5.2	Defining the UAV Design Problem	313

5.2.1	Decomposition of Tasks Required of UAV	317
5.2.2	Relating Design Missions to UAV Configuration	328
5.2.3	Determination of Drivers Relevant to UAV Design	328
5.2.4	Conclusions from Defining the UAV Design Problem	329
5.3	Establishing a “Valuable” UAV Product Architecture	329
5.4	Generating Alternative UAV Product Architectures	333
5.5	Evaluating Alternative UAV Product Architectures	336
5.5.1	Experiment 1: Testing the Evaluation Metrics Results and Conclusions	337
5.6	Final Decision of UAV Product Architecture to Implement	343
5.6.1	Experiment 2: Sensitivity Studies on the Evaluation Metric Weightings Results and Conclusions	345
5.6.2	Experiment 3: Observing the Impact of Requirements on the Product Architecture Results and Conclusions	348
5.6.3	Summary of Experiments 1, 2, & 3	354
VI	CONCLUSIONS	355
6.1	Contributions	356
6.2	Important Insights from the Research	358
6.3	Final Thoughts	360
APPENDIX A	— AUTOMOBILE CASE STUDY	362
REFERENCES	388

LIST OF TABLES

1	System Engineering Specifications and Baselines [42]	54
2	Example UAV Trade Tree Options	58
3	Evaluation of the QFD Method	64
4	Evaluation of the ROSETTA Framework	66
5	Evaluation of UTE Method	67
6	Evaluation of the Customer Demands Method	70
7	Evaluation of the Robust Concept Exploration Method	72
8	Evaluation of the Variation-Based Platform Design Methodology . . .	73
9	Evaluation of Multi-Disciplinary Analysis and Optimization (MDAO)	76
10	Evaluation of the Contact and Channel Model	80
11	Evaluation of the Architectural Enumeration and Evaluation Framework	82
12	Evaluation of the ASCEND Framework	84
13	Evaluation of the Product Family Design Methodology Employing Pat- tern Recognition	86
14	Evaluation of Existing Architecture Selection Methods	88
15	Evaluation of the Degree of Commonality Index	91
16	Evaluation of the Total Constant Commonality Index	91
17	Evaluation of the Product Line Commonality Index	93
18	Evaluation of the Percent Commonality Index	95
19	Evaluation of the Commonality Index	96
20	Evaluation of the Component Part Commonality Index	97
21	Evaluation of the Comprehensive Metric for Commonality	99
22	Evaluation of the Machine Reconfigurability Index	101
23	Evaluation of the Multi-Attribute Reconfigurability Index	103
24	Evaluation of the Previously Developed Commonality and Reconfigu- rability Indices	104
25	Equation 20 A Coefficient Values [81]	112

26	Equation 20 Scaling Exponents (SF_j) and Cost Drivers (EM_i) [45] . . .	113
27	Evaluation of the AFCAA REVIC or COCOMO Model	114
28	Evaluation of the SEER-SEM Model	116
29	Putnam Special Skills Factor [92]	117
30	Evaluation of the QSM SLIM-Estimate or Putnam Model	117
31	Putnam Productivity Parameter [92]	118
32	Evaluation of the Identified Software Models	119
33	Calculating the Total Number of Unadjusted Function Points [84] . .	120
34	Converting UFP to SLOC [92]	121
35	Automobile Industry Architecture Evolution	159
36	Multi-Role Helicopter Industry Architecture Evolution	165
37	US Carrier Fighter Industry Architecture Evolution	170
38	Industry Architecture Evolution	176
39	Product Architecture Selection Drivers Identified in Section 3.2.1 . .	183
40	Test Case and Driver Impact Framework	184
41	Breakdown of Unique Components in the F-14 Product Line	199
42	Components Broken Down by Discipline	210
43	Options of Available Airfoils in the FA ² UST Module	238
44	Calculating the Total Number of Unadjusted Function Points for a Radar [84]	241
45	Calculating the Total Number of Unadjusted Function Points for a EO-IR Sensor [84]	243
46	Calculating the Total Number of Unadjusted Function Points for a GPS [84]	245
47	Calculating the Total Number of Unadjusted Function Points for a INS [84]	247
48	Calculating the Total Number of Unadjusted Function Points for a Communications Subsystem [84]	248
49	Calculating the Total Number of Unadjusted Function Points for a Piston or Turboprop Engine [84]	251

50	Calculating the Total Number of Unadjusted Function Points for a Turbofan or Turbojet Engine [84]	254
51	Calculating the Total Number of Unadjusted Function Points for a Main Wing [84]	258
52	UAV Industry VRIO Analysis	306
53	Technical Requirements of UAV Case Study	317
54	UAV Industry Civil Surveillance Design Mission [43]	319
55	UAV Industry Military-Grade Surveillance Design Mission [43]	322
56	UAV Industry Military-Grade High-Speed Design Mission [43]	326
57	Correlations between UAV Performance and Cost Metrics	330
58	UAV Civil and Military-Grade Surveillance Missions QFD	330
59	UAV Military-Grade High-Speed Mission QFD	331
60	UAV Design Variable Ranges	334
61	First and Second Order Term's Impact on Evaluation Metrics	341
62	Online Reconfigurability's Interaction Term's and their Impact on Evaluation Metrics	341
63	Composition of Components for Final Selected Product Architecture	345
64	Composition of Online Reconfigurable Interfaces for Final Selected Product Architecture	345
65	Late 1970s Automobile Industry VRIO Analysis	368
66	Technical Requirements of Automobile Case Study	374
67	Mid-Sized and Luxury Automobile QFD	377
68	Automobile Design Variable Ranges	380

LIST OF FIGURES

1	Architecture Development [82]	4
2	Example Operational Viewpoint of UAS Operations in the US National Airspace System [61]	5
3	Conceptual Product Design and Development [71]	8
4	Design Process	9
5	Unmanned Aerial Vehicle Configurations	10
6	Reconfigurable Architecture: Porsche 918 Spyder	13
7	Product Family Architecture: Sikorsky UH-60 Black Hawk	14
8	Fixed Architecture: B-2 Bomber	15
9	Online Reconfigurable Architecture: Grumman F-14 Tomcat	16
10	Modular Architectures	17
11	Scale-Based Product Family Architecture: Boeing 737	18
12	Venn Diagram of Qualitative Architecture Space	19
13	USA Defense Budget as Percentage of USA GDP and Tax Revenue [3, 4]	26
14	Systems Engineering Process [42]	34
15	Example Function Flow Block Diagram of UAV SEAD Mission	40
16	Example Simplified Functional/Physical Matrix of a UAV	42
17	Example UAV Product Breakdown Structure	44
18	Example N ² Diagram of a Basic UAV	46
19	Example UAV Trade Tree	57
20	Quality Function Deployment [122]	63
21	Relational-Oriented Systems Engineering and Technology Trade-off Ana- lysis (ROSETTA) [91]	65
22	Customer Demand Method[155]	69
23	Multidisciplinary Feasible Structure [16]	74
24	Individual Discipline Feasible Structure[16]	75
25	Contact Channel Model [15]	78

26	Architectural Enumeration and Evaluation [157]	81
27	Architecture Selection under multiple Criteria and Evolving Needs for improved Decision-making (ASCEND) [58]	83
28	A Product Family Design Methodology Employing Pattern Recognition [59]	85
29	Example Complex System Design Structure	105
30	Flexibility and Robustness as a Function of the System's Objectives and Environment [126]	107
31	The growth of Software in the Aerospace Industry [5]	111
32	This Dissertation's Framework Overview	125
33	Framework for Product Architecture Analysis of Unmanned Systems and Technologies: FA ² UST	136
34	Process to Formulate a Product-Based/Customer-Oriented Business Strategy	137
35	VRIO Framework [125]	142
36	Value Chain Analysis [117]	143
37	Strategic Position and Competitive Scope [125]	146
38	The Capability and Market Size Relational Space	147
39	Origins of Needs for a New Product	150
40	High Customer Power Market	151
41	Low Customer Power Market	151
42	Process to Transform Customer Needs to Functional Requirements . .	155
43	Breakdown of Requirements	157
44	Architecture Driver Magnitude in the Space	186
45	Driver Impact Mappings	188
46	Transformation of Product Architecture Space	194
47	Two Dimensional Representation of Quantitative Architecture Space	194
48	UAV Tree Diagram Outlining Physical Connections of Components and Subsystems (Fuselage Platform)	196
49	UAV Tree Diagram Outlining Physical Connections of Components and Subsystems (Wing Platform)	197

50	Grumman F-14 Tomcat in Architecture Space	198
51	Tree Diagram Outlining F-14 Tomcat Physical Interfaces between Components and Subsystems	201
52	Grumman F-14 Tomcat in Architecture Space	202
53	Desirability	205
54	Overall Evaluation Criteria Creation	206
55	Flexibility	207
56	Complexity	209
57	Theorized Product Architecture Pareto Frontier	214
58	Theorized Driver Impact Categories: Radial and Tangential	216
59	Experimental Plan	219
60	Verification and Validation Plan for Proposed Method	226
61	Definitions of Entities and Objects used within FA ² UST	229
62	Example Planform Shape for Aerodynamic Surfaces	237
63	Example Cylindrical Geometric Entity	238
64	Range-Resolution Model Incorporated into Radar and EO-IR Sensor Subsystems	239
65	UAV Tree Diagram Outlining Physical Connections of Components and Subsystems for a Traditional Tube-Body, Wing, Horizontal Tail, and Vertical Tail Configuration	266
66	UAV Tree Diagram Outlining Physical Connections of Components and Subsystems for a Traditional Tube-Body, Wing, and Horizontal Tail Configuration	266
67	UAV Tree Diagram Outlining Physical Connections of Components and Subsystems for a Tube-Body, Wing, and Vertical Tail Configuration	266
68	UAV Tree Diagram Outlining Physical Connections of Components and Subsystems for a Flying Wing Configuration	266
69	Final UAV Layout (m)	280
70	Drive Cycles Used to Size and Test the Automobiles	293
71	Integration of UAV Inner Layer Models with the Outer Layer of the FA ² UST Module	298

72	Integration of FASTSim Inner Layer with the Outer Layer of the FA ² UST Module	299
73	The Capability and Market Size Relational Space for the UAV Industry	309
74	Civil Surveillance Mission Historical Vehicles	315
75	Military-Grade Surveillance Mission Historical Vehicles	316
76	Military-Grade High-Speed Mission Historical Vehicles	318
77	Function Flow Block Diagram of UAV Civil Surveillance Mission . . .	320
78	UAV Civil Surveillance Mission Functional/Physical Matrix	321
79	Function Flow Block Diagram of UAV Military-Grade Surveillance Mission	323
80	UAV Military-Grade Surveillance Mission Functional/Physical Matrix	324
81	Function Flow Block Diagram of UAV Military-Grade High-Speed Mission	325
82	UAV Military-Grade Surveillance Mission Functional/Physical Matrix	327
83	Design Variable Ranges Historical UAVs	335
84	An Example of an Online Reconfigurable Wing's Impact on the Mattingly Constraint	336
85	The Product Architecture's Impact on Desirability	338
86	The Product Architecture's Impact on Flexibility	339
87	The Product Architecture's Impact on Complexity	340
88	The FA ² UST Framework Decision Facilitator	343
89	Final Decision of UAV Product Architecture	344
90	UAV Product Architecture Selection based on Desirability	346
91	UAV Product Architecture Selection based on Requirement Flexibility	347
92	UAV Product Architecture Selection based on Design Complexity . .	348
93	The Requirements' Impact on Product Architecture Evaluation Metrics	350
94	The Final Driver Impact Mappings for the UAV Case Study	353
95	The Final Driver Sensitivity Mappings for the UAV Case Study . . .	353
96	The Capability and Market Size Relational Space for the Automobile Industry	371

97	Validation of FA ² UST Framework Product Architecture Drivers . . .	382
98	Validation of FA ² UST Framework Evaluation Metrics	384
99	Validation of Automobile Historical Case	386

SUMMARY

Recently, military and civilian entities use unmanned aerial vehicles (UAVs) in many diverse operations because of their low cost, versatility, lack of human element onboard, efficiency, and connectivity. However, challenges exist when developing these platforms. UAVs are highly sensitive to changing technologies. They must satisfy varying customer expectations, experience low levels of demand, and face stringent regulations. UAVs have become feasible due to the recent advancements in electronics technology and loosening of regulations. The electronics market's and government's influence on the industry constantly shifts from disruptive advancements and uncertain policy. As a result, UAV producers have implemented many different product architectures, including reconfigurable and product family architectures, hoping to satisfy all niches of a constantly changing marketplace. Reconfigurable architectures possess system components that can be modified during or after operating, increasing the system's capabilities. Product-family architectures contain elements of commonality with other systems which reduce the production costs. This is not just specific to the UAV industry. The US military's F-35, the US Navy's Littoral Combat Ship, and the automobile industry all vary levels of commonality and reconfigurability to achieve ever increasing stringent requirements. However, there are some unforeseen consequences of implementing these characteristics, including cannibalization of performance from commonality and increased complexity from reconfigurability. As a result of the expansion of choices, a new method is required to assist in the process of selection.

Product architecture selection is a discrete categorical problem that occurs at the beginning of the design process, often before the down-selection of configurations

and the conceptual phase of design. Due to its occurrence early in the process, all decisions made after this are directly affected, and mistakes during the selection are compounded and grow exponentially throughout development and production of the product. Consequences of selecting the wrong architecture include suboptimal performance, cost overruns, loss of customers, and possible restart or scrapping of the product's production. From an extensive literature review, it is apparent there exists no method that thoroughly explores product architecture alternatives. Therefore, a new framework is presented.

The proposed methodology breaks down the process into six steps: establishing the needs of the customer and manufacturer; identifying drivers that influence the decision; establishing metrics to determine the favorability of a product architecture; generating alternative product architectures; developing a means to evaluate alternative architectures; and identifying methods that help engineers make a decision. The product architecture space is qualitative by nature, but a quantitative approach is presented that converts the space into a quantitative one. This approach identifies the proportion of component characteristics in the product architecture. An examination of multiple industries' pasts identifies possible drivers. Strategically selected architectures decompose the architecture space, to provide insights on how alternative product architectures should be evaluated. This analysis provides evaluation criteria based on the product architecture's resilience to changing requirements and complexity. This dissertation tests the new concepts and formulation of a new framework for selecting product architectures, specifically in the UAV and automobile industries.

The framework identifies requirements or drivers that influence architecture selection and provides a way to generate and evaluate alternative architectures. The output of the framework is the composition of component characteristics of the product (the percent commonality and reconfigurability). The benefits of the new framework include increased traceability of the decisions made throughout all phases of

the design process and general insights on the product architecture selection problem.

CHAPTER I

INTRODUCTION

The use of unmanned aerial vehicles (UAVs) is becoming widespread across government, military, and civil operations. UAVs possess “attributes of persistence, endurance, efficiency, and connectivity [which] are proven force multipliers across the spectrum of global Joint military operations [51].” Furthermore, civilian UAV applications are expected to grow at an annual compound growth rate of 7.6 percent, meeting the commercial objectives of deliveries, operations management, asset tracking, mapping, environmental monitoring, and many other surveillance objectives [73, 29]. UAVs have inherent benefits. They are cheaper than conventional, commercial aircraft. They are used in a wide range of mission types, and do not have a human element, hence reducing costs and risk. The reduced costs and risk lead to the emergence of new uses and applications for unmanned vehicles. As the market expands, diverse market niches will emerge. However, this developing industry has many external factors that influence its evolution, including economic, legal, political, and technological constraints.

In today’s budgetary environment, government customers are under increasing pressure to develop cost-effective, timely products that require designs with greater capabilities to meet multiple mission requirements and objectives [51]. In the civilian industry, there were 456 major worldwide UAV manufacturers trying to capture a \$4.5-billion market in 2016 [73, 93]. Of the \$4.5 billion, \$1.6 billion was generated by commercial or civilian sales. With DJI (Dajiang) Innovations controlling 70-percent of that market, the remaining manufacturers are competing over the remainder of the market, making margins extremely thin. As a result, many of the manufacturers

no longer design vehicles for a single mission or market niche but rather multiple missions and roles to try to capture as much of the market as possible. However, greater capability often leads to increased complexity, inducing higher development and production costs [20]. Though UAVs are cheaper than conventional aircraft, UAVs are still constrained by these economic factors, requiring low acquisition and operating costs.

In 2012 and 2013, the Federal Aviation Administration initiated the process to integrate UAVs into the United States airspace [52, 37, 36]. Though the civilian UAV market is not completely open, the FAA grants special exemptions to approved services in the United States [2], and the European Union plans to have UAVs integrated into its airspace in 2019 [37, 36]. There are still challenges in determining the best regulatory frameworks for the UAV industry, making the FAA and Eurocontrol slow to open up the market. As a result, performance and market requirements are still evolving.

Unmanned systems are heavily reliant on their electronic subsystems, driving the operational performance, capability, and cost of the system. The subsystems replace a human presence with digital senses (sensors). The subsystems are highly sensitive to the rate of change of the electronics industry. The rapid pace of technology and electronic evolution hastens the obsolescence of the sensors, and consequently that of the UAVs as well.

The tight fiscal constraints, uncertain status of regulations, market deregulation, and technology evolution drive system engineers to develop creative solutions. These solutions include various forms of commonality and reconfigurability in UAV design to satisfy the cost and capability requirements. Commonality is the possession of the same components or attributes across two or more concurrently developed entities [142]. Reconfigurability is the ability of a system or design to rearrange its composition of elements to modify how the system behaves or performs [111]. The selected

level of commonality and reconfigurability must meet the present and evolving needs of the customer, and the needs usually reflect market forces, commercial, or military strategies. Selecting the appropriate level of commonality and reconfigurability is essential because of its impact on the cost, performance, and business model of the product offered to its customers and eventually on the manufacturer’s competitiveness and market dominance.

1.1 Reconfigurability and Commonality Selection in the Systems Engineering Process

Implementation of reconfigurability and commonality is a subset of systems engineering. Systems engineering is defined as “an interdisciplinary engineering management process that evolves and verifies an integrated, life-cycle balanced set of system solutions that satisfy customer needs [42]” where a system is “an integrated composite of people, products, and processes that provide a capability to satisfy a stated need or objective [42].” Applied to UAV design, a system is an integrated composite of physical components and software control logic that provide a capability to satisfy a stated need or objective. Implementation of reconfigurability and commonality determines the relations and characteristics of or between the components within one or more systems. In systems engineering, architectural layers help designers organize the design problem at hand. These layers assist in determining capability, operational, service, and systems characteristics [82, 44, 42, 66]. Breaking down the layers can help provide the scope of the systems engineering process that applies to determining the composition of reconfigurability and commonality.

The *Handbook of Systems Engineering and Management* [82] defines and identifies four of these layers: the dynamic operational, functional, physical, and technical architecture layers. It is important to understand that these layers do not represent the product but are system engineering processes that break down the problem to allow designers to make informed decisions throughout the design process. Furthermore,

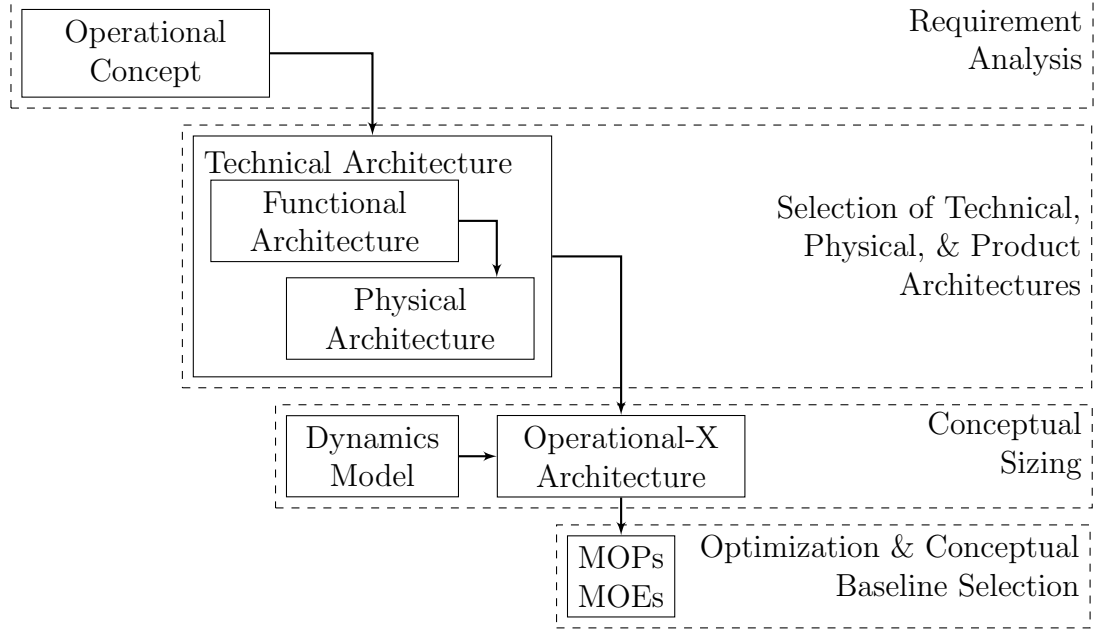


Figure 1: Architecture Development [82]

these layers are implemented by designers in different ways to facilitate designing a system and implementing various levels of commonality and reconfigurability. In the traditional systems engineering process, engineers choose the levels of commonality and reconfigurability through their intuition or experience, often without considerate analysis [82]. Figure 1 shows system engineers apply the layers during a design process.

The dynamic operational architecture is “a description of how the elements operate and interact over time while achieving the goals [82].” It describes how the system in development interacts with the environment and other systems during operations to achieve any capabilities defined by the customers or designers. It is essentially an applied version of the Department of Defense Architectural Framework (DoDAF) defined Operational Viewpoint, outlining connections and relations among systems or entities enabling combined capabilities [44]. Figure 2 provides an example Operational Viewpoint of UAS Operations in the US National Airspace System. It outlines the entities and systems and the connections among them. The connections show where

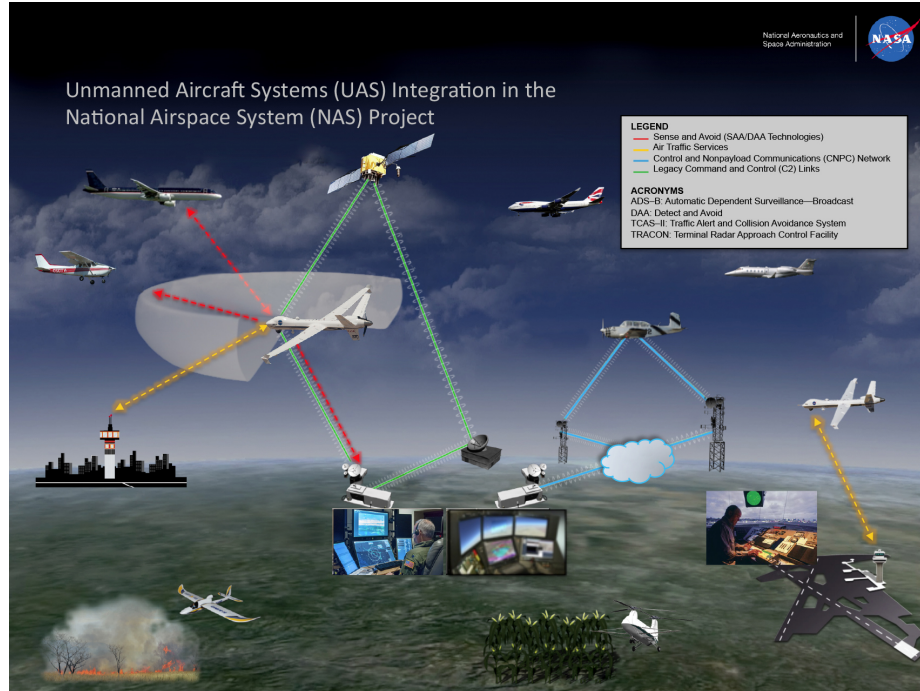


Figure 2: Example Operational Viewpoint of UAS Operations in the US National Airspace System [61]

resources are exchanged so the collaborative systems can achieve the capability of monitoring weather and forest fires without interfering with commercial aircraft operations. A generic dynamic operational architecture defines the systems and their interconnections. It demonstrates the relationships and logical flow between entities and detail how these entities behave together.

The functional architecture layer is “a partially ordered list of activities or functions that are needed to accomplish the system’s requirements [82].” In a nutshell, the functional architecture determines how the system interacts with the missions or tasks assigned to it. The product must be able to perform one or multiple tasks. Each of these tasks can be broken down into a step-by-step process which allows the designer to identify the components, functions, or capabilities that are required from the proposed system. By conducting this exercise, the designer can immediately eliminate options for the product architecture since some components and characteristics will not achieve the requirements set by the customer or developer. Four models make

up the functional architecture: the activity, data, rule, and dynamics model [82]. The activity model breaks down the tasks a system must perform into discrete steps through a function flow block diagram, displaying a series of discrete events. The data model simulates the events usually using simplified modeling, simulating the mission, operations, or interactions that take place in the operational architecture. If there are conditions or options the system might encounter during its task then a rule model provides the logic chains which outline how the system should behave, deploying a trade tree to outline all options. Finally, the dynamics model controls the flow of the simulation determining when to move on to the next element of the task and what information must be passed on.

The physical architecture is “a node-arc representation of physical resources and their interconnections [82],” often displayed as an N^2 diagram. The diagram shows which components interact with each other.

The technical architecture is “an elaboration of the physical architecture that comprises a minimal set of rules governing the arrangement, interconnections, and interdependencies of the elements, such that system will achieve the requirements [82].” In summary, the technical architecture is the combination of the functional and physical architectures, displaying how the system and its components interact with each other and the requirements.

Implementation of commonality and reconfigurability occurs in the Systems Viewpoint or the technical/systems architecture. The characteristics of commonality and reconfigurability influence the interconnections and interdependencies among the components within one or multiple systems. Since the characteristics of commonality and reconfigurability drive production and business relationships, the implementation is not only a part of systems engineering but also product development. Hence, the characteristics are a part of the **product architecture** which combines systems engineering principles with product development considerations.

1.2 Defining Design, Configurations, System Architectures, and Product Architectures

Product development outlines the process of defining the key elements involved in the development, production, and supporting of a new product. In the content of this work a product is defined as “something that is made to be sold, especially something produced by an industrial process [6].” A product can be designed separately or concurrently with multiple variants, in a product family or product line, to satisfy the demand of one or multiple market niches [22, 71].

Figure 3 displays the five domains in product development [71]: customer, functional, design, process, and logistics. The domains pass requirements, variables, and constraints to each other causing coupling and interactions whose strength increases with the system’s complexity. The customer domain is composed of the market’s, customers’, or stakeholders’ stated or unstated needs or desires [89, 71]. Examples of parameters in this domain include customer satisfaction and intimacy. The functional domain is composed of requirements that can be defined as “statement[s] that identif[y] system [...] characteristic[s] or constraint[s], which [are] unambiguous, [and] can be verified [151].” Examples include design mission(s), performance characteristics, technologies, and cost(s) - acquisition, operations, and logistics. The design domain can be defined as “qualitative and quantitative aspects of physical and functional characteristics of a component, [...] product, or system that are input to its design process [1].” Examples of design parameters are usually component characteristics such as wing area or engine horsepower/thrust. The process domain can be defined as “any of those varying operational and physical conditions associated with [manufacturing] operation [108].” Examples of process variables include manufacturability (cost, time, and quality) and process set up. Finally, the logistics domain is defined as “[those variables associated with] the organization of supplies, stores, quarters, etc. [131].” Examples include supply chain management, postponement, and results

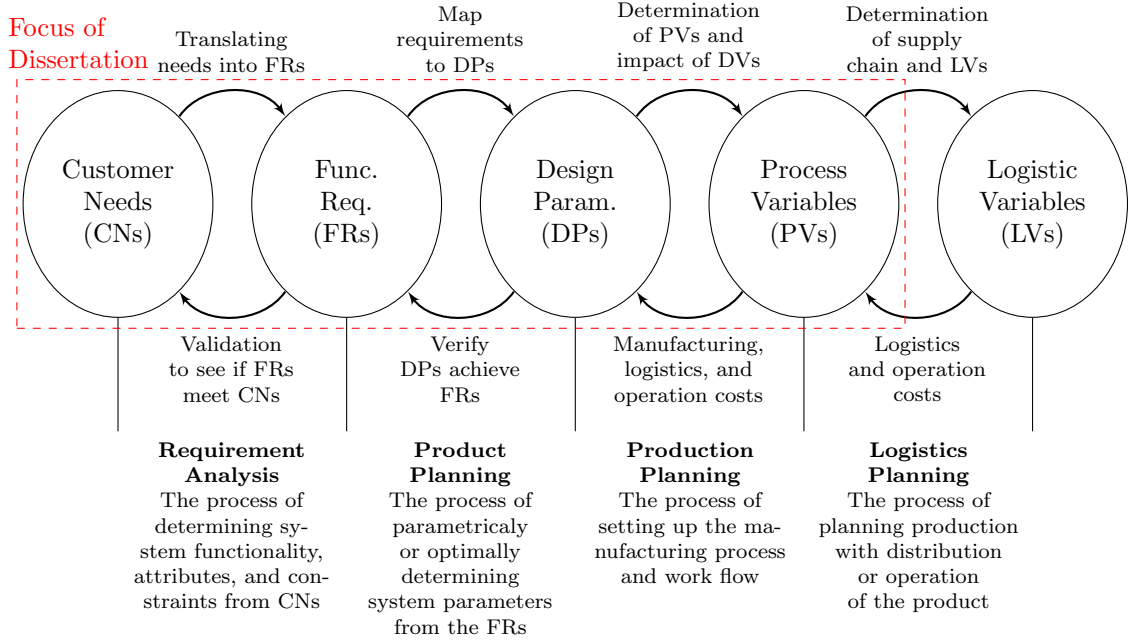


Figure 3: Conceptual Product Design and Development [71]

of operation analyses.

The process by which engineers define the variables and verify that the product meets the customer’s needs is the design process. In the aerospace community, the design process is “the intellectual engineering process of creating on paper a flying machine that either meets certain requirements and performance objectives or explores new concepts, technologies, and innovation [18].” During the process, designers translate an idea into a tangible object by defining the variables that outline the dimensions, structures, and performance characteristics of the system. As the product progresses through the process, designers gain more information about the product.

Aircraft design consists of three distinct phases: conceptual, preliminary, and detailed design, as seen in Figure 4. Conceptual design is the first step, where the designer conducts parametric or optimization-based analysis, determining the size, shape, and performance of the system and its components. Furthermore, the designer can analyze trade-offs amongst requirements, technologies, performance, and approximated costs. Preliminary design is the second phase where the designer locks

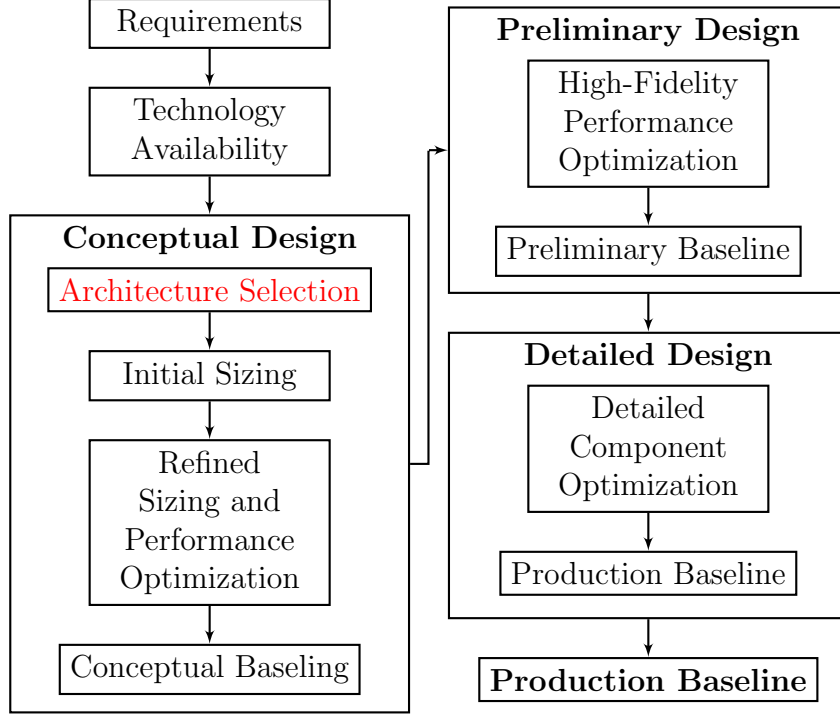


Figure 4: Design Process

in the main features of the system including the components’ explanatory design variables. Higher fidelity analyses are used to make minor changes and help determine economic feasibility. Detailed Design is the third and final phase where production of the system starts, and final adjustments occur.

During the process of designing a product, engineers begin to form its shape and the layout. According to Anderson, a configuration layout “[...] is a drawing of the shape and size (dimensions) of the airplane [18].” Raymer defines a design/configuration layout as a means to “[...] depict major ideas which the designer intends to incorporate into the actual design [121].” Raymer uses design and configuration layout interchangeably in his work. Often, the choice in configuration implies which types of components the vehicle has, since configurations’ names at times incorporate the types of components in the design. A good example is a quadrotor. When the architect adds more rotors, the configuration becomes a hex or octa-rotor.



Figure 5: Unmanned Aerial Vehicle Configurations

Figure 5 shows four configurations: tube-and-wing, single-rotor, blended-wing-and-body, and quadrotor, each being a description of its physical appearance.

Throughout the systems engineering world, there is a term called the physical architecture, which “depicts the system product by showing how it is broken down into subsystems and components [42].” For this work, a hybrid definition was created to summarize all of these points: *a configuration is a conceptual layout of the proposed system outlining key components and subsystems incorporated into the design.* The configuration consists of all the discrete elements of the product. Components are constituent parts of a design [8] and are added to the design to satisfy customer-defined or designer-derived functions [42]. Sometimes, the components cannot achieve a function alone, but can when interacting with others. For example, in a conventional aircraft, a wing, engine, or fuselage alone cannot carry a payload from one point to another, but together the product’s purpose is achieved. Therefore, the configuration defines the composition of the system but does not describe how the components interact and function together to achieve a task. How the components and subsystems interact is the system architecture.

In academia, engineers define a system architecture in multiple ways. David Wallace, a professor at MIT in the Mechanical Engineering Department, defines it as “the arrangement of functional elements into physical chunks which become the building blocks for the product or family of products [152].” Karl Ulrich, a professor at the MIT Sloan School of Management, defines it as “the scheme by which the function of a product is allocated to physical components [145].” Edward Crawley, a professor at MIT Engineering Systems Division, defines it as “an abstract description of the entities of a system and the relationships between those entities [40].” Chris Paredis, in his robotics work, defines the system architecture as the organization of either physical components or software elements as a role-based structure which define how each piece interacts with another [46, 133, 47]. Finally, according to the US Department of Defense, a system architecture “identifies all the products (including enabling products) that are necessary to support the system and, by implication, the processes necessary for development, production/construction, deployment, operations, support, disposal, training, and verification [42].” In the context this research, a system architecture is defined as *the arrangement or allocation of components and their relations amongst each other that form an integrated solution and resultant capability*. By selecting an architecture, the designer is choosing the product’s components and how they interact and function together. Therefore, the system architecture consists of the configuration and the relations or interactions amongst a system’s components and subsystems.

Often, a manufacturer designs and develops individual components by separate entities, either internal departments or external contractors [125]. The divisions must interact with each other to create a coherent product. Therefore, interactions amongst components occurs not only in the performance domain but also in the production/business domain. The system does not define these relationships, especially during concurrent production of multiple products.

A product architecture consists of one or more systems designed using a common product line. A product line is a group of related products usually sold by the same business or entity. In the context of this research, a product architecture is defined as *the arrangement or allocation of components and their relations amongst multiple systems that form an integrated solution and resultant capability across a product line*. A product can be defined by its configuration, system architecture, and product architecture. In summary, the main difference between a system architecture and a configuration is the functionality versus the type of components used in the design. The difference between a product and system architecture is the relationships among one or multiple systems.

1.3 Expansion of Product Architectures

Originally, the product architecture space was only composed of “fixed” product architectures, but “the rate of technological advancement and complexity of these systems has increased the design configuration [and product architecture] trade space[s] [21].” Implementing commonality and reconfigurability within a product caused two qualitatively defined product architecture subspaces to emerge. These two subspaces are reconfigurable and product family product architectures.

1.3.1 Defining Types of Product Architectures

Currently, three types of architectures exist: fixed, re-configurable, and product family. A reconfigurable architecture allows a design to reconfigure itself based on mission requirements [129]. As such, reconfigurability allows for the physical components in a configuration to be swapped, changed, or rearranged. Reconfigurability allows a product to achieve better performance and broader capabilities. Reconfigurable architectures can be broken down further into two types: online and offline. Online and offline reconfigurable architectures are considered hybrid architectures (defined in Section 1.3.2). Figure 6 displays a Porsche 918 Spyder which is designed to be a

high-performance hybrid automobile. It has a reconfigurable engine mode that changes the electrical and mechanical responsibilities. The driver can change modes while operating the vehicle. The car also can use modular breaks, engines, and electronics. The Porsche 918 collection of components defines it as a reconfigurable architecture.



Figure 6: Reconfigurable Architecture: Porsche 918 Spyder

Reconfigurable architectures allow the design to be more flexible to changing requirements or various mission scenarios the product will operate under. However, the trade-offs that emerge when implementing reconfigurability are not well understood and any hypothetical gains in performance could be offset by emerging and unforeseen consequences.

A product family is “a group of related products that is derived from a product platform to satisfy a variety of market niches [132],” or “a set of products that share a unique number of common components, [processes,] and functions with each product having its unique specifications to meet demands of certain customers [116, 94].” In general, product family architectures enforce commonality to reduce costs by decreasing the number of processes required to produce the product. Similar to reconfigurable architectures, product families can be broken down into two types: scale-based and modular. Scale-based and modular architectures are considered hybrid architectures (defined in Section 1.3.2). Figure 7 displays the Sikorsky UH-60 Black Hawk product family, designed to be multi-role helicopters where each member specializes

in specific tasks. The family shares a common platform whose design variables are “stretched” depending on the family member. The vehicles also contain modular subsystems that can be swapped or easily upgraded.

Product family architectures reduce the number of processes involved in the production, reducing development and production costs. However, enforcing common components across multiple systems can often cannibalize performance [132, 110].



Figure 7: Product Family Architecture: Sikorsky UH-60 Black Hawk

A fixed architecture is a product that contains no components that are either common or reconfigurable. It is designed to satisfy specific requirements and achieve robust performance across multiple missions. Figure 8 displays a B-2 Bomber, designed for long-range, stealth-bomber missions. The design is rarely used for other missions, it does not change, and has no product family members. It is defined as a purely fixed architecture. Designers implement fixed architectures to optimally or robustly satisfy one mission. Therefore, the trade-off among performance and cost was conducted to maximize the value to the manufacturer and customer.



Figure 8: Fixed Architecture: B-2 Bomber

1.3.2 Defining Types of Hybrid Product Architectures

Hybrid architectures are a combination of architectures which include online reconfigurable, modular, and scale-based architectures. An online reconfigurable architecture is a hybrid between a fixed and reconfigurable product architecture. Online reconfigurability allows the physical components in a configuration to rearrange or change their orientation during operations, increasing the performance of the product or expanding the types of missions the vehicle can perform. However, the ability to morph the shape of the aircraft or rearrange the physical components requires extra subsystems and structures to control the motion of the structures. Figure 9 displays a drawing of a Grumman F-14 Tomcat, designed with the capability to launch from an aircraft carrier but still reach supersonic speeds. The design has a wing that changes its sweep and indirectly its aspect ratio during flight, giving it the ability to satisfy both extreme requirements. Online reconfigurable architectures increase the performance of a design that operates in multiple conditions. The ability to change requires larger structures to handle changing loads and more electronics/hydraulics to control the movement. As a result, the product's complexity increases and flexibility decreases.

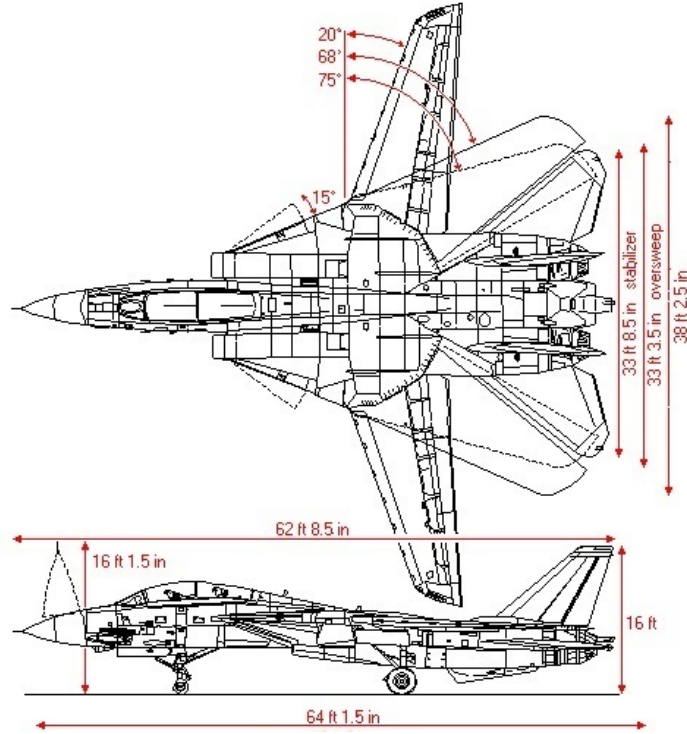


Figure 9: Online Reconfigurable Architecture: Grumman F-14 Tomcat

Offline reconfigurability gives the system the ability to swap components at the conclusion of operations. The ability to swap components requires stringent standards and oversizing of the overall product, which could offset the hypothetical gains in performance. The standardization of interfaces between components is a common trait of modular product architectures. Therefore, modular and offline reconfigurable product architectures possess the same characteristics. In the context of this work, they have been combined into one hybrid architecture: a modular architecture. A modular product architecture allows components or subsystems to be swapped when the product is offline or between operations. The ability to change the physical components increases the capability of the vehicle or the number of missions the vehicle can perform. Modular product families contain “components [which] are parts, structures, or subsystems that are self-contained and designed with specific characteristics that allow them to be repeatedly removed and replaced during the operational lifetime of the vehicle [110].” The ability to swap components require

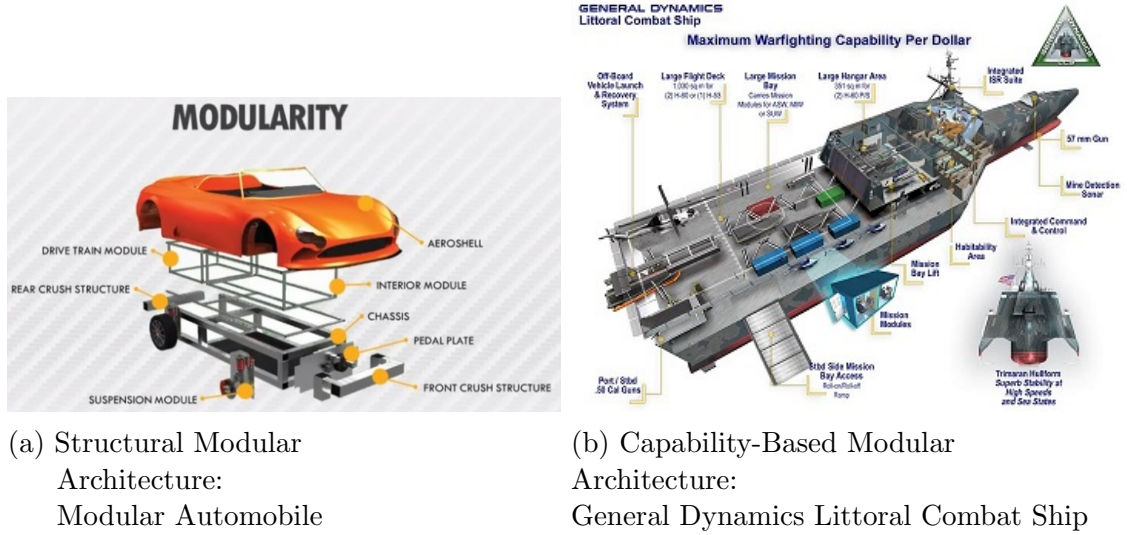


Figure 10: Modular Architectures

strict standards defining the interface among different parts so swapping requires no redesign. Modular product families can again be broken down into two sub-types: structural and capability-based. Structural, modular product families contain components that are added or removed from the design, changing the physical structure of the vehicle. Capability-based, modular product families contain subsystems that are added or removed, hence changing the vehicle's capability. Figure 10 displays a modular automobile (Figure 10a) and the General Dynamics Littoral Combat Ship (Figure 10b). The modular car, which represents many of the products in the motor vehicle industry, has interchangeable components that modify the overall structure of the vehicle, changing its purposes or capabilities. The General Dynamics Littoral Combat Ship has mission specific modules, changing what the vehicle can do. The implementation of a modular architecture increases the product's list of capabilities, making the product more flexible. However, the greater number of components increases the complexity, adding considerations and constraints during the design and development stages. Also, the inclusion of common components can cause a reduction in performance.

Scale-based product families use a method of design where "one or more scaling

variables are used to ‘stretch’ the platform in one or more dimensions to satisfy a variety of market niches [132].” A product family tends to be manufactured concurrently, which reduces the manufacturing cost. The scaling of one or more variables increases the product family’s diversity of capabilities. Figure 11 displays the Boeing 737 product family. The wing and empennage are common among all the designs; however, the fuselage length has been stretched, and various engines are used depending on the design. Scale-based product families increase the product’s list of capabilities, creating multiple systems with common components and production processes. However, by increasing the flexibility and the product’s complexity is increased, due to the number of constraints added to the design process.



Figure 11: Scale-Based Product Family Architecture: Boeing 737

1.3.3 Summary of the Product Architecture Space

Figure 12 shows the space projected as a Venn diagram. The circular subspaces represent the fixed, reconfigurable, and product-family architectures. The scale-based, online, and offline hybrid architectures are captured by the intersection of the circular subspaces. Figure 12 represents the component characteristics qualitatively, dividing

the space up into three categories and three hybrids. An area that is not identified is the center portion which combines all of the possible characteristics. Most products incorporate characteristics off all three product architectures. For example, an automobile has common components shared by other products (common chassis), fixed components specific to the specific component (outer frame), and reconfigurable components that change the products performance during or between operation (gear boxes or wheels). It is important for engineers to understand early on in the design process what proportion of each characteristic should be implemented.

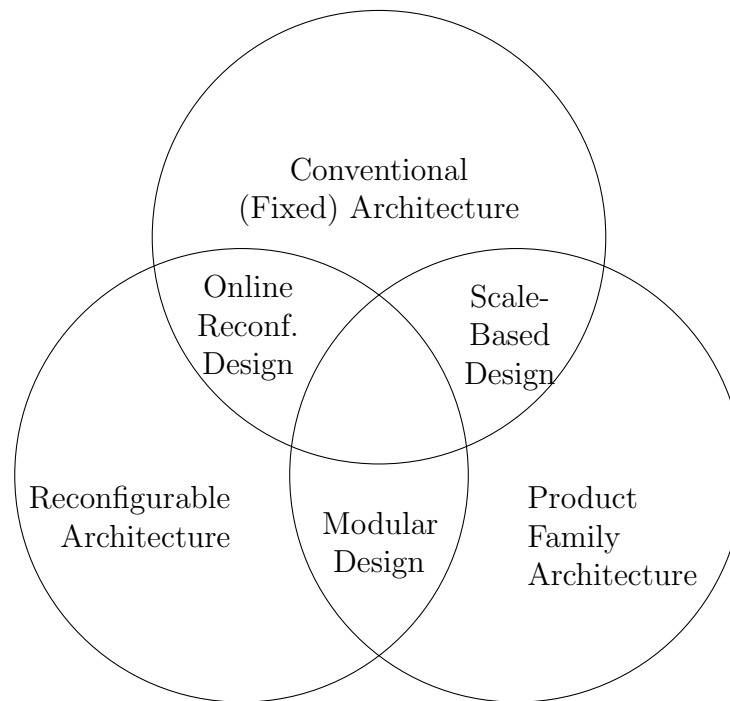


Figure 12: Venn Diagram of Qualitative Architecture Space

Furthermore, the product architecture only defines the characteristics of the components, not the performance or cost of the product or its configuration. The design and configuration (physical architecture) define these product attributes.

1.4 Importance of Product Architecture Selection in the Design Process

The selection of a product architecture impacts all phases and aspects of the design process. Traditionally, systems engineers make decisions regarding the levels of commonality and reconfigurability desired on their knowledge or experience [121], often without fully exploring the entire architecture space. System engineers methodically manage the decisions regarding the requirement analysis, design, technical management, operations, and retirement of a system. Since product architecture selection occurs so early in the design process and impacts all of the decision that follow, the project's success heavily depends on the system engineers' decisions. If the engineers choose a poor product architecture, the product could experience:

- Sub-optimal functional performance
- Cost overruns in production or operations and support
- Long term loss of customers
- Possible restart or scrapping of the product's design, development, or production

Two examples of poor product architecture selection are the F-35 Lightning and the Pierce-Arrow Motor Car Company. The F-35 Lightning's architecture has demanding requirements and advanced technologies. The selected product architecture contributed to the program's acquisition cost and manufacturing time to increase 72.5% and 104.3% respectively from 2008 to 2013 [138]. In the 1920s, the Pierce-Arrow Motor Car Company decided to pursue luxury vehicles targeting the upper-class market. The company's luxury-based, unstandardized product architectures were not flexible to changes in the market that occurred during the Great Depression, causing the company to go out of business [100].

1.5 Challenges of Selecting a Product Architecture

1.5.1 Review of Past Product Architecture Selections

The decisions that determine the composition of commonality and reconfigurability are incredibly important in determining the success of a designer. Over the years, systems engineers have implemented various levels of commonality and reconfigurability to meet various capability and economic constraints. Reviewing some of these historical cases will provide more information about the problem and help identify what information is required to select a product architecture.

1.5.1.1 Grumman F-14 Tomcat

The F-14 Tomcat was designed to conduct air-to-air, precision air strike, and naval air defense missions while being launched from a naval aircraft carrier. To complete the required missions, the F-14 had to be capable of defeating existing fighter aircraft. The aircraft was required to achieve maximum speeds around Mach 2. Therefore, an online reconfigurable wing was used to take off from a carrier and reach those speeds. The wing sweep varies from 20° to 68° during missions [13]. Wing sweep at 20° performs better at subsonic speeds, while sweep at 68° performs better at supersonic speeds. The reconfigurable wing also helped with storage on the carrier, since the wing can sweep back to 75° when the aircraft is not operational. These benefits also came at a cost. Variable sweeping wings increase the structural weight needed to support and manipulate the wings and require more power consumption and control computers [54]. Overall, the online reconfigurable architecture selection gets rid of the carrier takeoff and high speed dash requirements interaction but increases the coupling between wing design and structures. Its primary feature is its online reconfigurable wing. The system has been upgraded over the years creating a product family by giving it some commonality among variants. Therefore, the F-14 Tomcat is primarily described as an online reconfigurable design.

1.5.1.2 Boeing 737 Product Family

The Boeing 737 is a scale-based product family. In the 737 MAX family, the most recent, the wing is common, but designers stretched the fuselage among all three designs in the product family (the 737 MAX 7, 8, and 9) [25]. The various fuselage lengths allow for the different designs to carry more or fewer passengers. Commonality among the different designs tends to reduce development and manufacturing costs, however its commonality characteristics slightly reduce performance since it has the same engines and empennage across the designs in the product line. The stretching of the fuselage length is the product line's main characteristic. Thus, it is primarily described as a scale-based design.

1.5.1.3 Sikorsky UH-60 Black Hawk Product Family

The UH-60 Black Hawk is a multi-role helicopter product family platform developed by Sikorsky [96]. Variants include the SH-60 Sea Hawk, the HH-60J Jay Hawk, and the HH-60G Pave Hawk. Each corresponds to a different design mission. The product family has elements of scale-based and module-based architectures. The commonality among the designs has drastically reduced the costs of manufacturing and development, but again, the common platform causes some drops in performance. Its dominant characteristics are its commonality and offline reconfigurability due to its combination of scale-based and module-based architectures. Therefore, the UH-60 is primarily described as a generic product family.

1.5.1.4 Lockheed Martin F-35 Lightning II

The Lockheed Martin F-35 Lightning II is currently designed to be the next generation fighter for the US Air Force, Navy, and Marine Corps [83]. Also, it uses multiple outside manufacturers to produce some of the lesser components in the product line. The stakeholders involved in the design demand very different qualities of the product line. Each branch of the military has a corresponding vehicle. The Navy requires its

corresponding vehicle to have short takeoff and landing (STOL) capabilities. The Marines require vertical takeoff and landing (VTOL) capabilities due to the variety of missions the branch conducts. Lockheed Martin is concurrently designing all of the vehicles since they share common components. The VTOL and STOL capabilities require an online reconfigurable engine that operates when activated by the pilot. It incorporates a center-oriented fan that operates during these mission segments. Since the F-35 is a next generation fighter, the vehicle has state-of-the-art electronics and materials to give the pilot maximum situational awareness and reduce the vehicle's detectability. Also, this design is extremely complex due to concurrent design, contradictions among requirements, and implementation of state-of-the-art technologies. As a result, the F-35 program has become costly and time-consuming to develop [138].

1.5.1.5 Summary of Past Product Architecture Selections

Often, the systems engineers implement reconfigurability and commonality to achieve many of the performance and fiscal requirements. Also, systems engineers implement a product architecture to reduce the coupling among design variables or interactions among the requirements.

For example, the F-14 uses a online reconfigurable wing to reduce the interactions from the aircraft-carrier takeoff and supersonic speed requirements. Another example, the F-35 uses a combination of reconfigurability and commonality to achieve the many stakeholder needs. However, the implementation of the product architecture has some consequences. For example, the online reconfigurable wing forces the F-14 to adopt more structures and subsystems to handle the movement of the wing.

The Boeing 737 product family was a part of a business strategy to provide a commercial aircraft platform that provided multiple variants that are designed to transport various number of passengers over various ranges. The inclusion of common engines and empennage helps to reduce the number of processes during production,

reducing the cost of equipment required to produce the product family. Though the commonality reduces the cost of producing the product family, the common parts must be designed with respect to each variant. The concurrent design considerations can add constraints to common parts, decreasing the amount of design freedom during the product line's development.

The Sikorsky UH-60 multi-purpose helicopter product family utilizes scale-based and modular characteristics to provide multiple vehicles that can achieve the multiple tasks required by various customers. Similar to the Boeing 737 product family, designers stretched some of the dimensions of each family member to allow it to carry various payloads and perform differing tasks. Also, the UH-60 utilizes multiple mission-specific payload packages to expand the capabilities of the product line. Again, designers must add constraints and considerations so the common components and subsystems can be integrated into each system.

The F-35's multiple stakeholders demanded many different requirements and capabilities in the product line. These requirements created constraints on the designs and any design changes that were made throughout the product's development had a significant impact on the components incorporated in the designs. There were only a few customers so the customers had significant influence in the product line's development. Therefore, as the stakeholders' decisions changed throughout the F-35's development, costs exploded [138]. As each time a stakeholder's demands changed, modifications reciprocated throughout the design due to the number of constraints limiting the design space.

All of the examples identified in this section stress the need to understand how the requirements and the relations among the components impact the choice of product architectures.

1.5.2 Challenges Selecting a Product Architecture Concerning Unmanned Aerial Vehicles

The Silicon Valley revolution in the 1980s and 1990s was a catalyst for emerging unmanned systems. STEM programs and venture capitalism provided the catalyst for increasing computational speeds, reducing the size of the computers and electronics, and reducing power requirements [31]. The new technologies made UAVs feasible by reducing the size and energy consumption of the subsystems required for autonomous flight. The catalyst technologies continue to evolve at a rapid pace, causing the electronics to become obsolete rapidly.

Currently, government programs require systems to be more affordable [51]. Because of the global reduction in defense spending, governments have continued to search for programs that reduce costs but remain effective. Figure 13 displays the United States' Defense Budget as a percentage of tax revenue and GDP. Since the formation of the North Atlantic Trade Organization (NATO), the United States has reduced the percentage of wealth spent on Defense, as illustrated in Figure 13. (Data collected from SIPRI and the US Department of Commerce: Bureau of Economic Analysis [3, 4]). Since UAVs are beginning to integrate into civil operations, a UAV's price must reflect the market and business strategy. As a result, fiscal concerns must be considered in the UAV industry, as is the case for most industries.

Globalization of world economies has caused businesses to expand beyond national borders. Their products can be found all over the world in diverse environments, causing the system to be designed to operate in extreme temperatures, sea conditions, or dusty/dirty conditions. The diverse environment requirements drive reliability, operation, and support costs of the systems adding additional constraints to the design and increasing the product's complexity.

In the past, engineers designed aircraft for one purpose or mission, such as achieving a given Mach number, carrying a payload over a given range, or partaking in

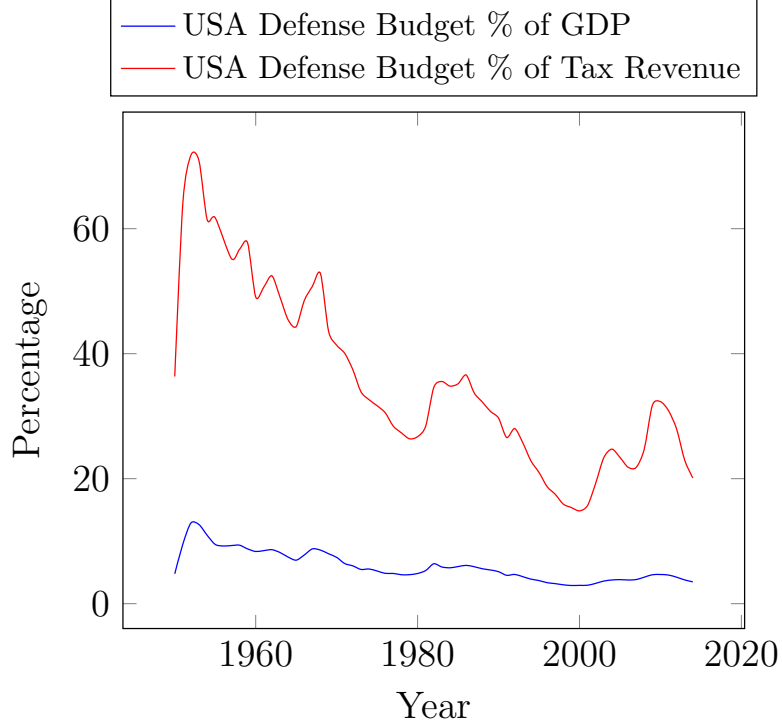


Figure 13: USA Defense Budget as Percentage of USA GDP and Tax Revenue [3, 4]

air-to-air combat at a given altitude. Though designed for specific requirements, customers often use the aircraft in other missions where their performance lacks. As a result, designers implemented capability and robust-based principles to reduce inefficiencies in design. The system's capability is the system's overall mission effectiveness, and robustness is the system's ability to perform under multiple environmental conditions consistently.

The exploration of past product architecture selections and the analysis of the UAV industry show product architecture selection is complicated for the following reasons:

- Uncertain markets and customer needs/requirements
- The rapid pace of technology changes and obsolescence
- Increasingly tighter fiscal constraints and requirements for affordability
- Rapidly evolving operating environments

- Greater emphasis on multi-mission or joint capability across different entities

Due to the numerous economic and performance requirements that constrain the design space, engineers can find capability and robust design principles limited. Therefore, engineers add reconfigurable and commonality characteristics to the design. However, the impact of the product architectural decisions is often not well understood, leading to unexpected design challenges.

1.5.3 Requirements of an Informed Product Architecture Selection

The exploration of past product architecture selections and the analysis of the UAV industry provide insights on what systems engineers require to make an informed decision regarding the levels of commonality and reconfigurability to be implemented. The product architecture has a considerable impact on the production and management characteristics of the overall product line. Therefore, it is important to understand the way the manufacturer's business strategy relates to the product architecture. From the review of past product architecture selections, the fiscal and functional requirements must be considered. The requirements emerge from the customer's needs which reflect the manufacturer's business strategy. Combined, it is important to create a traceable line of logic between the customer's demands, functional requirements, and implemented product architecture. Finally, the past examples show how systems engineers chose a product architecture often to reduce couplings among design variables and interactions among requirements. Therefore, it is important to understand the sensitivities and relations among the design and implementation of reconfigurability and commonality. All of these considerations combined would provide the necessary information to make an informed decision on the product architecture to implement.

1.6 Problem Definition

As stated in Sections 1.4 and 1.5, systems engineers have to design products that address increasingly more demanding fiscal and capability requirements. The customers

demand lower acquisition and life cycle costs. They also demand that the products to operate in different conditions or fulfill varying tasks. Therefore, engineers find themselves making a decision between reducing cost, increasing performance, or sufficing both. As a result, manufacturers try to reduce cost and increase performance by implementing various forms of reconfigurability and commonality. Because of these emerging complexities, “the design process [is] no longer [...] able to rely on the intuitive expertise of a small number of designers to make the initial down-selections [21].” Therefore, designers implement reconfigurable and product family architectures, without full understanding of their impact. As a result, this dissertation’s problem definition is:

Problem Definition:

A design framework is needed that facilitates manufacturer's product architecture selection process. This framework needs to account for the product line's:

- Relation to the overall manufacturer's business strategy
- Requirements or capabilities that drive the selection of the product architecture
- Sensitivity to these "drivers" over time
- Relation to other product architectures allowing the comparison various alternatives
- Ability to satisfy the market segment's (identified in the business strategy) performance and fiscal needs
- Impact on the internal design dynamics, allowing for a greater understanding of how reconfigurability and commonality impact the relations between components, subsystems, and disciplines

In particular, the formulated method should aid in architecture selection problems, including forecasting the impact of changing requirements. The ability to predict customer's desires aids architects in strategic road-mapping, determining the most beneficial evolutionary path and resource management. The method must also allow for trade-offs to be performed between different product architectures by identifying and managing flexibility and complexity of the design. Finally, the method must provide a means to evaluate product architectures. A reliable and efficient framework will help manufacturer's systems architects make decisions, increase market competitiveness and reduce the risk associated with architecture selection.

1.7 Overview of Dissertation

This dissertation is organized to introduce the reader to the subject matter starting with the background research through the formulation of a new method. Also included are two case studies demonstrating the new framework's benefit. Chapter 2 provides relevant background research including an extensive review of the literature focused on methods leveraged to define the architecture space. Hence, Chapter 2 first reviews the current system engineering process, where the product architecture selection traditionally occurs, often creating the crux of the problem. The review of past product architecture selections analyzes past industries and system's product architecture selections, identifying the reasons driving the decisions. From the observations and insights, additional terms are introduced and defined, driving the requirements of the new framework. Finally, existing methods used in industry and academia are identified and the benefits, and gaps of each are analyzed, creating a set of evaluation criteria for the new framework. Finally, the chapter defines the research objective and questions that drive the formulation of the framework.

Chapter 3 reviews the steps in the new framework. First, the manufacturer needs to define its business strategy which outlines the market and customers the new product should apply to. The market and customer analysis allows the systems engineers to identify the new product's needs. Next, methods are identified that translate the customers' needs to the product's functional requirements. Also, the possible requirements that drive the product architecture selection process are identified. Next, what makes a product architecture valuable needs to be established. Though there are many ways to evaluate a product architecture, desirability, flexibility, and complexity are identified as three key metrics. Weighting among the three metrics allows an overall metric to determine an alternative's favorability. Next, the qualitative architecture space must be converted into a quantitative one to facilitate the generation of alternative product architectures. To evaluate these alternatives, desirability,

flexibility, and complexity must be defined and quantified. These metrics reflect the product architecture's ability to achieve customer demands and manufacturing requirements. Finally, methods to select the final product architecture are introduced. This chapter introduces the UAV product architecture analysis tool FA²UST. Here, the outer elements of FA²UST which include enforcing the indices and calculation of metrics. Following the formulation of the new framework, three experiments are developed to test the validity of the framework. These experiments are conducted in two case studies presented in Chapter 5.

Chapter 4 breaks down the computational capabilities required by the new framework in analyzing a UAV product architecture. This chapter outlines the construction of these internal processes that exist within FA²UST that allow for the mission simulation and constraint calculations.

Chapter 5 presents a case study of a UAV manufacturer. The case looks at a small-sized manufacturing firm that is trying to distinguish itself in the UAV industry. The case looks at the business environment and industry to develop needs and desires of the new product line. From the new product line's needs, design missions are identified that the product line should be able to complete. After setting up the problem, the product architecture options, design variable ranges, and requirement variable ranges are set to create the space to be explored. The results provide data for the experiments which test the assumptions and hypotheses created in the formulation of the method. Finally, a decision is made on the composition of product architectures to implement and the implications of this decision are discussed.

Finally, Chapter 6 summarizes the conducted research and provides overarching conclusions and insights.

Appendix A presents a historical case of a late 1970s American car manufacturer. This case is supposed to be a validation of framework, applying it to a very different industry. The chapter goes through the same process conducted in Chapter 5. Once

the data is collected, the results from the automobile case are compared against the results from the UAV case, providing insights on the implications of implementing certain product architecture characteristics. A final decision is made on which product architecture to implement. This decision is compared to what happened in the industry.

CHAPTER II

BACKGROUND RESEARCH

As stated in the introduction, product architecture selection is a difficult process and has significant consequences for the profitability and success of the manufacturer. Traditionally, engineers approach the problem with logical, qualitative methods. This research identifies these methods and creates a starting point for the research conducted in this chapter.

2.1 Current Systems Engineering Process

Engineers and management use the current systems engineering process to select the product architecture, refine the design, and provide traceable recordings of their decisions. Figure 14 displays the state-of-the-art, systems engineering process.

The process consists of three stages: requirements analysis, functional analysis and allocation, and design synthesis. Within the three phases, there are three feedback loops which ensure consistency and ensure the product meets its intended goals or purposes. The three loops are the requirements, design, and verification loops. Finally, the process results in a conceptual design with a set product architecture and a traceable line of logic to justify decisions made throughout.

2.1.1 Process Input

Designers and engineers start with a list of inputs for the systems engineering process. These inputs include but are not limited to the following:

- Customer Needs / Objectives / Requirements
- Missions

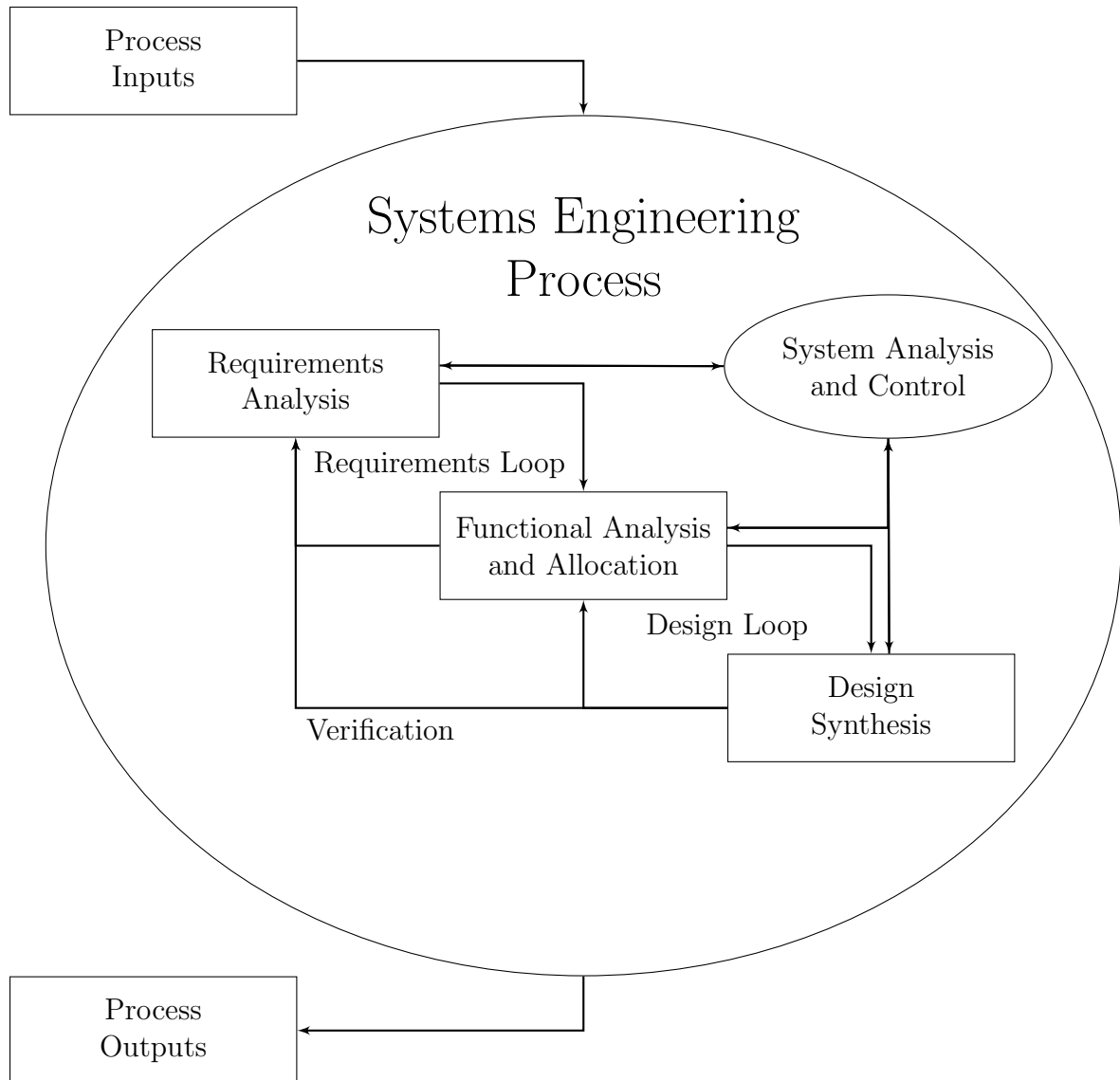


Figure 14: Systems Engineering Process [42]

- Measures of Effectiveness
- Environments
- Constraints
- Technology Base
- Output Requirements from Prior Development Effort
- Program Decision Requirements

- Requirements Applied through Specifications and Standards

The manufacturer and the customers set or derive the process inputs from either market analysis, request for proposals, or requirement engineering. The manufacturer often has three options to develop the needs independently, working with the customer, or receive them directly from the customer. Not all of the inputs are immediately obvious. They require some quantitative and qualitative analysis during the requirement analysis phase to justify the development of a product.

2.1.2 Requirements Analysis

The requirements analysis phase starts with the process inputs and tries to convert them into metrics, capabilities, the product's characteristics, and context of planned use. Designers and engineers conduct this analysis to “refine customer objectives and requirements, define initial performance objectives and refine them into requirements, identify and define constraints that limit solutions, and define functional and performance requirements based on customer provided measures of effectiveness [42].” To achieve this goal, designers must ask the following questions:

- What are the reasons behind the system's development?
- What are the customer's(s') expectations?
- Who are the users and how do they intend to use the product?
- What do the users expect of the product?
- What is their level of expertise?
- With what environmental characteristics must the system comply?
- What are existing and planned interfaces?
- What functions will the system perform, expressed in customer language?

- What are the constraints (hardware, software, economic, procedural) to which the system must comply?
- What will be the final form of the product: such as model, prototype, or mass production?

The result is three views of the product. First, the operational view (operational architecture - Section 1.1) addresses how operators will use the system, how well it will behave, and under what conditions. The operational view provides the systems engineers with the information required to develop the functions required of an independent system. The functional view (functional architecture - Section 1.1) identifies what the system must do to achieve the desired operational behavior. The tasks identified in the functional view provide the engineers with discrete tasks. The tasks can be paired with the subsystems or components and their interactions required to complete a task. Finally, the physical view (physical architecture - Section 1.1) describes how the product is constructed to achieve the desired functions. At this point in the process, the product architecture is selected based on the requirements and customer needs. Though the systems engineering process is an iterative, the engineers can only test a few options. The choices tend to be based on intuition or the most demanding requirements.

The analysis consists of fifteen tasks to achieve the desired result [42]:

1. *Customer Expectations*: Define and quantify customer expectations, including operational requirement documents, mission needs, technology-base opportunity, direct communications with customer, or requirements from the higher system level
2. *Project and Enterprise Constraints*: Identify and define constraints which include:

- *Project specific:* Approve specifications and baselines previously developed, costs, updated technical and project plans, team assignments and structure, and control mechanisms
 - *Enterprise:* Identify and define management decisions, general enterprise specifications, standards or guidelines, policies and procedures, domain technologies, and physical/financial/human resource allocations
3. *External Constraints:* Identify and define external constraints, including public and international laws and regulations, technology base, compliance requirements, threat system capabilities, and interfacing systems
 4. *Operational Scenarios:* Identify and define operational scenarios, including interactions with the environment and other systems and physical inter-connectivities with interfacing systems, platforms, or products
 5. *Measures of Effectiveness and Suitability (MOE/MOS):* Identify and define system effectiveness measures that reflect customer expectations and satisfaction
 6. *System Boundaries:* Define which elements are under and outside of design control
 7. *Interfaces:* Define the functional and physical interfaces to external or higher-level and interacting systems, platforms, and products in quantitative terms
 8. *Utilization Environments:* Define the environments for each operational scenario, including weather conditions, temperature ranges, topologies, biological, time, and induced
 9. *Life Cycle Process Concepts:* Analyze outputs of tasks 1-8 to define key life cycle process requirements in developing, producing, testing, distributing, operating, supporting, training, and disposing of the system

10. *Functional Requirements*: Define what the system must accomplish or can do
11. *Performance Requirements*: Define levels of performance for each higher-level function
12. *Modes of Operation*: Define key modes of operation, including environmental conditions, configuration, and operational
13. *Technical Performance Measures (TPMs)*: Identify key indicators of system performance which tend to be technical thresholds and goals that must be met or production risk increases
14. *Physical Characteristics*: Identify and define physical characteristics, including color, texture, size, and weight
15. *Human Factors*: Identify and define human factors, including physical space limitations, ergonomics, and human interface

2.1.3 Functional Analysis and Allocation

The functional analysis and allocation phase decomposes the tasks or missions outlined in the requirement analysis to lower level functions. The decomposition takes apart the mission and arranges parts into logical sequences. One of the key facilitators to the analysis is a functional flow block diagram.

It allocates performance and other limiting requirements to the functional levels, further defining the tasks and clarifying intangible capabilities into measurable metrics.

Next, the phase identifies and defines all internal and external functional interfaces by conducting sensitivity studies on the arrangement and groupings of functions. The rearrangement minimizes control interfaces and reduces the complexity of completing a task.

Also, the phase allocates functions to components in the system, creating the system or product architecture. By assigning functions to components, the designers create relations among the component determining how they will interact with each other. In the current era, software has an ever more important role in the functionality of the software. During the functional analysis, the systems engineers assign the software’s responsibilities to functions. This process determines the cost and development time of the software which ends up becoming a driving cost of product development [21].

Finally, the phase examines the life cycle functions (development, production, operations and support, and disposal) for the product.

The functional architectural layer assists designers in this phase. It “is a top-down decomposition of system functional and performance requirements [42].”

2.1.3.1 Function Flow Block Diagram

A function flow block diagram (FFBD) shows the relationship of functions or tasks that must be achieved by a system. There is a direction to these charts, showing what must occur to achieve a task. However, these diagrams do not provide the time required by or complexity of a function element.

The diagrams break down a process to organize task information. The diagrams help answer the question “what” needs to happen not “how” the task must be performed, providing the designers an open problem where unlimited options are permitted as long as the designs achieve the required task. Figure 15 displays an example FFBD. Each element can be broken down with its own (lower level) FFBD. The lower levels provide a greater description of the task, and as the designer adds levels, the design becomes increasingly constrained [76]. The example given in Figure 15 is a UAV conducting a basic suppression of enemy air defenses (SEAD) mission.

The mission includes eleven stages. First, the aircraft must takeoff by catapult.

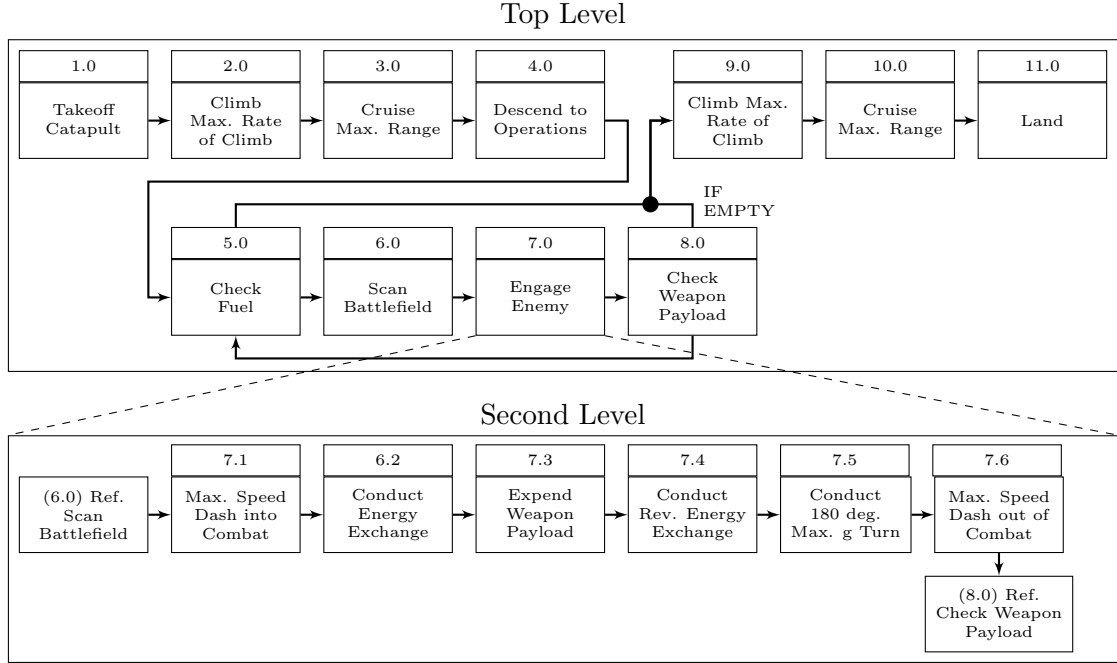


Figure 15: Example Function Flow Block Diagram of UAV SEAD Mission

Second, it must climb at the maximum rate of climb to the cruise altitude. There, it will cruise at speed and altitude for best range. Next, it will descend to a preset, operational altitude. Now the task becomes an iterative loop where it checks fuel and weapons stores while it scans for enemy targets. When it detects an enemy position, it will engage the enemy. At the time the aircraft runs out of arms or reaches a fuel limit, the aircraft climbs back to cruise altitude, where it cruises at speed and altitude for best range. Finally, the aircraft lands.

Each of these stages can be broken down into more detail. Figure 15 looks at the engage enemy stage and breaks it down to the second level. The engage enemy stage consists of six sub-stages. First, the aircraft must conduct a maximum speed dash into the battle space. Once the UAV is above the target, it drops its payload. During this time, it must be capable of conducting two energy exchanges to avoid enemy fire. It then makes a 180° turn and exits the battle space with a maximum speed dash.

This mission can be broken down even further so that designers can assign components to each task and a physical architecture/configuration begins to emerge.

2.1.3.2 *Functional/Physical Matrix*

The functional/physical matrix is a facilitator that allows designers to relate the functional architectural layer to the physical architecture or configuration. The matrix allows for consistency and traceability between the functional analysis and allocation and the design synthesis stages of systems engineering. The matrix breaks down the task/mission required of a system into segments and relates them to components required to complete the segment. The matrix displays the functional architectural layer and the configuration as a table and tree diagram respectively. Figure 16 displays an example simplified functional/physical matrix of a UAV.

In Figure 16, the functional/physical matrix breaks up a simplified suppression of enemy air defenses mission and applies it to a reduced version of a UAV's configuration. The matrix decomposes the mission into the preflight check, load, warm-up and taxi, take-off, cruise, scan for enemies, and engage enemies segments. Furthermore, the UAV must be able to communicate, provide surveillance, and conduct combat. This example ignores some of the segments since segments such as climb and cruise would be repetitive. The configuration in this example is composed of the airframe, engine, communications, sensors, and weapon systems.

The matrix draws connections between mission segments and components. During the preflight check segment, operators must check all of the components. While in the load segment, operators must load the weapon systems onto the aircraft. In the warm-up and taxi segment, the aircraft rolls along the ground via landing gear and is powered by the engine as the subsystems turn on and the engine heats up to the operating temperature. In this example, the landing gear is assumed to be a part of the air frame. In a more detailed example, designers can decompose the configuration further. During the take-off and cruise segments, the airframe must provide lift, the engine provides the power and propulsion, and the sensors assist in navigation of the vehicle. In the scan for enemies segment, the communications system allow for

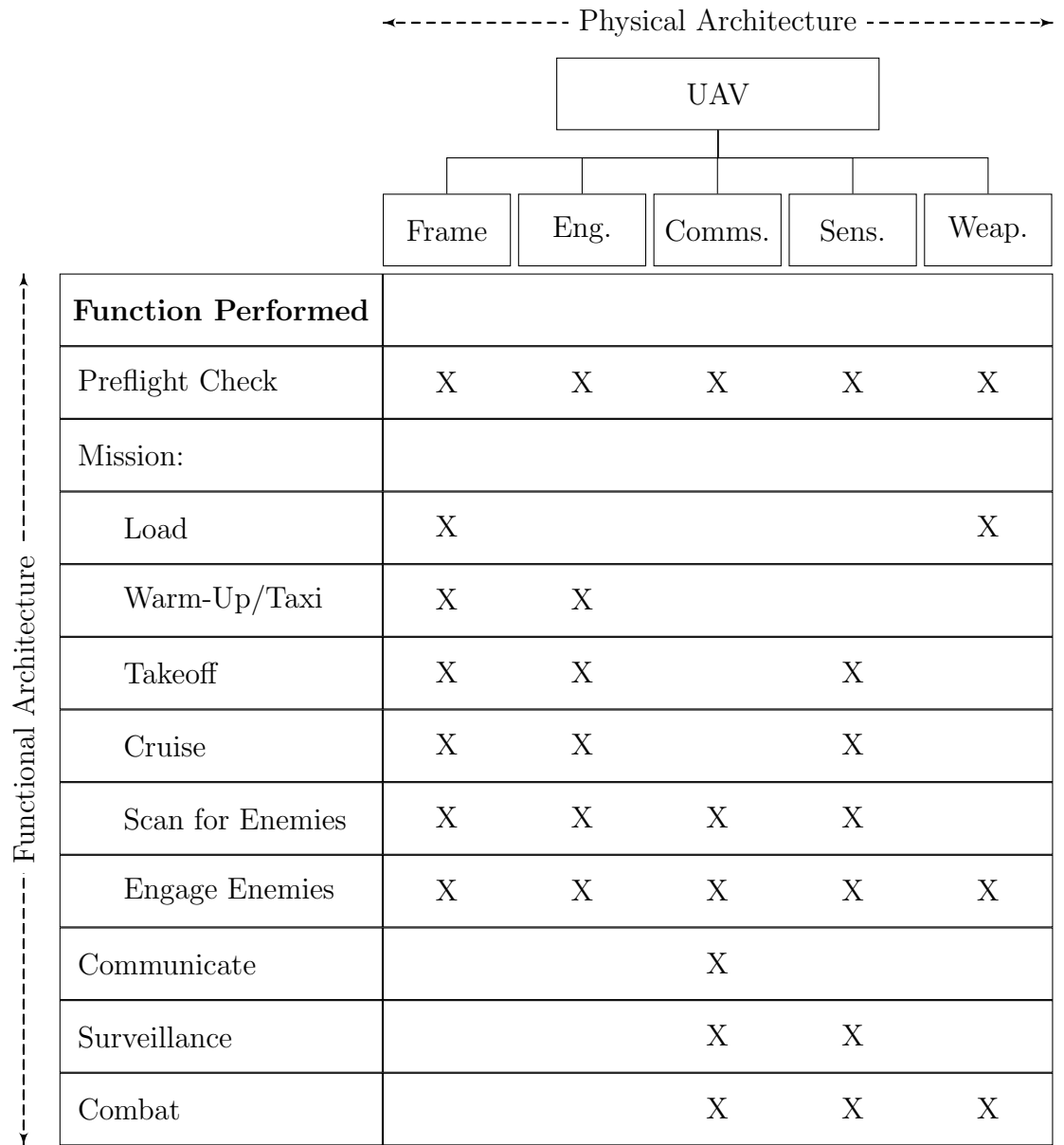


Figure 16: Example Simplified Functional/Physical Matrix of a UAV

communications among friendly entities. While in the engage enemies segment, the UAV can deploy the weapons system to destroy any enemies identified.

There are three sub-functions the UAV must be capable of conducting. The UAV must be able to communicate, scan for enemies, and engage enemies. All three sub-functions use the communications system, the scan for and engage enemies segments require the sensors, and the engage enemies function uses the weapons system.

2.1.4 Design Synthesis

The design synthesis phase develops tangible products with dimensions and characteristics. Furthermore, it ensures the architecture can achieve the functions set and determines whether the performance and functional requirements are feasible. This phase achieves the following actions:

- Transform the architectural view (Functional to Physical)
- Define alternative system concepts, configuration items, and system elements
- Select preferred product and process solutions
- Define or refine physical interfaces (Internal or External Interfaces)

This phase takes advantage of numerous sizing and synthesis tools which incorporate multiple disciplines to determine the design and performance of a product. These tools include “Computer-Aided Design (CAD), Computer-Aided-Systems Engineering (CASE), and the Computer-Aided-Engineering (CAE) can help organize, coordinate and document the design effort [42].” In aerospace engineering, simplified modeling tools allow designers to size aircraft based on specific missions.

The final result is a conceptual product with outlined dimensions, design variables, and characteristics. Also, it provides a full picture of the product line and the product architecture. Two tools that allow the systems engineers understand the physical or

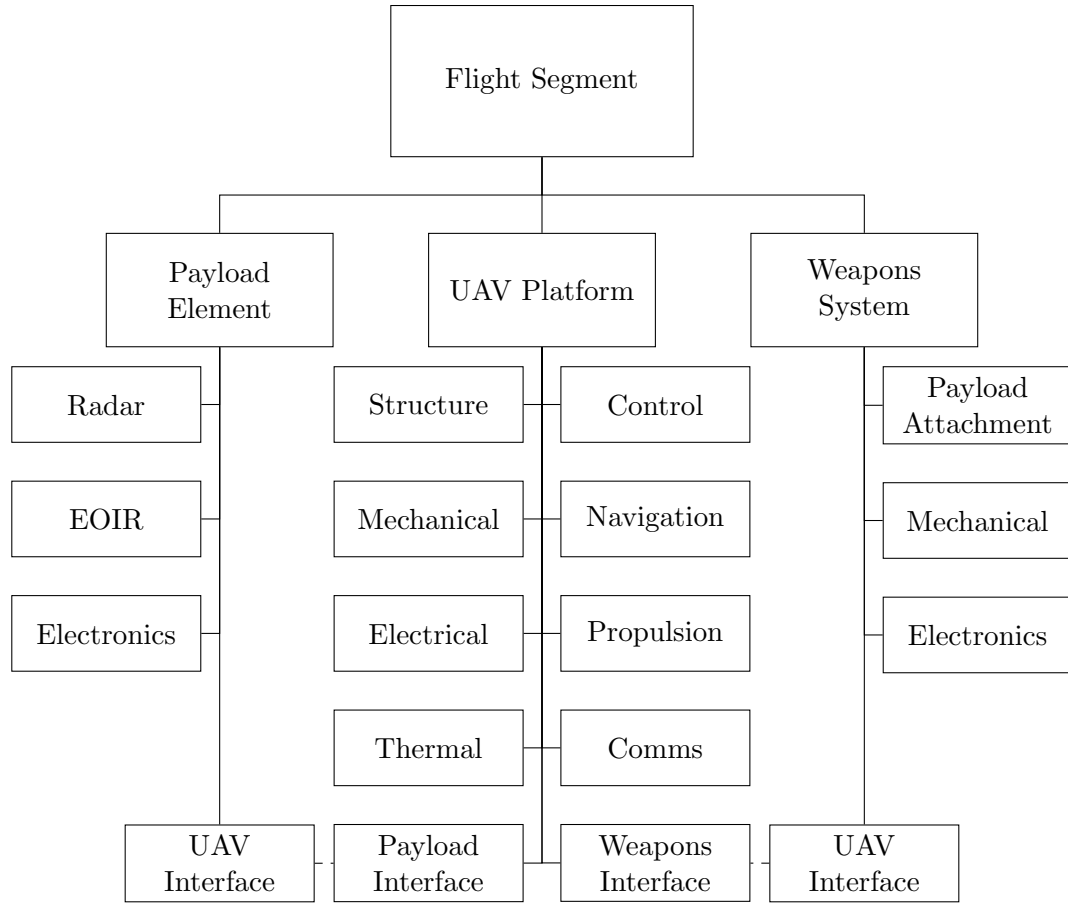


Figure 17: Example UAV Product Breakdown Structure

system architecture are the product breakdown structure and the N² Diagram. Also, systems engineers use simplified modeling in the early phases of design to understand how the proposed system performs while completing its assigned tasks.

2.1.4.1 Product Breakdown Structure

The product breakdown structure is a hierarchical tree diagram of the components, subsystems, hardware, software, and any other element of the design. The hierarchy of the design displays the projects managerial organization. It also shows what parts interact with another, as well as a vague idea how the design will look when finished [76]. Figure 17 displays an example product breakdown structure of a UAV.

In Figure 17, the UAV is broken down into three elements which are the payload, weapons system, and the main UAV platform. During each mission or flight segment,

each component and discipline have various roles and responsibilities it must attend. The payload element consists of the Radar and EOIR sensors. The main platform consists of the structures, mechanical, electrical, thermal, control, navigation, propulsion, and communications systems. The weapons system consists of the weapons payload and its associated electrical and mechanical systems. The payload and weapons systems each have interfaces with the main UAV platform and the main platform with the payload and weapons systems. The product breakdown structure provides a means to organize responsibilities for each of these components during a mission segment, allowing the designers to ensure the physical architecture can achieve the goals set in the functional architecture.

2.1.4.2 N² Diagram or Design Structure Matrices

N² diagrams are commonly used to show interdependencies between functions of subsystems, components, or mission segments [113]. It also can display the feed forward or feedback of information between functions. The decomposition of the systems provides the designer with an understanding of each subsystem, component or mission requirement's impact on the design. N² diagrams can be displayed as organizational charts that show how information passes from one element to another or as mathematical matrices composed of the strength of the relation. Figure 18 displays an example N² diagram of a basic tube-body-wing UAV. The elements (components/requirements) appear on the diagonals, and the arrows represent the flow of information from one element to another.

In the example found in Figure 18, there are nine components, the wing, empennage, fuselage, engine, computer/processor, Radar, EOIR sensor, navigation (INS/GPS) subsystem, and communications subsystem.

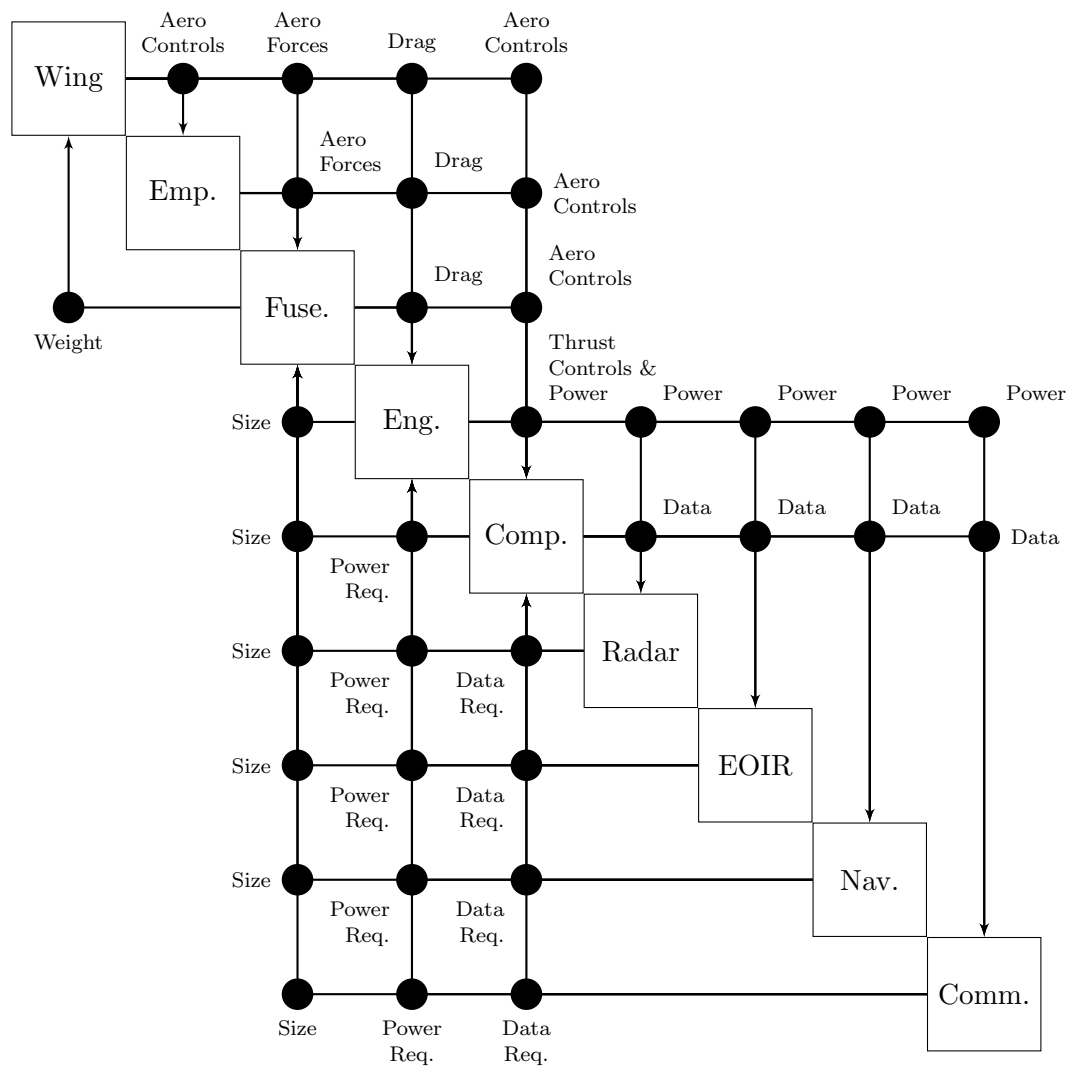


Figure 18: Example N² Diagram of a Basic UAV

The wing is responsible for providing lift for the aircraft. Therefore, by experiencing various loads and environmental conditions, the wing transfers various aerodynamic forces to the fuselage, drag to the engine, and control characteristics of the empennage and computer/processor.

Similar to the wing, the empennage experiences various aerodynamic loads to control the aircraft. These forces influence the structural design of the fuselage, thrust required of the engine, and control characteristic of the computer/processor.

The fuselage is the main platform of the UAV. It is the main structural element of the aircraft which houses the subsystems, engine, and bulk of the fuel. Therefore, it provides drag to the engine, aerodynamic controls to the computer/processor, and weight to the wing.

The engine provides energy and power to the electronics and subsystems. Furthermore, it provides thrust characteristics to the computer/processor, which influences the control of the aircraft during operations.

The computer/processor acts as the brain of the system. It provides data characteristics to the electronic subsystems (Radar, EOIR sensor, navigational unit, and communications unit). Furthermore, it provides the power required to the engine and size required to the fuselage.

The electronic subsystems are the Radar, EOIR sensor, navigational unit, and communications unit. Combined with the computer/processor, they make up the central nervous system of the aircraft. The mission requirements drive their characteristics, including weight, size, and power required. The weight and size drive the design of the fuselage, and the power required drive the design of the engine.

2.1.4.3 *Simplified Modeling*

Simplified modeling is used to understand how a system performs during a mission. Often, simplified modeling consists of simulating the mission or functions of the proposed design [113]. Designers primarily use these models to assist in making major decisions during the conceptual phase of design. The models predict the system's performance quickly, taking advantage of past test vehicles and experiments, allowing the designers to collect significant amounts of data. The models include mass fraction (otherwise known as mass-balancing), power-balancing, force calculation, heat/cooling, energy consumption, structural, and manufacturing process analyses [121]. Furthermore, there are systems engineering models that help designers organize and structure the decisions made during the design process. These are the primarily used for aerospace systems, but other models exist for other industries.

Fidelity of the models can be changed and usually relates inversely to computational time. Designers primarily use lower fidelity models in the conceptual design phase and Higher fidelity models in preliminary and detailed design phases where refined decisions occur.

Mass-fraction analysis incorporates energy-consumption and force-balancing. It is used to calculate how much fuel or energy is necessary to complete a mission. The analysis calculates the force or power required for a mission segment which relates to the amount of consumed energy during the mission segment [121].

Power balancing analysis provides constraints for the engine and vehicle sizes for each mission segment, with each mission segment having a model associated with it. The models include the required power to overcome the drag and weight of the aircraft. The engine power or thrust, wing size, and weight must not violate any of the power constraints [121].

Force balancing models in aerospace include aerodynamic, thrust, and weight forces. During each segment, the system must be able to provide enough lift and

thrust to overcome the drag and weight [121].

Heating and cooling models predict the subsystem requirements to maintain nominal temperatures in the electronics. Many models exist including ones that use excess fuel, convective cooling from bypassed external air, and refrigerated subsystems. Depending on the subsystem selected appropriate thermal models are needed [49].

Structural models help size the elements that make up the structure of the aircraft. The models require computer-aided-design (CAD) models to define the dimensions of the structural elements and can create finite element models (FEM) which are used to simulate and size the items based on loading characteristics. Some examples include AutoCAD, ProE, Nastran and Patran, and Hypersizer.

Manufacturing models calculate the cost of developing and producing a system as well as determine whether or not the design is feasible to manufacture. These include manufacturer process scheduling, structural fabrication, and discrete time modeling. These models require the layout of the system, manufacturing processes and technologies, and the number produced. Without these models, it is impossible for the producer to predict the economic feasibility of the design.

Finally, designers use systems engineering models to help organize and trace decisions made during the design process. These models are called computer-aided engineering (CAE) and systems engineering (CASE) models. These standardized software packages provide documentation, three-dimensional drawings, and track technical requirements of the product throughout the design process [42].

All of the analyses provide the models necessary to evaluate a design. Combined with other techniques, the formulated methods facilitate the designer in architecture selection.

2.1.5 System Analysis and Control

Systems engineers conduct system analysis and control throughout the process to ensure the design is feasible, viable, and below a certain risk threshold. Engineers and product management conduct the following studies to ensure these product characteristics:

- Trade-Off Studies
- Risk Management
- Configuration Management
- Technical Reviews and Audits
- Modeling and Simulation
- Metrics in Management

Trade-off studies change design variables within a possible range to determine their impact on performance, operational, and production requirements or metrics [42]. They tend to incorporate many disciplines and engineering teams to analyze the incidence of each design variable. Furthermore, by changing variables, designers create alternative designs and challenge all predetermined assumptions. The studies consist of six steps: establish the problem, review inputs, set up method, identify and select alternatives, measure performance, and analyze results. Thus, the studies allow designers to make evidence and data-based decisions about the product.

Risk exists in all fields and activities in the real world. In systems engineering, risk becomes even more apparent. Engineers can start with factors they know will change or are unknown (known-unknowns). However, complexity adds emergent uncertainty otherwise called unknown-unknowns. Risk management attempts to characterize all possible uncertainties and incorporate them into the design of the product. In the

systems engineering environment, there are four types of risk: internal processes, external influences, prime mission products, and supporting products [42]. First, the risk from internal processes deal with the managing, engineering, and producing of the product. Second, the risk from external influences arises from regulations, upper-management, and value-chain (product raw goods or resources) dynamics. Third, the risk from the prime mission products relates to the uncertainty that the product will meet its performance and cost requirements. Fourth, the risk of supporting products relate to the availability of resources the product in question is dependent on during operations. Management uses a four-step process planning for possible outcomes, assessing and identifying risk when it occurs, handling risk to mitigate problems, and monitoring the results of attempting to manage the risk ensuring the steps taken address the problem.

Configuration management allocates tasks and production to various teams and external entities along the value-chain to ensure the development of each component and subsystem. A manager has the options to either internally develop, taper, or outsource production of configuration elements [125]. By building elements internally, management has complete control over the processes, but at a significant fixed cost. Tapering development allows some minor tasks of the value chain to be outsourced usually reducing cost but at a loss of control. Finally, management can outsource the production of a configuration element drastically reducing cost but completely losing control of the production. Furthermore, barriers between firms make design changes extremely costly. In the end, the main producer must ensure all the configuration elements integrate in a way that achieves all of the desired operations and functional requirements.

Technical reviews and audits are key assessments of the development of the product. Usually, these reviews occur during all three stages of the systems engineering process. They include the:

- Alternative System Review
- System Requirements Review
- System Functional Review
- Preliminary Design Review (includes System Software Specification Review)
- Critical Design Review
- Test Readiness Review
- Production Readiness Review
- Functional Configuration Audit
- System Verification Review
- Physical Configuration Audit

In the end, the reviews “assess the maturity of the design/development effort; clarify design requirements; challenge the design and related processes; check proposed design configuration against technical requirements, customer needs, and system requirements; evaluate the system configuration at different stages; provide a forum for communication, coordination, and integration across all disciplines; establish a common configuration baseline from which to proceed to the next level of design; and record design decision rationale in the decision database [42].”

Modeling and simulation is a mathematical, logical, or even physics-based process of determining how the product will perform various operations or tasks determined during the functional analysis [42]. There are three classes of simulations: virtual, constructive, and live. Virtual simulations involve humans operating the product in a simulated environment. Constructive simulations take advantage of computer-aided models to generate the product and describe its physical features, taking advantage

of tools such as CAD, CAE, CAM, CASE, and life cycle costing. Live simulations test prototypes in as-close-as-possible to real operational situations.

Metrics can provide engineers and product managers with useful indicators of the product development's progress. They measure the cost and time it has taken to design and test the system and track the success of the management's product development strategy. Furthermore, they track the effectiveness, suitability, and performance of the product, ensuring the product can achieve the desired operations and functional requirements.

2.1.6 Process Output

The process's output includes the system architecture, technical specifications, baseline designs, acquisition baseline, and decision database.

The system architecture involves the composition of components, otherwise known as the configuration, and the relations among the components. The components and their relations detail how the system functions together to achieve all the tasks. It allows the manufacturing entity to organize and plan further development of the design.

The specifications detail the items, materials, or services required and the procedures to determine whether the design will achieve the desired capabilities. Specifications help provide “accurate estimates of necessary work and resources, act as a negotiation and reference document for engineering changes, provide documentation of configuration, and allow for consistent communication among those responsible for the eight primary functions of system engineering [42].” Finally, they act as guides during the verification process. Furthermore, there are various levels of specifications which define corresponding baseline designs, as shown in Table 1.

Baselines document the product at the different levels of the systems engineering process [42]. There are three baselines: functional, allocated, and product. The

Table 1: System Engineering Specifications and Baselines [42]

Specification	Content	Baselines
System Specs.	Defines mission/technical performance requirements and allocates requirements to functional areas and defines interfaces	Functional
Item Performance Specs.	Defines performance characteristics of CIs and CSCIs and details design requirements and with drawings and other documents form the Allocated Baseline	Allocated “Design To”
Item Detail Specs.	Defines form, fit, function, performance, and test requirements for acceptance (Item, process, and material specs start the Product Baseline effort, but the final audited baseline includes all the items in the TDP)	Product “Build To” or “As Built”
Process Specs.	Defines process performed during fabrication	
Material Specs.	Defines production of raw materials or semi-fabricated material used in fabrication	

functional baseline consists of the “top-level” functions, requirements, and interfaces that engineers derive from the operational requirements or tasks desired by the customer or set by the designers. The allocated baseline defines the lower-level subsystem and component design criteria, interface definitions, drawings, and processes. The product baseline defines the subsystems and components with physical characteristics completely defining the product, allowing for production to start. Furthermore, there is an acquisition program baseline which assesses the products maturity and economic viability.

Finally, the last output of the systems engineering process is the decision database. The database is a record of decisions made during the process, outlining the logic and justification for each decision made during this process. The database can include trade studies, simulations, quality function deployment (QFD) analysis, and analysis of alternatives (AoA).

2.1.7 Requirements Loop

The requirements loop provides feedback to the initially set functions and capabilities the system must be able to achieve. This process results “in a better understanding of the requirements and should prompt reconsideration of the requirements analysis [42].” The loop ensures all functions identified trace back to a specific requirement outlined in the requirements analysis.

2.1.8 Design Loop

The design loop provides feedback to the initially set system architecture and functions outlined in the functional analysis and allocation phase. It maps the functional architectural layer to the physical architecture. Furthermore, it provides feedback to determine whether the architecture can meet the functional and performance metrics set in the earlier phases. The loop also ensures the system can achieve all desired functions and evaluates how well the system performs each of these tasks.

One method that helps systems engineers translate the functional view to the physical view is the functional-physical matrix. It allows the engineers to visibly trace the functional tasks to physical elements required to complete the tasks. Furthermore, if there are multiple options available in a discrete space, a trade tree can provide sensitivity studies showing the trades of changing the functional or physical architectures.

2.1.8.1 Trade Trees

Trade trees are a way to organize discrete elements of the architectures where trade-off or sensitivities studies can occur. Figure 19 displays an example trade tree diagram. A trade tree breaks down all the possible combinations of discrete decisions. Each path represents the series of decisions leading to a particular architecture, creating a tree-like structure. For sensitivity studies, the designer can implement the same approach except each branch represents a variation in one of the input variables, not a decision [56, 19].

In Figure 19, five categories explain the possible options of the UAV's role. The categories include range/endurance, speed, size, configuration, and launch/recovery. Table 2 lists the options for each below:

As the designer chooses an option from each category, the number of total options decreases due to compatibility issues, leading the engineer down a path of decisions, each with its consequences. Each branch of the tree diagram represents a decision along with a path. The result is a clear role for the UAV to be designed (shown in the matrix of the figure).

2.1.9 Verification

Verification is a process that ensures the resulting configuration from the design synthesis stage achieves the customer needs and functional requirements set during the requirements analysis stage. During verification, designers and systems engineers

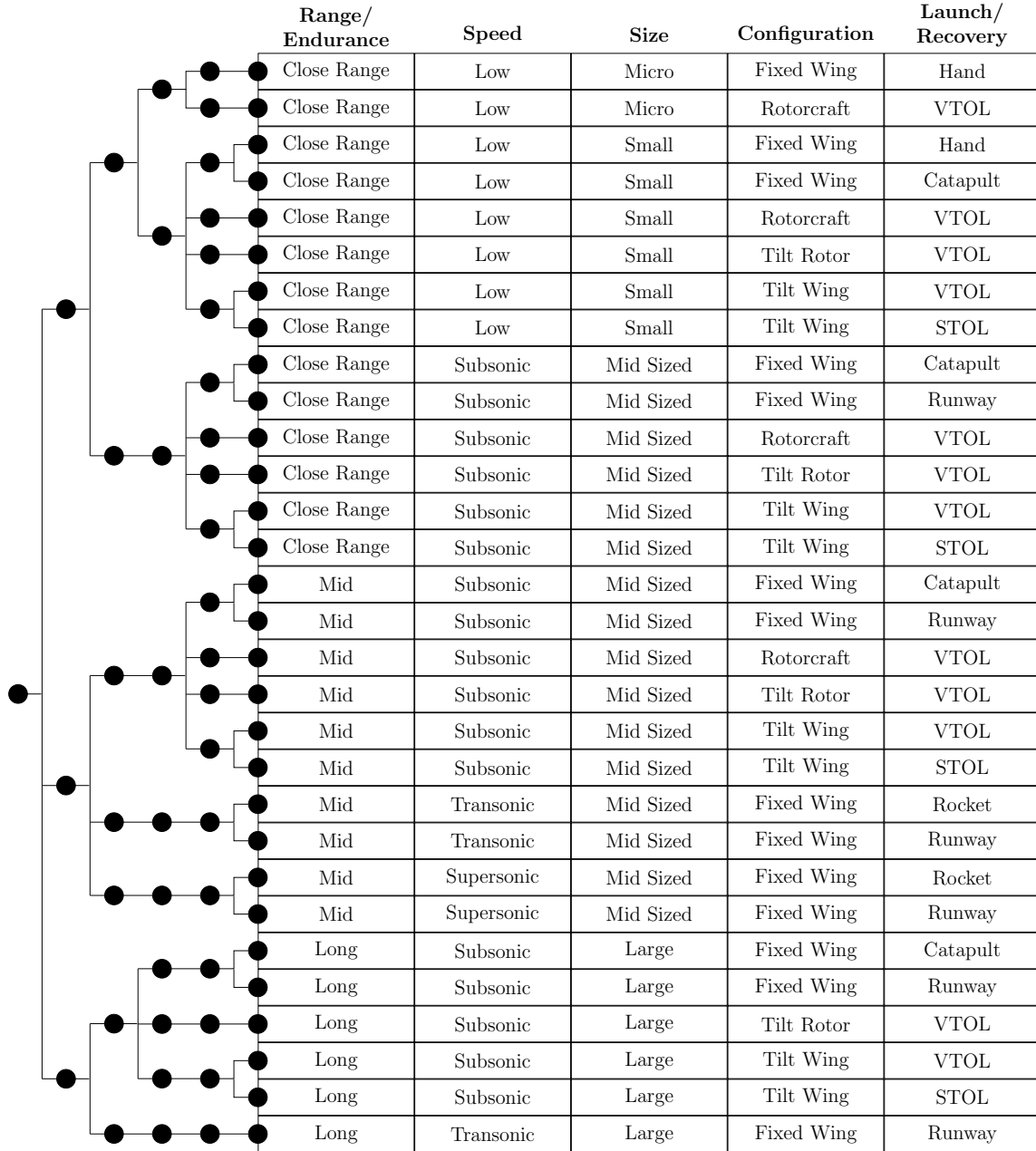


Figure 19: Example UAV Trade Tree

Table 2: Example UAV Trade Tree Options

Range/ Endurance	Speed	Size	Configuration	Launch/ Recovery
1. Close Range	1. Low	1. Micro	1. Fixed Wing	1. Hand
2. Mid	2. Subsonic	2. Small	2. Rotorcraft	2. Catapult
3. Long	3. Transonic	3. Mid Sized	3. Tilt Rotor	3. Runway
	4. Supersonic	4. Large	4. Tilt Wing	4. Vertical Takeoff and Landing (VTOL)
				5. Short Takeoff and Landing (STOL)
				6. Rocket Boost

must conduct the following activities [42, 66]:

- Analysis
- Inspection
- Demonstration
- Test

The analysis uses mathematical and analytical tools to determine whether the selected system achieves the functional requirements based on data from system, component, and subsystem-level analysis and testing. The analysis can assist in determining if the proposed system is feasible or viable.

Inspection is “the visual examination of the system, component, or subsystem [42].” This process ensures the physical features or the system meets the specifications set by the customer or manufacturer.

Demonstration examines the system during staged operations, demonstrating that the proposed system can achieve the desired capabilities in real-life situations.

Testing is the process that uses the system in real-life operations to obtain detailed data to verify the performance or provide sufficient information to verify performance characteristics.

2.1.10 Systems Engineering Process Summary

The current systems engineering process provides frameworks and methods that allow the product architecture to be selected logically, but primarily provide trade-offs between types of components and design variables rather than sensitivity of the relations among the components. For all intents and purposes, the process requires the engineers to come to a conclusion early on about which product architecture to implement, perhaps conduct a trade-off between a few competing product architectures. As a result, engineers have come up with new methods that analyze and provide methods or frameworks to assist in selecting the most favorable system or product architecture.

2.2 Existing Product Architecture Selection Methods

In academia and various industries, engineers have formulated methods to help facilitate product architecture selection. As discussed in the problem definition, a framework should provide the system architect/engineer with the following means to properly implement a product architecture. First, it must form requirements from customer demands and business goals of the parent company. Then, the framework must define the products functional requirements from customer demands. Furthermore, the framework must provide the system architect with an understanding of how and which the functional requirements drive the product architecture. These drivers must include ones that are constantly changing and evolving. Then, the framework must be able to efficiently generate alternative product architectures, by implementing a way to characterize and explore the product architecture space. From the wealth of choices, the framework must provide the architect with insights on what types of

product architectures manufacturers and customers favor. From the background research conducted in this chapter, flexibility and complexity can determine the product architecture's favorability. In design or robust design, there are trade-offs between performance and cost or optimality and consistency. However, when choosing product architectures, a compromise between requirement satisfaction, flexibility, and complexity exists. The product architecture must be able to achieve the customer demands, be resilient to changing needs, and reduce a manufacturer's cost by limiting the number of design changes during production. Both resilience to changing requirements and reduction of design changes relate to the product architecture's flexibility and complexity, respectively. Finally, to decide on which product architecture to implement, the method must provide the architect with insights on what types of product architectures are favorable. Therefore, the framework must provide a way to derive what qualifies a "good" product architecture.

After reviewing the existing methods, an architecture selection method must include the following:

Essential Elements of the new Framework:

1. Relation to the overall manufacturer's business strategy
2. Identification of major product architecture selection drivers
3. Examination of stochastic and time dependent product architecture drivers
4. A numerical way to relate product architectures
5. Determination of a product architecture's ability to achieve customer needs, functional requirements, and fiscal concerns
6. Determination of the product architecture's impact on the interactions and couplings between various disciplines and components
7. Identification of favorable product architectures

2.2.1 Quality Function Deployment (QFD) or Relational-Oriented Systems Engineering and Technology Trade-off Analysis (ROSETTA)

2.2.1.1 Qualify Function Deployment

Quality Function Deployment (QFD) is the combination of management and systems engineering tools [122]. The method achieves two objectives:

1. "To convert the users' needs (or customer demands) for product benefits into substitute quality characteristics at the design stage."
2. "To deploy the substitute quality characteristics identified at the design stage to the production activities thereby establishing the necessary control points and check points *prior* to production start-up."

QFD elicits knowledge from system matter experts to identify key drivers, trades, risk, and desirability. Alternative designs can be compared and again are weighed based on expert knowledge. The group of experts tends to consist of individuals from various disciplines to reduce bias. Often an expert's bias will drive the design chosen towards his or her "pet" project. However, with healthy group dynamics, the team can choose an appropriate configuration.

The process uses the chart found in Figure 20. The chart can be broken down into various spaces:

- A. Customer Demands
- B. Weighting of each Customer Demand
- C. Relation between Customer Demands and Functional Requirements
- D. Goals of each Functional Requirement
- E. Functional Requirements
- F. Alternatives Space
- G. Relation among Customer Demands
- H. Relation among Functional Requirements
- I. Risk or Importance of each Functional Requirement

The 'F' space can be modified from configurations to product architectures for this problem, but the designer can only analyze a limited number due to the time required to evaluate each architecture.

Overall, QFD visually and efficiently displays trade-offs, drivers, risk, and desirability. It also allows the designer to trace decisions made throughout the design

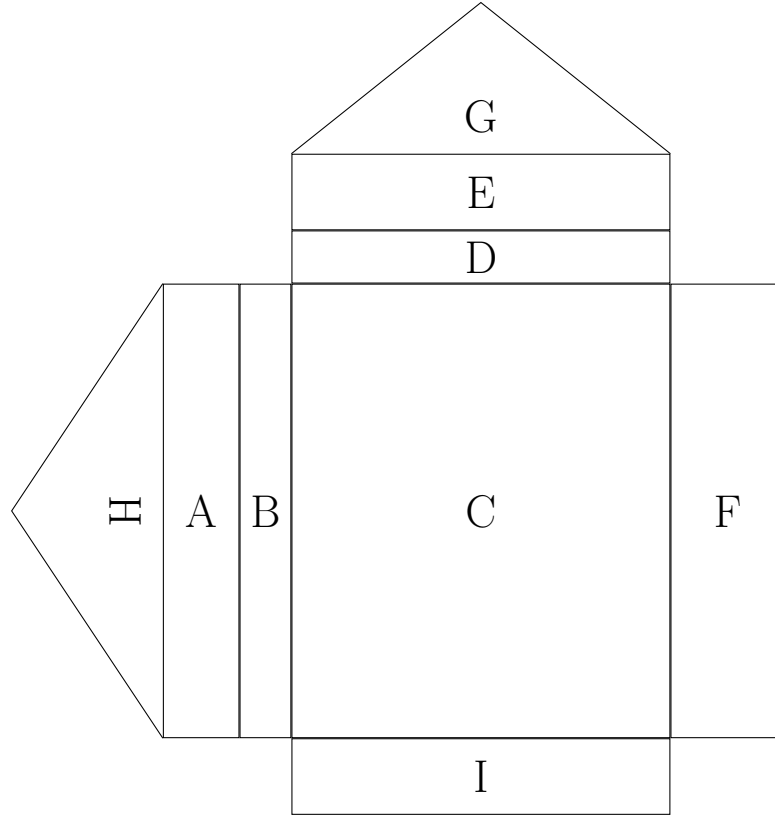


Figure 20: Quality Function Deployment [122]

process. However, it relies on expert opinions, analyzes a limited number of architectures, and does not account for interactions and couplings. These limitations may cause problems later on in the design process.

Table 3 shows where the QFD provides the information required to make an appropriate decision on what levels of reconfigurability and commonality. The QFD provides a great way to connect the customer needs with functional requirements to provide the systems engineers with a quality information on how to evaluate a product architecture. Furthermore, it can identify requirements that drive the design and selection process by providing correlations and weightings of importance between customer needs and functional requirements. However, it does not provide sufficient information on the time-dependency of these requirements, numerical relations among alternative product architectures, and analysis of the interactions and couplings the

Table 3: Evaluation of the QFD Method

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
QFD	✓	✓	✗	✗	✓	○	○

product architecture impacts. Though it provides key benefits it does not meet all the requirements desired to make a decision on the product architecture to implement.

2.2.1.2 Relational-Oriented Systems Engineering and Technology Trade-off Analysis

The Relational-Oriented Systems Engineering and Technology Trade-off Analysis (ROSETTA) shifts the QFD from expert to physics and numerical-based decision making, [91]. It employs modeling and simulation to create data that shows trade-offs and relations. The paradigm shift removes expert opinion and any bias that might occur from those types of decisions. However, some of the same problems exist as the QFD. It can analyze more, but still a limited number of architectures and can miss some interactions and couplings during the analysis.

Table 4 shows where the ROSETTA framework provides the information required to make an appropriate decision on what levels of reconfigurability and commonality. ROSETTA provides a great way to connect the customer needs with functional requirements to provide the systems engineers with a quality information on how to evaluate a product architecture. Furthermore, it can identify requirements that drive the design and selection process by providing correlations and weightings of importance between customer needs and functional requirements. It is a superior version of a QFD since it uses historical or simulation data to provide the correlations or weightings. However, it does not provide sufficient information on the time-dependency of these requirements, numerical relations among alternative product architectures, and

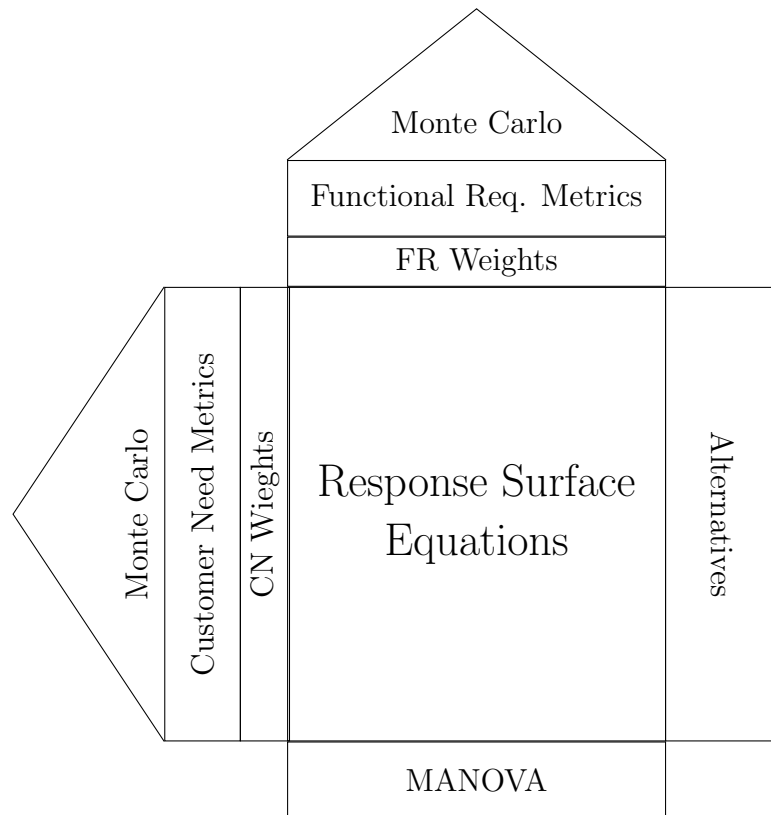


Figure 21: Relational-Oriented Systems Engineering and Technology Trade-off Analysis (ROSETTA) [91]

Table 4: Evaluation of the ROSETTA Framework

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
ROSETTA	✓	✓	⊘	⊘	✓	○	○

analysis of the interactions and couplings the product architecture impacts. Though it provides key benefits it does not meet all the requirements desired to make a decision on the product architecture to implement.

2.2.2 Unified Trade-off Environment

The Unified Trade-off Environment (UTE) is a method whose objective is to “capture the simultaneous impact of requirements, technologies, and design variables [90]” by running physics-based sizing and synthesis tools with various mission requirements, concepts (design variables), and technologies. Usually, by integrating all of the considerations, dimensionality becomes a problem. Therefore the method assumes the impact of all three categories can be superimposed. After collecting the data, surrogates provide the means to run time-efficient Monte Carlo simulations allowing probabilistic-based design and sensitivity studies of desirables (outputs) and constraints.

The UTE can facilitate in probability-based design problems and allows designers conduct sensitivity studies among design variables, requirements, and technologies. However, it does not capture the interactions among the requirements, design variables, and technologies, since it considers each category separately. Implementing this method in product architecture selection would involve creating data sets and models for each architecture considered.

Table 5 shows where the UTE method provides the information required to make an appropriate decision on what levels of reconfigurability and commonality. The

Table 5: Evaluation of UTE Method

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
UTE	○	✓	✓	⊘	✓	✓	⊘

UTE method starts with the functional requirements, assuming the systems engineers already derived the functional requirements from the customer needs. Also, it provides systems engineers with a way to decouple requirements, design variables, and other considerations in the design process. This feature allows the systems engineers to identify product architecture drivers and their sensitivity to time. However, the UTE method is often used looking at one product architecture analyzing a specific grouping of inputs and design variables. Though, it can measure the vehicles ability to satisfy the functional requirements and the interactions and couplings between components it does not provide a way to relate or compare alternative product architectures easily. Though it provides key benefits it does not meet all the requirements desired to make a decision on the product architecture to implement.

2.2.3 Customer Demands

Defining the product architecture based on the customer demands is a method based on logical reasoning, not purely numerical analysis [155]. One major assumption for this approach is that manufacturing costs do not impact the choice of architecture. Two sub-assumptions support the latter: First, fixed architectures are more cost effective than modular architectures. Second, modular architectures are more cost effective than multiple fixed platforms.

The method starts by interviewing customers from various target markets using questionnaires, interviews, study groups, or conjoint analyses to identify customer

needs or requirements. Then, during the second round of questionnaires, the designers conduct interviews, study groups, or conjoint analyses to obtain importance of each need or requirement. After collecting the results, the designers or management can identify the critical needs or requirements of the product. The third round of surveying provides target values for the critical requirements. From the results of the third round, the designers or management can divide the customers into various target markets, sorted by time, market, or total population, providing means and variances of requirements. The statistical values allow the design team to select the product architecture.

A flow diagram, displayed in Figure 22, provides the logical process the designer should follow. The method identifies four architecture options: platform generations (modular), fixed portfolio (fixed), platform family (product family), or adjustable portfolio (reconfigurable) architectures. Yu does not identify what type of product family designs to use for a platform family architectures. A platform family corresponds to a scale-based product family since Yu identifies platform generations as modular designs. Finally, an adjustable portfolio or a reconfigurable architecture is easily upgradeable, allowing for the platform to reflect the significant standard deviation of customer demands. The adjustable portfolio can be modular or online reconfigurable designs based on what the designer believes is best.

Yu's customer demand method has some benefits drawbacks. Yu's understanding of the problem provides a consistent benchmark of what influences the architecture. It also permits stochastic and time varying requirements. However, Yu only defines four possible outcomes of the method, and a product can contain characteristics of multiple architectures. Also, he fails to identify what makes a population mean constant with time and what makes a standard deviation large. This lack of clarity leaves the problem open for the designer to make heuristic decisions.

Table 6 shows where the Customer Demands method provides the information

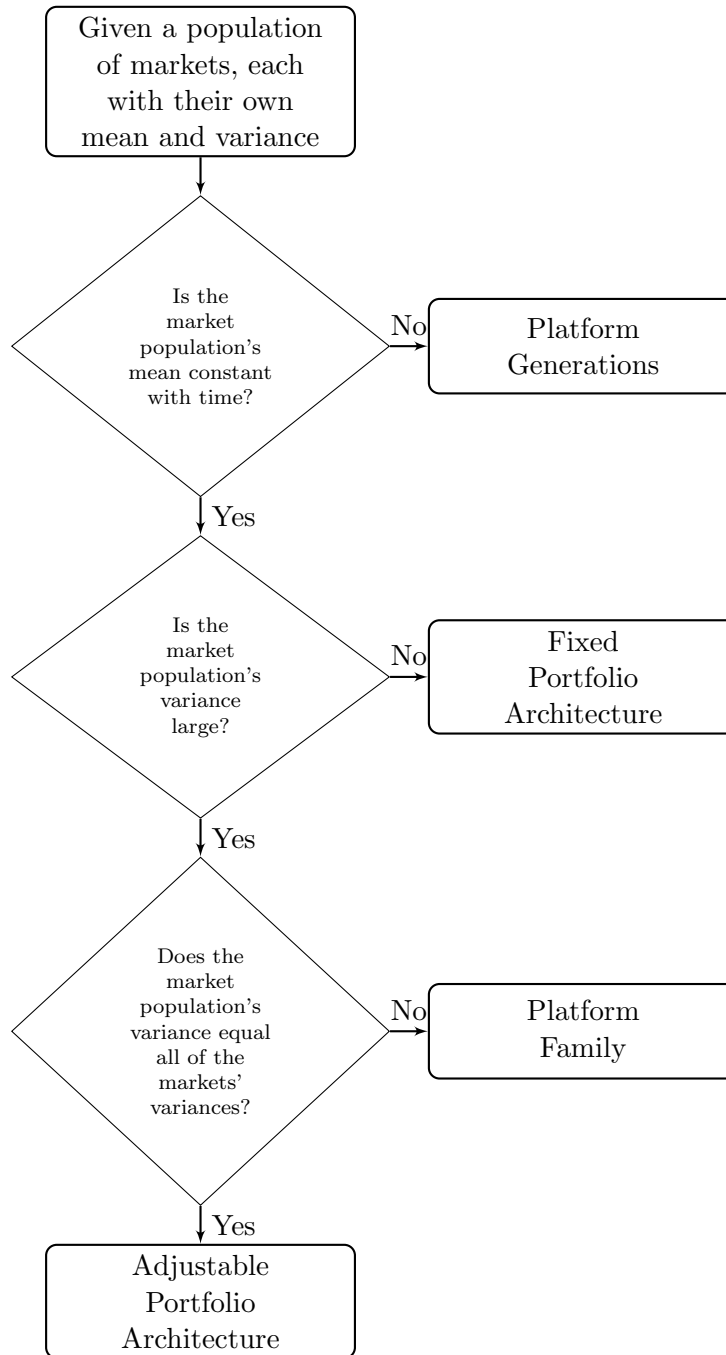


Figure 22: Customer Demand Method[155]

Table 6: Evaluation of the Customer Demands Method

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
Customer Demands	✓	✓	○	⊘	⊘	⊘	✓

required to make an appropriate decision on what levels of reconfigurability and commonality. The Customer Demands method requires surveying the possible customers to determine the needs and requirements of the new product. Then, it outlines a logical process of determining what type of product architecture to implement. However, it does not provide a relation between or a way to compare or evaluate the alternative product architectures. Though, the method is simple to understand it provides vague outlines on how to determine which product architecture is best. Also, it can handle stochastic requirements but does not determine how the distributions change over time and how this will impact the product architecture.

2.2.4 Robust Concept Exploration Method

The Robust Concept Exploration Method applies a parametric, robust design method [32]. The method instructs the designer to go through the following steps:

1. Classify all design parameters
2. Determine the design parameters to evaluate
3. Set the ranges and distribution of the parameters
4. Conduct a sensitivity analysis (an ANOVA analysis) on the parameters, identifying the terms with the greatest impact on output metrics
5. Reduce the Number of Parameters using results from the sensitivity study

6. Create response surface equations of various architectures to reduce run time of the analysis
7. Run a Monte Carlo
8. Select architecture evaluated using statistical process control metrics (C_{dk} , C_{dl} , and C_{du}). These metrics require some target and upper and lower bounds to identify the preference of each metric.

This method can handle stochastic requirements, fully explores the design domain, and ensures robust or flexible, modifiable top-level specifications. However, comparisons among architectures are difficult due to the construction of codes corresponding to each, slowing down the process and making it hard for the architect to explore the entire space. Therefore, the method requires some architecture down-selection beforehand.

Table 7 shows where the Robust Concept Exploration Method provides the information required to make an appropriate decision on what levels of reconfigurability and commonality. The Robust Concept Exploration Method starts with distributions of functional requirements, making the assumption the systems engineers already derived the functional requirements distributions from the customer needs. The ability of the method to handle the stochastic requirements allows system engineers to identify drivers and make some inferences on the time-dependency of the product architecture. However, the RCEM can only analyze one product architecture at a time. This feature reduces the systems engineers' ability to relate and compare the alternative product architectures easily. Though it provides key benefits it does not meet all the requirements desired to make a decision on the product architecture to implement.

Table 7: Evaluation of the Robust Concept Exploration Method

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
RCEM	○	✓	✓	⊘	✓	⊘	⊘

2.2.5 Variation-Based Platform Design Methodology

Raviraj Nayak, Wei Chen, and Timothy Simpson developed the Variation-Based Platform Design Methodology for product-family architecture optimization [103]. With this methodology, a designer can construct a product family while maximizing commonality, using stochastic performance requirements.

The approach selects a common product platform while trying to maximize standardization of components. First, “information that characterizes the needs and requirements for a product [are converted] into knowledge about a product [103],” setting means, ranges, and goals of desirables. Second, the designer selects a common platform by entering design variables’ averages and variances into the products analysis, then chooses a product with maximum commonality and satisfaction of all requirements. The process selects platform variables from the design’s variables that are similar based on mean and variance. Third, non-platform variables are optimized for various products in the family, concerning each mission or requirements.

This method is a straightforward and easy to implement. It also can handle stochastic requirements and design variables. Though it explores the entire product-family architecture space, the method is not applicable to other architectures, such as reconfigurable ones, and assumes the configuration has already been down selected.

Table 8 shows where the Variation-Based Platform Design Methodology provides

Table 8: Evaluation of the Variation-Based Platform Design Methodology

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
VBPCM	○	○	○	⊘	✓	⊘	⊘

the information required to make an appropriate decision on what levels of reconfigurability and commonality. The VBPDm starts with distributions of functional requirements, making the assumption the systems engineers already derived the functional requirements distributions from the customer needs. The ability of the method to handle the stochastic requirements allows system engineers infer which requirements drive the selection process and make some inferences on the time-dependency of the product architecture. However, the VBPDm can only analyze one product architecture at a time and assumes the product architecture is a product family with a common platform. This feature reduces the systems engineers' ability to relate and compare the alternative product architectures easily. Though it provides key benefits it does not meet all the requirements desired to make a decision on the product architecture to implement.

2.2.6 Multi-Disciplinary Analysis and Optimization (MDAO)

Multi-Disciplinary Analysis and Optimization (MDAO) decomposes a complex system into subsystems, disciplines, members, elements, components, sub-spaces, and sub-problems. It also analyzes the partitioned elements individually. The individual analyses provide design variables or constraints to the overall system, allowing the system to be optimized using serial or parallel computation. MDAO consists of two types of optimization methods: multidisciplinary feasible (MDF) and individual discipline feasible (IDF).

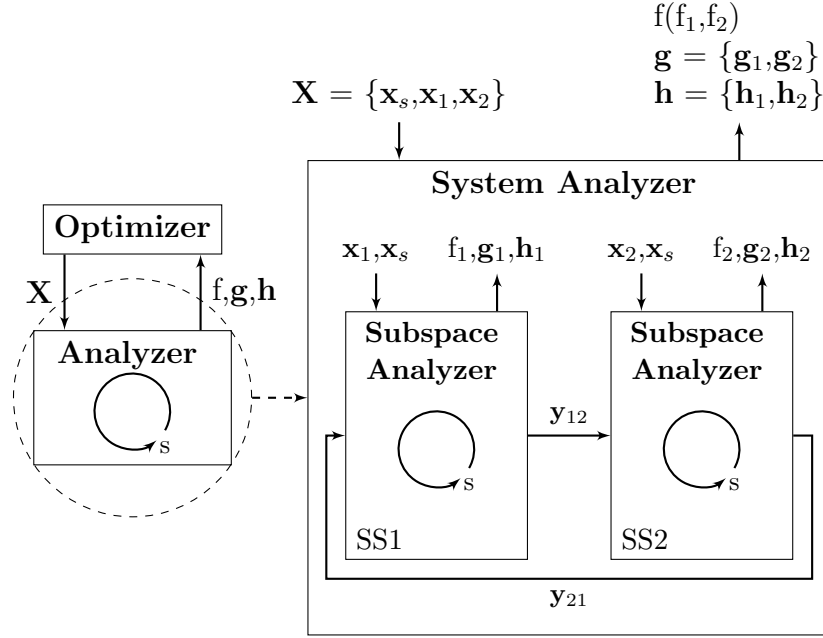


Figure 23: Multidisciplinary Feasible Structure [16]

2.2.6.1 Multi-Disciplinary Feasible

MDF is a serial method which uses an external optimizer that sends design variables to internal analyses that evaluate the partitioned elements' characteristics. There are three types of design variables: shared, local, and coupled design variables. More than one element use shared variables and only one element uses local variables. Coupled variables are outputs of one or multiple analyses and inputs for one or multiple analyses. An analysis updates associated coupled variables and send the coupled variables to the next analysis. This process repeats itself until the coupled variables converge, resulting in a converged system. The external optimizer then evaluates the design and constraints until it finds an optimal one. The external optimizer tends to use fixed point iteration due to its simplicity. This method is computationally inexpensive if there is low coupling. However, MDF can converge on sub-optimal designs due to divergence in fixed point iteration algorithms. Figure 23 displays a MDF structure.

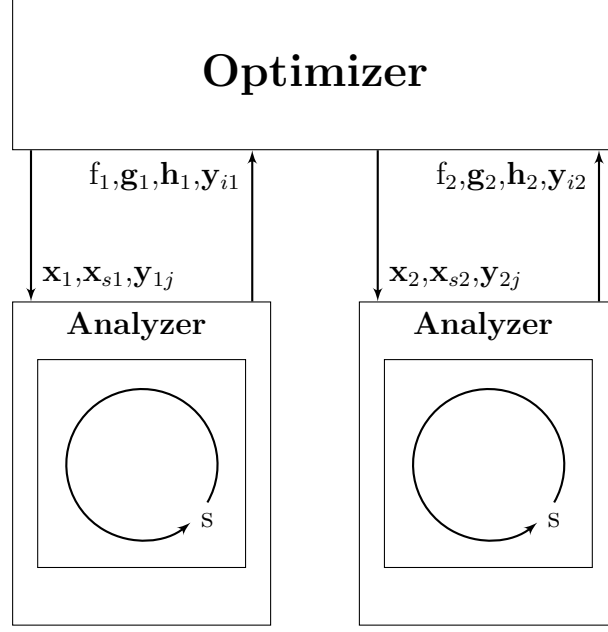


Figure 24: Individual Discipline Feasible Structure[16]

2.2.6.2 Individual Disciplinary Feasible

IDF is a parallel optimization which uses an external optimizer that sends design variables to each discipline/components analysis. The external optimizer defines all (shared, local, and coupled) design variables. The analyses calculate their constraints and provide output variables. Since the coupled variables before and after the analyses are not always consistent, equality constraints are added to the external optimizer to ensure consistent coupled variables. This method is computationally more expensive than MDF except when the coupling is high. Also, this method will always converge on optimal designs. Figure 24 displays an IDF structure.

Although MDAO techniques are commonly used throughout industry, because they provide a means to conduct data driven analysis, there are considerable gaps in the implementation. Depending on the system to be evaluated, one optimization might be better than the other. Also, since these techniques rely on optimization, stochastic input design variables can not be used without Monte Carlo simulations which are feasible but due to complicated optimization computation quickly becomes

Table 9: Evaluation of Multi-Disciplinary Analysis and Optimization (MDAO)

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
MDAO	○	✓	○	⊘	✓	✓	⊘

expensive. Furthermore, the designers must create specific simplified modeling must for each architecture analyzed. Therefore it becomes costly to analyze all of the architecture space, requiring some architecture down-selection beforehand.

2.2.6.3 Review of MDAO Methods

Table 9 shows where the Variation-Based Platform Design Methodology provides the information required to make an appropriate decision on what levels of reconfigurability and commonality. MDAO is a process that breaks down the design of a product into components or disciplines with analyses assigned to each element. The analyses determine the performance and other technical measures important in analyzing the design. The combination of analyses allow the process's operator to optimize the design or conduct analysis of the design space. It uses functional requirements derived from the customer needs as inputs, allowing systems engineers to determine which requirements drive the design of the product. However, the way MDAO is structured provides a limited amount of stochastic and time-dependent analysis. One of the key features of MDAO is its structure that decomposes the design problem into components and disciplines. This feature allows systems engineers to study the interactions and couplings between disciplines and components. However, it does not provides a way to numerically relate alternative product architectures. Studying a product architecture using MDAO requires switch modules on and off creating a discrete and categorical problem. Though it provides key benefits it does not meet all the requirements desired to make a decision on the product architecture to implement.

2.2.7 Contact and Channel Model

The contact channel model (C&CM) method connects geometry information with integration analysis to analyze architectures [15]. Then, using optimization techniques, a final architecture is selected. Elements are modeled and grouped into subsystems. C&CM analyzes the integration of subsystems by working surface pairs (WSP) and channel and support structures (CSS).

- “WSP are pairwise interfaces between two components or between a component and its environment. Working surfaces can be a solid surface of a body or a boundary, a surface of liquid, gas, or field which comes into permanent or occasional contact with the working surface (WS). They take part in the interchange of energy, material, or information within the technical system [15].”
- “CSS are a physical component a volume of liquid, gas, or space containing field which links exactly two WSP. They do not only participate in a transfer of energy, material, and information from one WSP to another, but they can also store them (e.g. the mass inertia) [15].”

The method can then be broken down into three steps - generating a C&CM dependency matrix (CDM), evaluating the CDM, and analysis of clusters in the CDM. Figure 25 displays the method.

The first step requires some preliminary work. The designer must determine requirements of the product architecture, define the importance of each requirement, select a principal architecture and create its corresponding FFBD, and combine the FFBD with its geometry information creating the C&CM. However, a complex system requires many CSS and WSP. “This makes the system representation relatively unorganized and not suitable to analyze the integration.” A CDM is a modified N^2 diagram consisting of weighted importance factors for all CSS interactions (I_{CSS}).

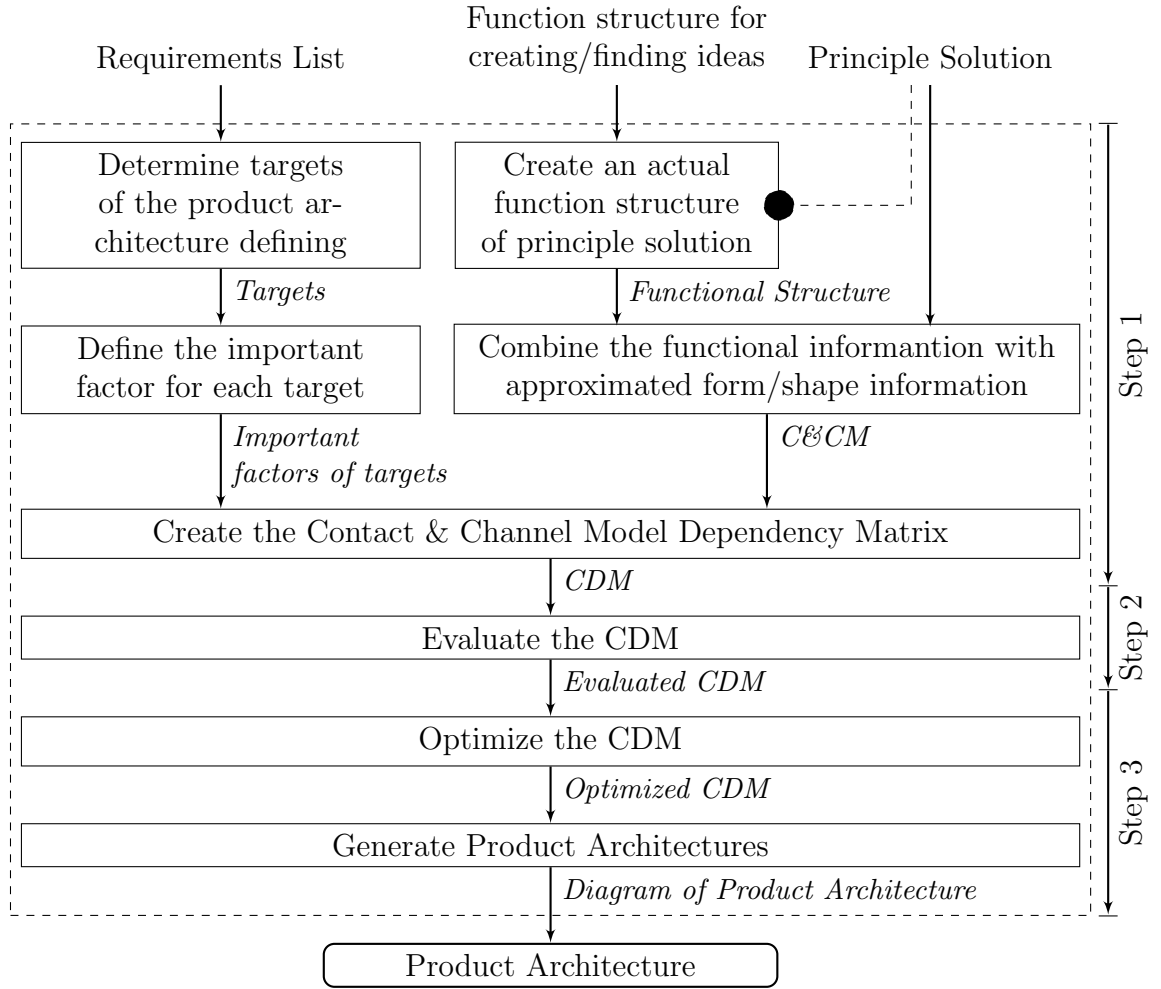


Figure 25: Contact Channel Model [15]

The CSS interactions are a combination of the WSP influence associated with the CSS's suitability. It has a minimum value of zero and a maximum of one. The CDM is then evaluated by breaking down the product into modules and evaluating how well the architecture can serve its purpose. Albers recommends using Equation 1.

$$Z = \frac{\left(\frac{\sum I_{CSS}}{\sum I_{CSS,max}} \right)_{insidemodule} - \left(\frac{\sum I_{CSS}}{\sum I_{CSS,max}} \right)_{outsidemodule}}{Numberofmodules} \quad (1)$$

Finally, since the goal is to maximize the architecture's performance, a genetic algorithm can be used to optimize the product architecture. This process finds the number of modules in the product and arrangement of the CSS.

The contact channel model utilizes an FFBD and an N² diagram. It even identifies how the architecture is set up, referring to energy, force, and information transferring. However, the process is heavily biased by heuristic determinations of WSP influence, CSS suitability, general product architecture geometry and is only applicable to modular designs. The method has potential to be data driven if simplified models were used to determine these parameters and geometry.

Table 10 shows where the Contact and Channel Model provides the information required to make an appropriate decision on what levels of reconfigurability and commonality. The Contact and Channel Model does a good job at decomposing the design problem into components and disciplines allowing for a great understanding of the internal dynamics of the product. The use of the CDM provides ways to evaluate and, in a way, relate alternative product architectures. However, it is limited in its ability to handle time-dependent requirements. It has many of the elements required to make an informed decision about the amounts of reconfigurability and commonality to implement, but it could probably be extended upon to improve its ability to handle extremely complex problems.

Table 10: Evaluation of the Contact and Channel Model

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
C&CM	○	✓	⊘	○	✓	✓	✓

2.2.8 Architectural Enumeration and Evaluation

The United Technologies Research Center (UTRC) developed the Architectural Enumeration and Evaluation (AEE) framework “to enable the efficient and traceable decision making that rapidly reduces the entire design space to regions that warrant further investigation [21, 156, 157].” AEE Decomposes the system into disciplines or components with corresponding functions, relations, and constraints. Then, the framework uses numeric analyses to evaluate the performance and feasibility of an architecture and configuration pairing. Pairings are generated using a design of experiments or in some cases full factorial of discrete component combinations. Ones that can not achieve required functions, derived from the customers’ demands, are considered infeasible and removed from further analysis. Next, the framework uses higher fidelity analyses to evaluate the performance and cost of each feasible pairing. If there are multiple evaluation criteria, an MADM technique can be used to rank the possible architecture and configuration pairings. Figure 26 displays the AEE two step process of enumeration and evaluation.

The AEE framework quickly filters and evaluates many architectures while providing performance, cost, and feasibility of each architecture numerically. However, it does not provide a way to analyze architectures resilience to requirement change or time dependency. Therefore it does not provide the means necessary to aid system architects in planning product development and strategic road mapping.

Table 11 shows where the Architectural Enumeration and Evaluation framework

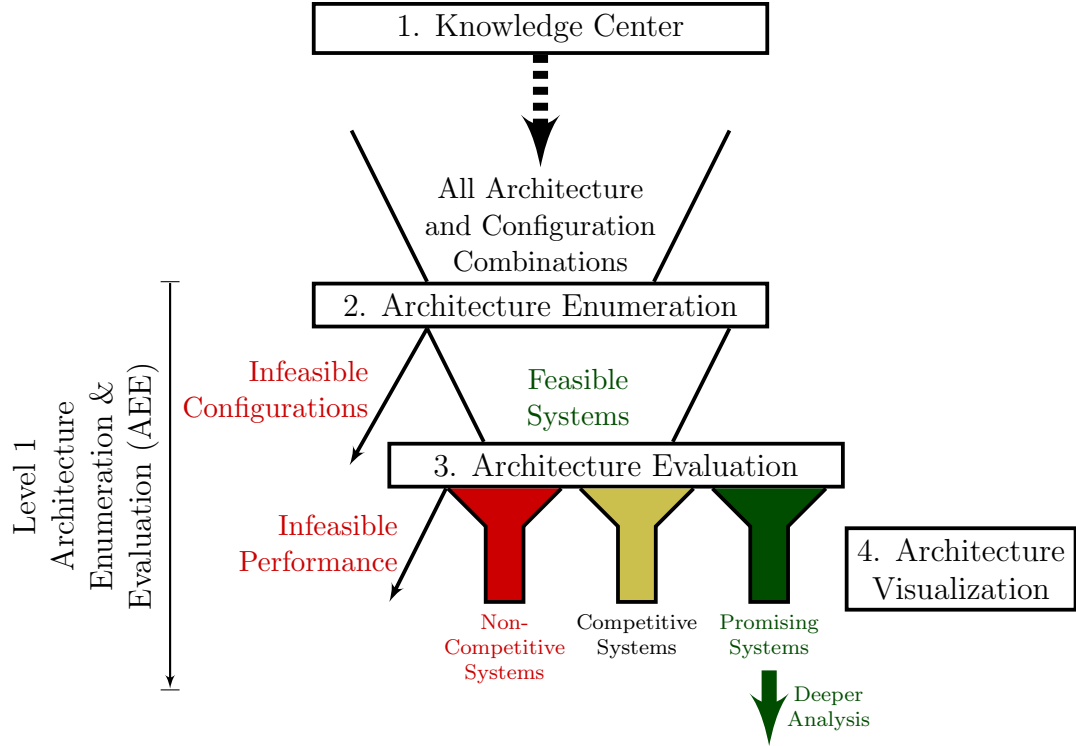


Figure 26: Architectural Enumeration and Evaluation [157]

provides the information required to make an appropriate decision on what levels of reconfigurability and commonality. The AEE framework can search all types of architectures. It uses the functional architecture to determine which physical architectures are feasible. The connection of the functional to physical analysis allows drivers to be easily determined. However, the framework primarily focuses on fixed requirements and does not provide the ability to analyze the internal dynamics of the design problem. Though, the framework has clear benefits in selecting the physical architecture of a system, it lack some of the requirements to determine the product architecture.

2.2.9 Architecture Selection under multiple Criteria and Evolving Needs for improved Decision-making (ASCEND)

The ASCEND framework is a generic top-down design decision support process that addresses the gaps in traditional design methods where engineers only analyze a few

Table 11: Evaluation of the Architectural Enumeration and Evaluation Framework

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
AEE	✓	✓	⊘	⊘	✓	⊘	✓

physical architectures [58]. It combines performance, life-cycle cost, and safety as a means of determining viable options. It combines traditional sizing and synthesis techniques with functional decomposition analysis to determine the feasibility and viability of a physical architecture.

The framework was demonstrated through a case study of a suborbital vehicle. These vehicles require high levels of safety and manageable costs. The highly constrained environment requires creativity to determine which physical architecture is the most favorable.

Figure 27 outlines the steps in the framework. First, the framework identifies metrics of interest and constraints relative to the design problem. The metrics are used in developing the overall evaluation criteria of the system. The constraints limit the design space, ensuring the system can complete the task or meet customer needs. Second, the framework breaks down the mission or capabilities required of the system. Then, conducts a compatibility study to determine which physical architectures can feasibly conduct the required tasks. The feasible physical architectures provide the inputs for the performance, life-cycle cost, and safety analysis. Third, the framework sends the alternative physical architectures to an optimizer which robustly optimizes and evaluates each alternative with respect to stochastic and time-dependent requirements. Finally, a TOPSIS method is used to determine the best alternative based on a combination of evaluation metrics.

Table 12 shows where the ASCEND framework provides the information required

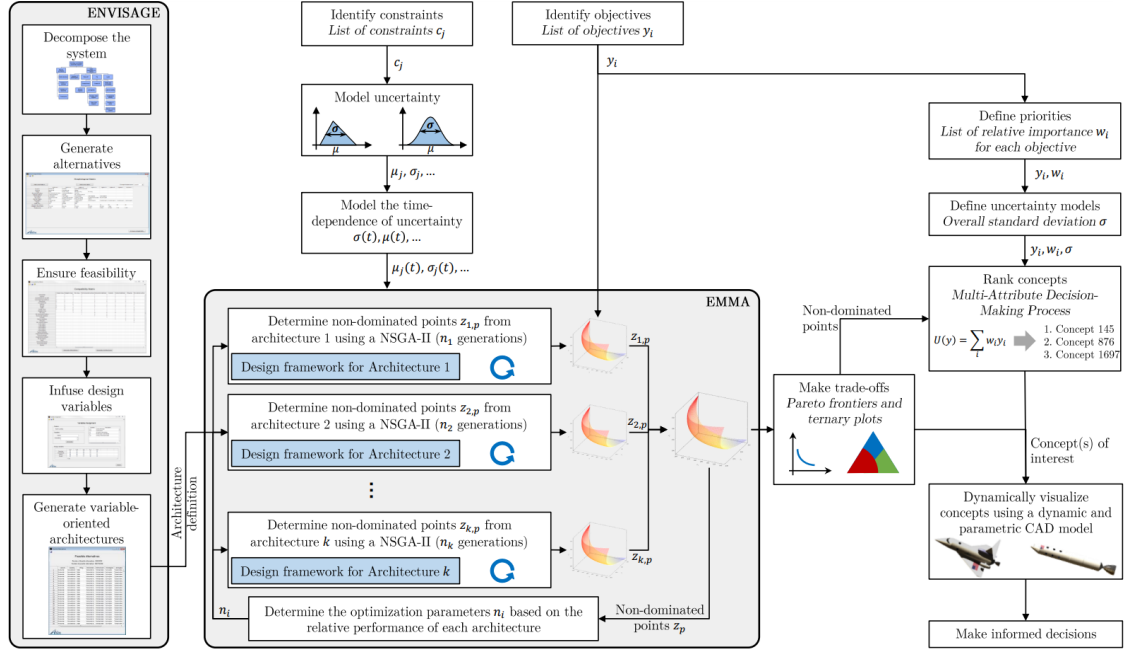


Figure 27: Architecture Selection under multiple Criteria and Evolving Needs for improved Decision-making (ASCEND) [58]

to make an appropriate decision on what levels of reconfigurability and commonality. The framework provides a clear connection between the functional and physical architectures of the system allowing for requirements that drive the physical architecture to be determined. Furthermore, it is built to handle time-dependent and stochastic requirements meaning the system is designed to robustly handle changing requirements. However, the framework does not present how the physical architecture might change over time. The framework links alternative physical architectures through the functional view but the problem is still presented as a categorical problem limiting the ability of systems engineers to quickly relate alternatives. The framework utilizes an MDAO method to optimize each alternative physical architecture. This implementation allows the system engineers to determine each alternative's ability to satisfy the customer needs, but it can only provide limited information about the internal dynamics of the problem. The ASCEND framework provides a valuable approach to selecting the physical architecture, but it was not designed to determine the levels of

Table 12: Evaluation of the ASCEND Framework

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
ASCEND	○	✓	✓	○	✓	⊘	○

commonality and reconfigurability of a product line.

2.2.10 A Product Family Design Methodology Employing Pattern Recognition

The Product Family Design Methodology Employing Pattern Recognition hopes to address the complex problem of product family design. Since product families share components, referred to as platforms, to “streamline design, improve manufacturing, and facilitate maintenance [59],” it is important the systems engineers correctly identify which components to make common and unique across the family. The methodology attempts to reduce the combinatorial problem by using cluster analysis to identify possible sets of commonality.

Figure 28 shows how systems engineers can break down the product family design space into higher level capabilities, functional requirements, products, and components. The methodology utilizes the fact that each level of the design space historically match with certain types of vehicles. Thus, clusters form in each level based on the capabilities required by each family member in the product line. The clusters form types of components or products. Using the clusters of historical data, engineers can determine options of the composition of components in each family member. The options provide the inputs to the design optimization step of the analysis.

After determining the composition of the physical architecture of each family member, the method places commonality constraints across the product line and optimizes the design of each alternative product architecture. The method can optimize based

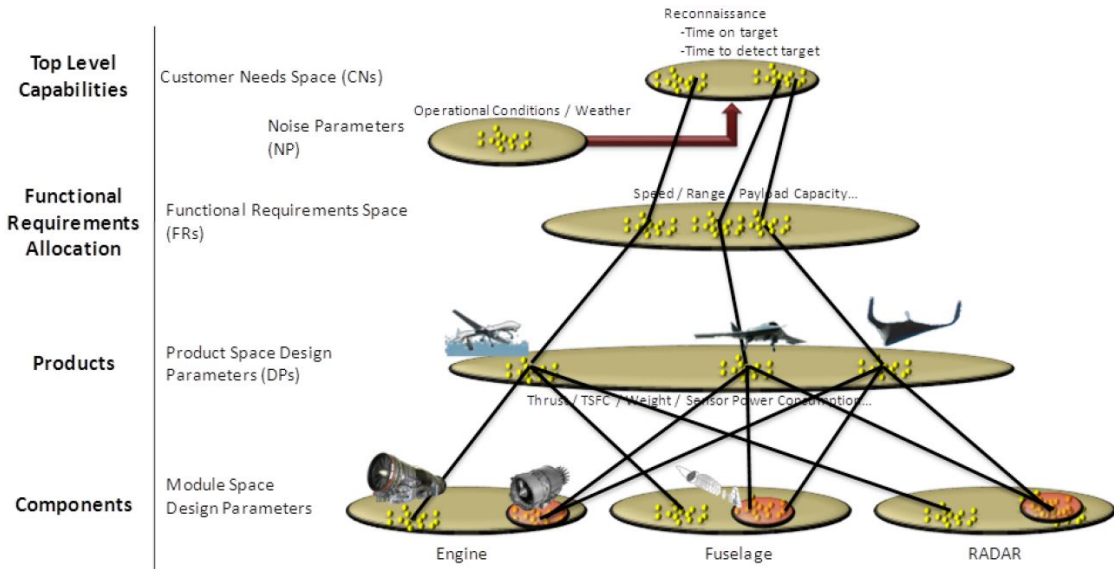


Figure 28: A Product Family Design Methodology Employing Pattern Recognition [59]

on absolute or robust performance and cost. The resulting performance and cost metrics can be traded or used in decision multi-attribute decision making methods to determine which product architecture to implement.

Table 13 shows where the Product Family Design Methodology Employing Pattern Recognition provides the information required to make an appropriate decision on what levels of reconfigurability and commonality. The method starts with the higher-level capabilities which the method assumes were already derived from the customer needs. The capabilities can be subjugated to noise parameter allowing for stochastic or time-dependent analysis. The functional requirements are discrete tasks summarized by variables to clarify the demand of each task. The combination of the capability and functional-based analyses allow the systems engineers to identify the drivers and their time-dependent impact on the product architecture. However, when the method starts to select the possible product architectures, it still describes the alternative product architectures combinatorially or categorically, lacking the ability to relate the alternatives quantitatively. Also, it only creates alternative product architectures with varying levels of commonality. It does not look at reconfigurability

Table 13: Evaluation of the Product Family Design Methodology Employing Pattern Recognition

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
PFDMPR	○	✓	✓	○	✓	⊘	○

at all. Finally, the method evaluates the alternative product architectures by analyzing their performance and cost. Analyzing the alternatives this way provides enough information to determine the product architecture’s ability to satisfy the functional requirements but does not analyze the internal dynamics of the design problem. Overall, the method provides many benefits in assisting the systems engineers select the configurations and commonality of the product line but fails add reconfigurability and analyze the risk of the design problem.

2.2.11 Evaluation of Existing Product Architecture Selection Methods

In general, to properly implement a product architecture, a framework should provide the system architect/engineer with a means to form requirements from customer demands and business goals of the parent company. Then, the framework must define the products functional requirements from customer demands. Furthermore, the framework must provide the system architect with an understanding of how and which the functional requirements drive the product architecture. These drivers must include ones that are constantly changing and evolving. Then, the framework must be able to efficiently generate alternative product architectures, by implementing a way to characterize and explore the product architecture space. From the wealth of choices, the framework must provide the architect with insights on what types of product architectures manufacturers and customers favor. From the background research conducted in this chapter, flexibility and complexity can determine the product

architecture's favorability. In design or robust design, there are trade-offs between performance and cost or optimality and consistency. However, when choosing product architectures, a compromise between requirement satisfaction, flexibility, and complexity exists. The product architecture must be able to achieve the customer demands, be resilient to changing needs, and reduce a manufacturer's cost by limiting the number of design changes during production. Both resilience to changing requirements and reduction of design changes relate to the product architecture's flexibility and complexity, respectively. Finally, to decide on which product architecture to implement, the method must provide the architect with insights on what types of product architectures are favorable. Therefore, the framework must provide a way to derive what qualifies a "good" product architecture.

All the methods reviewed can be evaluated by whether they meet this criterion. Figure 14 displays the evaluation. A green check mark means the method can already achieve this requirement, a circle means the method can be modified to meet this requirement, and a red circle with a slash through it means the method cannot achieve this requirement.

From the evaluation, it is apparent that no method achieves these requirements. Specifically, most of the methods or frameworks lack in the ability to relate alternative product architectures numerically. A number of past studies have developed indices to attempt to describe the levels of commonality and reconfigurability numerically. The review of these studies are found in the next section. Also, the identified methods do not sufficiently analyze the interactions that occur within the design problem. The interactions and couplings relate to the robustness, flexibility, and complexity of the design. Therefore, the next section introduces these terms as well.

The identified methods and frameworks inability to meet all of the attributes required to make an informed decision on what levels of commonality and reconfigurability to implement create a need for a new framework to be formulated. Therefore,

Table 14: Evaluation of Existing Architecture Selection Methods

	Relate Product Architec- ture to Business Strategy	Identify Product Architec- ture Drivers	Conduct Time- Dependent Driver Impact	Numerical Relations among Product Architectures	Evaluate Product Architec- tures	Analyze Design Interacti- ons & Coupling	Identify Product Architec- tures of Interest
QFD	✓	✓	⊘	⊘	✓	○	○
ROSETTA	✓	✓	⊘	⊘	✓	○	○
UTE	○	✓	✓	⊘	✓	✓	⊘
Customer Demands	✓	✓	○	⊘	⊘	⊘	✓
RCEM	○	✓	✓	⊘	✓	⊘	⊘
VBPCM	○	○	○	⊘	✓	⊘	⊘
MDAO	○	✓	○	⊘	✓	✓	⊘
C&CM	○	✓	⊘	○	✓	✓	✓
AEE	✓	✓	⊘	⊘	✓	⊘	✓
ASCEND	○	✓	✓	○	✓	⊘	○
PFDMPR	○	✓	○	○	✓	⊘	○

this dissertation formulates one which combines some of the elements of the past methods and introduces some new concepts, leading to the research objective of this dissertation, found in Section 2.4.

2.3 *Additionally Required Concepts*

While examining past architecture selections methods in Section 2.2, the examination brought forward a couple of concepts. These concepts include commonality and reconfigurability, complex systems, system flexibility and complexity, and software development. The product architecture is a combination of commonality and reconfigurability implemented across a product line. Therefore, concepts of how to represent commonality and reconfigurability must be explored. After understanding how to represent the product architecture, complex design analysis can be used to determine the relation between the product architecture and its capabilities. Therefore, complex design analysis must be understood. A few key concepts of complex

system design are the interaction between requirements and couplings between design variables. The interactions and couplings relate to the flexibility and complexity of the product. Therefore, flexibility and complexity must be defined in relation to product design and development. Finally, understanding the costs related to software development are especially important in modern product development. This section provides definitions and a detailed examination of these terms and concepts.

2.3.1 Past Commonality and Reconfigurability Studies

One of the key features in product architecture selection that the past methods struggle to address is the ability to relate alternative product architectures numerically. However, there have been studies in the product development field of research where indices have been created to present a numerical representation of the commonality and reconfigurability implemented in a product line.

Commonality and reconfigurability indices can represent a product architecture and serve as representations of the composition characteristics implemented in a product line. Since the indices will represent an inputted product architecture, they must be independent of the systems engineers' biases and qualitative decision making. Therefore, the previously developed indices must:

- Represent the composition of commonality or reconfigurability versus fixed or unique components
- Serve as inputs, independent of the design process analysis
- Show the relations and characteristics among the implemented components
- Represent commonality and reconfigurability across the product line
- Be independent of the systems engineers' qualitative decision making

These qualities will ensure the indices can serve as an independent input in the analysis of product architecture's impact on the design process. The review of previously

developed indices will provide possible options and traits that can be incorporated into the development of product architecture indices.

2.3.1.1 Commonality Studies and Indices

Researchers developed commonality indices to account for manufacturing processes, materials, assembly or fastening schemes, production volumes, and initial costs [143]. With varying respect to these considerations, many different commonality indices have been proposed.

Degree of Commonality Index (DCI) The first is the Degree of Commonality Index (DCI) [34]. The index is the most traditional form of reflecting “the number of common parent items per average distinct component [141].” Equation 2 shows the formulation of the DCI index, “where Φ_j is the number of immediate parents component j has over a set of end items or product structure level(s), d is the total number of distinct components in the set of end items or product structure level(s), and i is the total number of end items or the total number of highest level parent items for the product structure level(s) [141].”

$$DCI = \frac{\sum_{j=i+1}^{i+d} \Phi_j}{d} \quad (2)$$

The index can range from 1 to infinity, depending on the number of products and components. It represents the ratio of the number of common components to the total number of components in the product line. It is very easy to compute but it does not do well representing the ratio of common components when a new product is added to the line or redesigning the product line.

Table 15 shows how the Degree of Commonality Index meets the criteria required of a commonality index. The DCI, though a simple representation of commonality, almost meets all of the criteria required. Its range from zero to infinity limits its

Table 15: Evaluation of the Degree of Commonality Index

	Represent the Compo- sition of Components	Independent of the Design Process Analysis	Shows Cha- racteristics among Components	Represent Across the Product Line	Independent of Qual. Decision Making
DCI	⊗	✓	✓	○	✓

Table 16: Evaluation of the Total Constant Commonality Index

	Represent the Compo- sition of Components	Independent of the Design Process Analysis	Shows Cha- racteristics among Components	Represent Across the Product Line	Independent of Qual. Decision Making
TCCI	○	✓	✓	✓	✓

ability to measure the composition of commonality and represent commonality across the product line.

Total Constant Commonality Index (TCCI) Another index is the Total Constant Commonality Index (TCCI) [150]. The metric is an extension of the DCI, modifying it to range from zero to one. This makes up for the inability of the DCI to scale during redesign or adding products to the production line. Equation 3 shows the formulation of the TCCI index.

$$TCCI = 1 - \frac{d - 1}{\sum_{j=i+1}^{i+d} \Phi_j - 1} \quad (3)$$

Table 16 shows how the Total Constant Commonality Index meets the criteria required of a commonality index. The TCCI is an extended representation of the DCI. Therefore, it makes up for the DCI's inability to represent commonality across the product line. However, it still has difficulty representing the ratio of common versus unique components, especially if forms of reconfigurability are added to the product architecture.

Product Line Commonality Index (PCI) The first two indexes simply measure the ratio of common components across a production line. However, commonality can be a function of the size, materials, manufacturing, and assembly processes used in production. Therefore, the Product Line Commonality Index (PCI) was developed to account for these considerations [80]. Equation 4 shows the formulation of the PCI, “where P is the total number of non-differentiating components that can potentially be standardized across models, N is the number of products in the product family, n_i is the number of products in the product family that have component i , f_{1i} is the size and shape factor for component i , f_{2i} is the materials and manufacturing processes factor for component i , and f_{3i} is the assembly and fastening schemes factor for component i . f_{1i} is the ratio of the greatest number of models that share component i with identical size and shape to the greatest possible number of models that could have shared component i with identical size and shape (n_i). f_{2i} is the ratio of the greatest number of models that share component i with identical materials and manufacturing processes to the greatest possible number of models that could have shared component i with identical materials and manufacturing processes (n_i). f_{3i} is the ratio of the greatest number of models that share component i with identical assembly and fastening schemes to the greatest possible number of models that could have shared component i with identical assembly and fastening schemes (n_i) [141].”

$$PCI = 100 \times \frac{\sum_{i=1}^P n_i \times f_{1i} \times f_{2i} \times f_{3i} - \sum_{i=1}^P \frac{1}{n_i^2}}{P \times N - \sum_{i=1}^P \frac{1}{n_i^2}} \quad (4)$$

The PCI ensures a component is common if it has the same size, materials, manufacturing, and assembly processes used in production. Also, the index ranges from zero to one-hundred. Comparing the PCI and TCCI metrics allows the engineer to compare what processes are common in the development of the product. Also, it allows the engineers to get an understanding which components are scale-based. However, all of the three indices analyzed at this point determine the commonality

Table 17: Evaluation of the Product Line Commonality Index

	Represent the Compo- sition of Components	Independent of the Design Process Analysis	Shows Cha- racteristics among Components	Represent Across the Product Line	Independent of Qual. Decision Making
PCI	✓	○	✓	✓	✓

across a product line, not of an individual product, leading to the development of the next index.

Table 17 shows how the Product Line Commonality Index meets the criteria required of a commonality index. The PCI incorporates some of the manufacturing and material considerations that are relevant to determining commonality. However, many of these considerations can be the results of the design or production analyses. Therefore, the index becomes less of an input and more of an output of MDAO analysis.

Percent Commonality Index (%C) The Percent Commonality Index (%C) was developed to analyze the level of commonality within an individual product. It has three viewpoints: the component viewpoint, component-to-component connections viewpoint, and the assembly viewpoint [130]. These viewpoints can be combined through a weighting scheme to determine the overall %C of an individual product. Equations 5 through 8 show the formulation of the commonality viewpoints. The component viewpoint looks at the ratio of common components ($common_{comp.}$) to total components ($common_{comp.} + unique_{comp.}$). Equation 5 shows the formulation of the component viewpoint in the %C index.

$$C_c = \frac{100 \times common_{comp.}}{common_{comp.} + unique_{comp.}} \quad (5)$$

The component-to-component connection viewpoint looks at the interfaces present in the system architecture of a product. If two components share a connection with

the platform component, the interface is a common connection ($common_{conn.}$). The viewpoint looks at the ratio of these common connections to the total connections within the system architecture ($common_{conn.} + unique_{conn.}$). Equation 6 shows the formulation of the connection viewpoint in the %C index.

$$C_n = \frac{100 \times common_{conn.}}{common_{conn.} + unique_{conn.}} \quad (6)$$

The assembly viewpoint can be broken down into the assembly component loading and assembly workstation viewpoints. The assembly component loading viewpoint looks at the sequences while the discrete workstations in the production of the component. Both are the ratios of the common sequences ($common_{ACL}$) or workstations ($common_{AW}$) to the total sequences ($common_{ACL} + unique_{ACL}$) or workstations ($common_{AW} + unique_{AW}$). Equations 7 and 8 show the formulation of the assembly component loading and workstation viewpoints in the %C index respectively.

$$C_l = \frac{100 \times common_{ACL}}{common_{ACL} + unique_{ACL}} \quad (7)$$

$$C_a = \frac{100 \times common_{AW}}{common_{AW} + unique_{AW}} \quad (8)$$

Engineers can combine all of the viewpoints in the final %C index through a normalized weighting scheme (I_x). The weightings are determined at the discretion of the designers based on what they view are important in the product development process. Equation 9 show the formulation of the %C index using a weighted sum of the viewpoints previously described.

$$\%C = I_c \times C_c + I_n \times C_n + I_l \times C_l + I_a \times C_a \quad (9)$$

The index varies from zero to one-hundred. It is able to account for manufacturing and assembly considerations. However, it can only be applied to one product in the

Table 18: Evaluation of the Percent Commonality Index

	Represent the Compo- sition of Components	Independent of the Design Process Analysis	Shows Cha- racteristics among Components	Represent Across the Product Line	Independent of Qual. Decision Making
%C	✓	○	✓	⊘	⊘

product line and can't be scaled to account for the family as a whole.

Table 18 shows how the Percent Commonality Index meets the criteria required of a commonality index. The %C was specifically developed for individual system's commonality, not across a product line. Also, it incorporates some of the manufacturing and material considerations that are relevant to determining commonality. However, many of these considerations can be the results of the design or production analyses. Therefore, the index becomes less of an input and more of an output of MDAO analysis. Finally, the %C is a weight-based combination of various commonality metrics. The weightings are dependent on the systems engineers qualitative reasoning. This quality limits the index's ability to be an independent input in analyzing the product architecture's impact on the design process.

Commonality Index (CI) Another version of the DCI is the Commonality Index (CI) [88]. It provides a different view showing the number of unique parts present in a product line. Equation 10 shows the formulation of the CI, “where u is the number of unique parts, p_j is the number of parts in model j , and v_n is the final number of varieties offered [141].”

$$CI = 1 - \frac{u - \max p_j}{\sum_{j=1}^{v_n} p_j - \max p_j} \quad (10)$$

The CI is a simple index to compute and ranges from zero to one. It provides a quick and easy way to calculate the unique components within a product line but does not consider size, materials, manufacturing, and assembly processes used in

Table 19: Evaluation of the Commonality Index

	Represent the Compo- sition of Components	Independent of the Design Process Analysis	Shows Cha- racteristics among Components	Represent Across the Product Line	Independent of Qual. Decision Making
CI	✓	✓	✓	✓	✓

production when calculating the index.

Table 19 shows how the Commonality Index meets the criteria required of a commonality index. The CI is an extension of the DCI and TCCI. It compensates for the DCI's inability to represent the composition of commonality across the product line and for the TCCI's inability to differentiate between the common and unique components. As a result, the CI is the first index to meet all of the criteria.

Component Part Commonality Index ($CI^{(C)}$) All of the indices analyzed so far do not consider product volume, quantity per operation, and the cost of component part. Thus, the Component Part Commonality Index ($CI^{(C)}$) was developed to account for these considerations. Equation 11 shows the formulation of the $CI^{(C)}$, “where d is the total number of distinct component parts used in all the product structures of a product family, j is the index of each distinct component part, P_j is the price of each type of purchased parts or the estimated cost of each internally made component part, m is the total number of end products in a product family, i is the index of each member product of a product family, and V_i is the volume of end product i in the family. Φ_{ij} is the number of immediate parents for each distinct component part d_j over all the products levels of product i of the family. $\sum_{i=1}^m \Phi_{ij}$ is the total number of applications (repetitions) of a distinct component part d_j across all the member products in the family. Q_{ij} is the quantity of distinct component part d_j required by the product i [141].”

Table 20: Evaluation of the Component Part Commonality Index

	Represent the Compo- sition of Components	Independent of the Design Process Analysis	Shows Cha- racteristics among Components	Represent Across the Product Line	Independent of Qual. Decision Making
$CI^{(C)}$	○	⊘	✓	✓	✓

$$CI^{(C)} = \frac{\sum_{j=1}^d [P_j \sum_{i=1}^m \Phi_{ij} \sum_{i=1}^m (V_i Q_{ij})]}{\sum_{j=1}^d [P_j \sum_{i=1}^m (V_i Q_{ij})]} \quad (11)$$

Technically, the $CI^{(C)}$ can range from one to infinity, depending on the number of variants, components, and production quantities. This characteristic can cause problems in understanding the implications of the index especially if there is one really expensive component it will limit the effects of other components. Furthermore, the need to calculate the quantity demanded and cost of each component can be time consuming, at times using the metric is not feasible due to development scheduling.

Table 20 shows how the Component Part Commonality Index meets the criteria required of a commonality index. The $CI^{(C)}$ incorporates the costs of components into the formulation of the index. The inclusion of cost causes the index to range from zero to infinity. This characteristic limits the ability of the metric to determine the composition of commonality since the highest cost index can influence the metric the greatest. Also, the inclusion of cost is directly dependent on the design and production analysis. Therefore, instead of being an input the $CI^{(C)}$ is an output of the analysis, limiting the ability of the engineers to determine product architecture's impact on the selection process.

Comprehensive Metric for Commonality (CMC) After a comparison of the previously developed commonality indices, a new metric was developed to incorporate all of the benefits identified in the past. This index is the Comprehensive Metric for

Commonality (CMC) [143]. The CMC accounts for manufacturing processes, materials, assembly or fastening schemes, production volumes, and initial costs. Equation 12 shows the formulation of the CMC index, “where P is the total number of components. n_i is the number of products in the product family that have component i . f_{1i} is the ratio of the greatest number of products that share component i with identical size and shape to the number of products that have component i (n_i). f_{2i} is the ratio of the greatest number of products that share component i with identical materials to the number of products that have component i (n_i). f_{3i} is the ratio of the greatest number of products that share component i with identical manufacturing processes to the number of products that have component i (n_i). f_{4i} is the ratio of the greatest number of products that share component i with identical assembly and fastening schemes to the number of products that have component i (n_i). f_{1i}^{max} is the ratio of the greatest number of products that share component i with identical size and shape to the greatest possible products that could have shared component i with identical size and shape schemes. f_{2i}^{max} is the ratio of the greatest number of products that share component i with identical materials to the greatest possible number of products that could have shared component i with identical materials. f_{3i}^{max} is the ratio of the greatest number of products that share component i with identical manufacturing processes to the greatest possible number of products that could have shared component i with identical manufacturing processes. f_{4i}^{max} is the ratio of the greatest number of products that share component i with identical assembly and fastening schemes to the greatest possible number of products that could have shared component i with identical assembly and fastening schemes[143].”

C_i is the current total cost for component i : $C_i = \sum_{j=1}^{n_i} C_{ij}$, where C_{ij} is the total cost for component i variant j ($C_{ij} = Q_{ij} \times c_{ij}$), where Q_{ij} is the quantity and c_{ij} is the unit cost for component of component i variant j . C_i^{min} is the minimum total cost for component i : $C_i^{min} = \sum_{j=1}^{n_i} C_{ij}^{min}$. C_i^{max} is the maximum total component

Table 21: Evaluation of the Comprehensive Metric for Commonality

	Represent the Compo- sition of Components	Independent of the Design Process Analysis	Shows Cha- racteristics among Components	Represent Across the Product Line	Independent of Qual. Decision Making
CMC	✓	✗	✓	✓	✓

cost: $C_i^{max} = \sum_{j=1}^{n_i} C_{ij}^{max}$ [143].

$$CMC = \frac{\sum_{j=1}^P n_i \times (C_i^{max} - C_i) \times \Pi_{x=1}^4 f_{xi}}{\sum_{j=1}^P n_i \times (C_i^{max} - C_i^{min}) \times \Pi_{x=1}^4 f_{xi}^{max}} \quad (12)$$

The CMC metric encompasses all elements relevant in calculating the commonality of a product architecture. However, the use of costs, manufacturing processes, and assembly techniques rely on the outputs of design analyzes. Therefore, the metric is not an input but rather an output of the design process analysis.

Table 21 shows how the Comprehensive Metric for Commonality meets the criteria required of a commonality index. The CMC incorporates all of the characteristics of the previously reviewed commonality indices. As a result, the CMC meets all of the criteria, except for one. It relies on the design and production analysis to calculate the index. Therefore, it is not an input of the design process but rather an output, making it difficult to determine the product architecture's impact on the design process.

2.3.1.2 Reconfigurability Studies and Indices

Researchers developed reconfigurability indices to account for manufacturing processes, materials, assembly or fastening schemes, production volumes, and initial costs [143]. With varying respect to these considerations, many different commonality indices have been proposed.

Machine Reconfigurability Index (MR) A literature review found an index that represents the architecture's ability to change [62]. The Machine Reconfigurability Index (MR) was developed to analyze manufacturing reconfigurability. It can also be applied to a product architecture. The index ranges from zero to infinity. Infinity suggests there is an infinite number of configurations while zero means there is only one configuration in the product line. Equation 13 displays the formulation suggested by Goyal in his work. J_p is the total number of family members that belong to the product line, $\#Comp_{Tot_j}$ is the total number of components in a family member, and j represents the design index. Goyal added several more metrics including weights for online and offline reconfigurable components (emphasizing favor of one type of reconfigurability over another), and $\#Comp_{On_j}$ and $\#Comp_{Off_j}$ representing the number of online and offline reconfigurable components respectively.

$$MR = \frac{(J_p - 1)^2}{\sum_{j=1}^{J_p} \left(\frac{\alpha \times \#Comp_{Off_j} + \beta \times \#Comp_{On_j}}{\#Comp_{Tot_j}} \right)} \quad (13)$$

Since the metric does not range from zero to one, and there is no need for reconfigurability weightings, modified indexes were created for both online and offline reconfigurability. However, these indexes look at the interfaces between components. The reason for creating the indexes in this fashion is because online reconfigurability requires a joint or interface that creates a degree of freedom and offline reconfigurability requires a common interface between a platform component and its subcomponents.

Table 22 shows how the Machine Reconfigurability Index meets the criteria required of a reconfigurability index. The MR combines online and offline reconfigurability, through a weight-based scheme, into an overall reconfigurability index. Furthermore, it does not define what makes a component online or offline reconfigurable. The weight-based scheme and lack of clarity make the determining the composition of each type unclear.

Table 22: Evaluation of the Machine Reconfigurability Index

	Represent the Compo- sition of Components	Independent of the Design Process Analysis	Shows Cha- racteristics among Components	Represent Across the Product Line	Independent of Qual. Decision Making
MR	○	✓	✓	✓	⊘

Multi-Attribute Reconfigurability Index (MAR) Another reconfigurability index was created to account for the modularity, scalability, convertibility, and diagnosability [64]. The Multi-Attribute Reconfigurability Index (MAR) can be broken down into the listed characteristics. Modularity describes the product’s number of components and connectivity of components. Equation 14 shows the formulation of the Single Modularity Index (SMI) [68], “where σ_i are the singular values of the [design structure matrix (DSM)] arranged in a descending order, and N is the number of components (rows/columns in a DSM) [64].” The DSM is a matrix that show the connections between components within a product.

$$SMI = 1 - \frac{1}{N \times \sigma_1} \sum_{i=1}^{N-1} \sigma_i (\sigma_i - \sigma_{i+1}) \quad (14)$$

Scalability describes the ability to adjust the capacity of the product’s production. As a result, it is directly related to the ability to maintain cost effectiveness as the workload increases [64, 85]. Equation 15 shows the formulation of scalability or effectiveness, “where t_1 represents the time required to execute the work on a single processor, [...] k is the minimum number of operations required to complete the task, [...] and] N refers to the number of processors [64].”

$$Effectiveness = \frac{t_1 \times N}{k^2} \quad (15)$$

Convertibility captures the ability of a product to change its functionality or modify itself [64]. It is composed of three elements: configuration (C_C), machine (C_M),

and material handling (C_H) convertibility [86]. Overall convertibility is a weighted sum of all three. Equation 16 shows the formulation of configuration convertibility, “where R refers to the number of routing connections in each configuration, X is the minimum number of replicated machines at a particular stage in the process plan, and I is the minimum increment of conversion [64].”

$$C_C = \frac{R \times X}{I} \quad (16)$$

Equation 17 shows the formulation of machine convertibility, where C'_M refers to a single machine’s convertibility [64]. The C'_M should represent whether a machine is equipped with an automatic tool changer or multi head spindle, easily reprogrammed, with flexible software, equipped with flexible hardware components, equipped with flexible fixturing capability, or equipped with a large capacity tool magazine [64, 86].

$$C_M = \frac{\sum_{i=1}^N C'_M}{N} \quad (17)$$

Equation 18 shows the formulation of material handling convertibility, where C'_H refers to each material handling device’s convertibility. The C'_H assesses whether the device follows a free route, is multi-directional, is reprogrammable, has asynchronous motion, and is automatic [64, 86].

$$C_H = \frac{\sum_{i=1}^N C'_H}{N} \quad (18)$$

Diagnosability incorporates the time that passes until a failure occurs and is recognized, time until a failure will occur, and the time it takes to replace the component that was the cause of failure. The later is the distinguishability (D) which can represent diagnosability since it drives the time until the system is functional again. Equation 19 shows the formulation of distinguishability, “where n is the total number of possible indications, C_{LRU} is the total number of LRUs, C_i is the number of candidates for each indication i , and PI_i is the probability of indication i [64].”

Table 23: Evaluation of the Multi-Attribute Reconfigurability Index

	Represent the Compo- sition of Components	Independent of the Design Process Analysis	Shows Cha- racteristics among Components	Represent Across the Product Line	Independent of Qual. Decision Making
MAR	✓	○	✓	✓	⊘

$$D = \frac{\sum_{i=1}^n \left[PI_i \left(\frac{1}{C_i} - \frac{1}{C_{LRU}} \right) \right]}{\left(1 - \frac{1}{C_{LRU}} \right) \sum_{i=1}^n PI_i} \quad (19)$$

Gumasta combines modularity, scalability, convertibility, and diagnosability to create an overall reconfigurability index. The combination is a weight-based sum of the four variables. The use of weighting systems and some of the production and supply chain characteristics make the index dependent on analysis from the design process and at the whim of what the designers believe is important. Therefore, the index becomes less of an input but rather an output of the design process.

Table 23 shows how the Multi-Attribute Reconfigurability Index meets the criteria required of a reconfigurability index. The MAR combines production and material considerations into its formulation. Furthermore, it frequently relies on weightings to combine multiple concepts into its formulation. These two characteristics of the index do not make it independent of the design process analysis and mean the index can change depending on the systems engineer's whim. Overall the index is primarily developed to calculate the offline reconfigurability of the system with no consideration of online reconfigurability.

2.3.1.3 *Summary of Previously Developed Commonality and Reconfigurability Indices*

The indices identified provide multiple ways to represent the commonality and reconfigurability of a product architecture. Table 24 shows the review of all of the indices with respect to the criteria identified to represent the product architecture.

Table 24: Evaluation of the Previously Developed Commonality and Reconfigurability Indices

	Represent the Compo- sition of Components	Independent of the Design Process Analysis	Shows Cha- racteristics among Components	Represent Across the Product Line	Independent of Qual. Decision Making
Commonality					
DCI	⊗	✓	✓	○	✓
TCCI	○	✓	✓	✓	✓
PCI	✓	○	✓	✓	✓
%C	✓	○	✓	⊗	⊗
CI	✓	✓	✓	✓	✓
CI ^(C)	○	⊗	✓	✓	✓
CMC	✓	⊗	✓	✓	✓
Reconfigurability					
MR	○	✓	✓	✓	⊗
MAR	✓	○	○	✓	⊗

Concerning commonality, Table 24 shows that the CI meets all the criteria. It represents the composition of common components, is independent of the design process analysis, shows common characteristics among components, represents commonality cross the product line, and is independent of any qualitative decisions. Concerning reconfigurability, neither index identified meets all the criteria. MR can be translated from production analysis to product architecture analysis. However, the limited information on what makes a component online or offline reconfigurable makes the translation a bit difficult. Therefore, by taking some of the concepts from indices identified, two online and offline reconfigurability indices can be developed.

2.3.2 Complex System Definition

A complex system is “a system comprised of a (usually large) number of (usually strongly) interacting entities, processes, or agents, the understanding of which requires the development, or the use of, new scientific tools, nonlinear models, out-of-equilibrium descriptions and computer simulations [123].” In a design problem of a

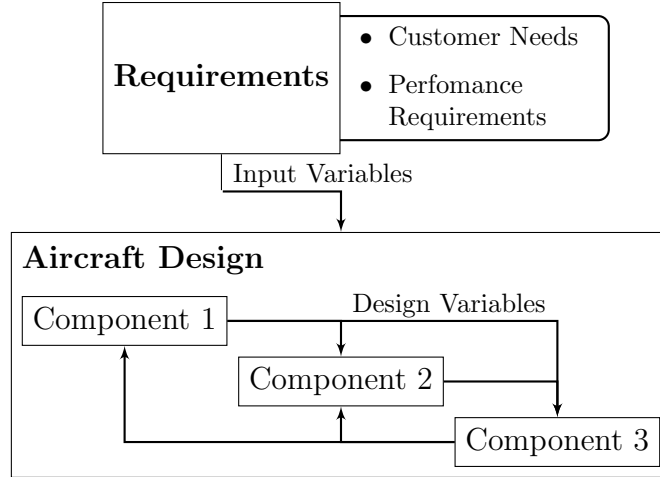


Figure 29: Example Complex System Design Structure

complex system, variables are input into component analyses determining the size and performance of the product. Figure 29 displays an example of the design problem.

During the design process, there are two types of variables: input and design variables. Input variables include requirements and technologies, and design variables are physical and performance characteristics. The problem tends to be highly nonlinear because of input interactions and design variable couplings. Interactions occur when one input variable's impact on a design variable depends on the value of another input variable [48]. Coupling is the inter-dependency/relation between two design variables [69].

The degree of inter-dependency relates to a systems complexity. In academia and industry, there are many ways to calculate a product's complexity, including information in design, traditional in design, as size, as coupling, and as solvability [139, 50].

Information complexity in design takes the relationship between design requirements and parameters to determine the independence and coupling among the design requirements and determines the uncertainty associated with the design problem. These two concepts create real and imaginary complexity. The first relating to the probability the product will achieve all the functional requirements, and the second

relating to the probability the product will accidentally achieve a requirement.

Traditional complexity in design takes a hierarchical approach to quantify complexity. A product's purpose or function consists of multiple levels. For example, a functional flow block diagram breaks down the task into multiple stages and levels. This approach combines the number of functions at each level to calculate complexity.

Complexity as size uses the degrees of freedom and interfaces amongst variables to calculate the difficulty of the design problem. The size includes the number of independent and dependent design variables, design requirement, measures of performance, independent and dependent modules, and evaluation or synthesis operators.

Complexity as coupling uses graph theory to organize connections among components or disciplines through connectivity. The connections are the dependent variables that rely on outputs of other modules. The number of connections then provide a surrogate for the calculation.

Complexity as solvability determines how difficult it is to execute the design process. Instead of looking at the size of the problem, it looks at the number of analyses and processes the design of the product requires. This approach focuses on the process rather than the problem. Therefore, it relates to complexity as size but varies slightly.

The exploration of existing methods to calculate complexity outlines reasons and goals of the research. This dissertation proposes using complexity to evaluate product architectures. From this research, a new framework must calculate complexity that is related to the past methods, combining elements of each.

2.3.3 Difference between Robust Design, Flexibility, and Complexity

Before this dissertation continues, it is important to distinguish between robust design, flexibility, and complexity.

Robust Design “is a design methodology developed in order to make a product's

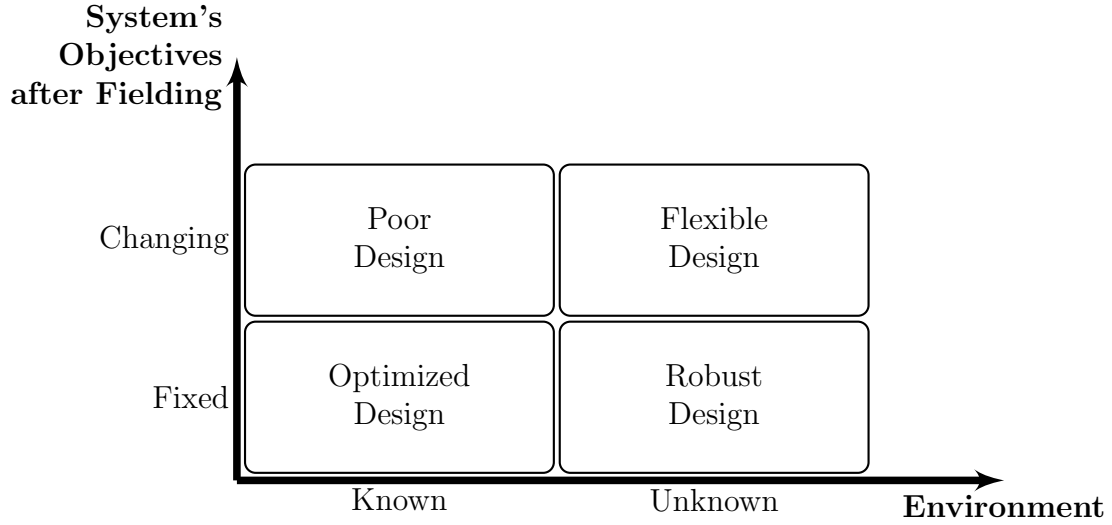


Figure 30: Flexibility and Robustness as a Function of the System's Objectives and Environment [126]

performance insensitive to raw material variation, manufacturing variability, and variations in the operating environment [114].” Therefore, as the environment surrounding the product, design, process, or other engineered product changes the product will still perform at a high standard.

Flexibility “implies an ability of the design to be changed in order to track requirement changes [127, 126],” suggesting if a product is flexible and an operator uses it for a purpose other than originally designed for, then it will still perform well.

The best way to the differential between the two is in Figure 30, which displays what type of design engineers should implement relative to the environment and system objectives.

Figure 30 shows three options and one poor design region. If the environment is known and the system's objectives are constant, then the engineers can develop an optimized product. Second, if the environment is unknown, but the system's objectives are constant, then the engineers must employ robust design methods to ensure the system operates consistently over changing environmental conditions. Third, if

the environment is unknown and the system's objectives are changing, then the engineers must develop a flexible design that performs consistently over multiple tasks and environmental conditions. The last case occurs if the environment is known, but the system's objectives are changing. If the design experiences these conditions, then the engineers probably blundered setting up the problem or created a faulty product.

Complexity is characterized as the combination of [21]:

- Numbers: number of domains, functions, or disciplines
- Degree of Interdependency: level of interdependency among the domains, functions, or disciplines
- Intricacy or difficulty: novelty of project
- Limitations: level constraints' stringency

Complexity varies from flexibility because it looks at the inner workings of the product. It increases with the number of components or subsystems, the number or strength of constraints, interdependency between the components or limitations, and whether the designers/engineers have approached a problem like this one before. When it comes to the cost of producing a complex product, it is not only important to understand the product itself, but also the business or entity that will be producing a product. A complex product requires many people working in different disciplines. Therefore, one of the best ways to combat complexity is to ensure clear and concise communication amongst teams. Furthermore, the best way to determine if a business or entity will succeed producing a product is to see whether the organizational structure is built to handle the problem.

Though there are many ways to evaluate architectures, this dissertation focuses on desirability, flexibility, and complexity. Many “-ities” exist to evaluate architectures, such as maintainability, availability, reliability, manufacturability, and more.

Depending on the design problem, it may be advisable to switch out or include other metrics. However, based on the background research, this dissertation recommends desirability, flexibility, and complexity since these metrics apply to the development and production of the product.

2.3.4 Software's Impact on the Product Architecture

Unmanned aerial vehicles have various levels of autonomy. They can fly supervised, semi-, or entirely autonomously. For each of these states, the pilot or computer controlling the aircraft's flight must rely on data provided by the vehicle's sensors. Today, much of this information is preprocessed by a computer on board, using complicated software embedded within. When a system engineer analyzes the Dynamic Operational Architecture of a UAV, it is impossible to ignore the impact software has on the product's capability and cost of the design.

The software's relation with the design and product architecture is purely a capability-cost trade-off. Since the software controls many of the electronics and subsystems in modern products, the capability of the system relates to the complexity of the software, meaning the higher the capability - the higher the cost and of the software's development. Therefore, the software development's cost directly impacts the choice of product architecture. As the functions or controls of the system increase, the software incorporated into the product line requires more lines of code and greater effort. Since cost is a major factor in the selection of a product architecture, the increase in software development cost could impact the final decision in which product architecture to implement.

The software's complexity can be broken down two ways. First, it depends on the controllability of the system. If the system's behavior is physically or conceptually unstable, the software is given greater responsibility to make the system stable or controllable. Therefore, if the behavior is non-linear or categorical, then more modules

or software functions are required. The number of modules and relations amongst them requires many lines of code often in the order of magnitudes of thousands or millions. In the future, the magnitude could be even higher.

The second in the number of functions relating to process subsystem information. Software controls functions of individual components, and in an integrated system, the components must work together to achieve the tasks required. Therefore, more lines of code are required to integrate the subsystems and sub-functions. The order of magnitude of lines of code is proportional to the factorial of the number of sub-functions [139]. For example, approximately 90% of the F-35s functions are managed by software [21]. Thus, the F-35's product development costs consisted of 20% software development costs [21].

Figure 31 displays the exponential growth of software in the aerospace industry over the last century. The number of lines of code relates to the time and cost of software development. Therefore, designers must consider software development's impact on performance, cost, and therefore the architecture.

Since the software controls many of the functions and autonomous activities that occur while operating a product, the cost for software development can be approximated from the number of functions the software package must control during its operations. With this knowledge, the model used to determine the cost of software development can be chosen. Therefore, to calculate the cost it takes to develop the software required of a product a model must:

- Calculate the total cost it will take to develop the software
- Relate the functions the software must conduct to the cost of the software
- Be available to the organization developing the product (If the company does not want to pay for proprietary programs, the model must be publicly available)

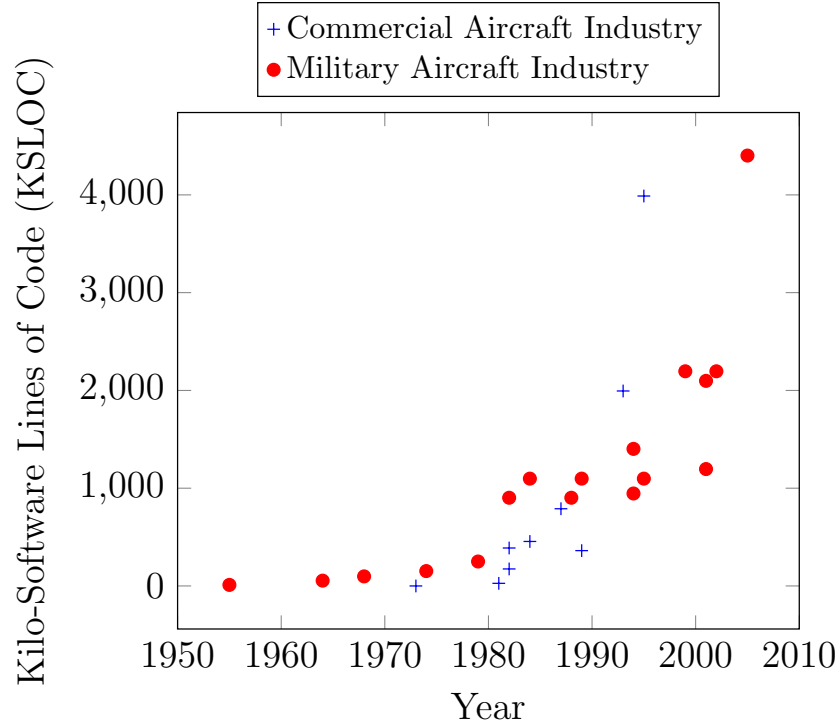


Figure 31: The growth of Software in the Aerospace Industry [5]

2.3.4.1 Existing Estimation Models of Software Development Cost and Time

There are multiple models that estimate the effort (cost and time) to develop the software required of a new system. An extensive literature review identified three models of interest. These models are the Air Force Cost Analysis Agency's Revised Enhanced Version of the Constructive Cost Model (AFCAA REVIC / COCOMO), Galorath's System Evaluation and Estimation of Resources - Software Estimating Model (SEER-SEM), and Quality Software Management's Software Life-cycle Management Estimate (SLIM-Estimate). These models are primarily proprietary but their development were based on a couple publicly available concepts.

Air Force Cost Analysis Agency Revised Enhanced Version of the Constructive Cost Model (AFCAA REVIC) The AFCAA REVIC is a model developed by the United States Air Force to predict the effort (person-months) and time it takes to develop software. Its based on the Constructive Cost Model (COCOMO)

Table 25: Equation 20 A Coefficient Values [81]

Model	Organic	Semi-detached	Embedded
A	3.2	3.0	2.8

which was developed in 1981 by Barry Bohem [81]. The model uses simple power functions to approximate the effort and time to develop software. Equation 20 shows how COCOMO calculates the effort required to develop the software, where E is the effort (person-months), A is a work-environment coefficients (shown in Table 25), SF_j are scaling exponents, EM_i are cost drivers, and $KSLOC$ is the size of the project (kilo-lines-of-code).

$$E = A \times KSLOC^{1.01 + \sum_{j=1}^5 SF_j} \times \prod_{i=1}^{17} EM_i \quad (20)$$

The A coefficients in Equation 20 depend on the type of work environment the software is developed in. The three options are [81]:

- Organic “relativity small software teams developing software in a highly familiar, in-house environment”
- Embedded “operating within tight constraints where the product is strongly tied to a complex of hardware, software, regulations and operational procedures”
- Semi-detached “an intermediate stage somewhere in between organic and embedded”

After determining the type of work environment, Table 25 shows how the work environment relates to the coefficients value.

For the scaling exponents and cost drivers, Table 26 shows the various cost drivers and their corresponding coefficients. The cost drivers are qualitative and have six levels: very low (VL), low (L), nominal (N), high (H), very high (VH), and extra high (XH).

Table 26: Equation 20 Scaling Exponents (SF_j) and Cost Drivers (EM_i) [45]

Driver	Sym	VL	L	N	H	VH	XH
Precendentedness	SF ₁	0.05	0.04	0.03	0.02	0.01	0.0
Development Flexibility	SF ₂	0.05	0.04	0.03	0.02	0.01	0.0
Architecture and Risk Resolution	SF ₃	0.05	0.04	0.03	0.02	0.01	0.0
Team Cohesion	SF ₄	0.05	0.04	0.03	0.02	0.01	0.0
Process Maturity	SF ₅	0.05	0.04	0.03	0.02	0.01	0.0
Required Software Reliability	EM ₁	0.75	0.88	1.00	1.15	1.40	
Data Base Size	EM ₂		0.94	1.00	1.08	1.16	
Product Complexity	EM ₃	0.75	0.88	1.00	1.15	1.30	1.65
Required Reusability	EM ₄		0.89	1.00	1.16	1.34	1.56
Documentation Match to LC Needs	EM ₅	0.85	0.93	1.00	1.08	1.17	
Time Constraint	EM ₆			1.00	1.11	1.30	1.66
Storage Constraint	EM ₇			1.00	1.06	1.21	1.56
Platform Volatility	EM ₈		0.87	1.00	1.15	1.30	
Analyst Capability	EM ₉	1.5	1.22	1.00	0.83	0.67	
Programmer Capability	EM ₁₀	1.37	1.16	1.00	0.87	0.74	
Personnel Continuity	EM ₁₁	1.26	1.11	1.00	0.91	0.83	
Applications Experience	EM ₁₂	1.23	1.10	1.00	0.88	0.80	
Platform Experience	EM ₁₃	1.26	1.12	1.00	0.88	0.80	
Language and Tool Experience	EM ₁₄	1.24	1.11	1.00	0.9	0.82	
Use of Software Tools	EM ₁₅	1.20	1.10	1.00	0.88	0.75	
Multi-Site Development	EM ₁₆	1.24	1.10	1.00	0.92	0.85	0.79
Required Development Schedule	EM ₁₇	1.23	1.08	1.00	1.04	1.10	

Table 27: Evaluation of the AFCAA REVIC or COCOMO Model

	Ability to Calculate Software Development Cost	Connects Software Functions to Cost	Publicly Available (Optional)
COCOMO	✓	✗	✓

The problem with using the REVIC or COCOMO model is the dependence on already having the size of the software determined before using the model. Also, there is no link between the functions the software must perform and the effort required for its development.

Table 27 shows how the AFCAA REVIC or COCOMO Model meets the criteria set for a software development cost model when analyzing the product architecture. The COCOMO model calculates the effort of the project which can be translated to cost by multiplying the effort by a nominal computer scientist salary. Also, the model’s parameters have been released allowing for the model to be implemented, even if the organization developing the product does not want to pay for a proprietary program. However, its inability to relate the software’s functions to its costs fail to relate the software to the product architecture.

Galorath System Evaluation and Estimation of Resources - Software Estimating Model (SEER-SEM) The second model is Galorath’s System Evaluation and Estimation of Resources - Software Estimating Model (SEER-SEM). The model can predict the effort (person-months) and time required to develop software. It “began with the Jensen model and diverged significantly in the early 1990s. Barry Boehm’s Constructive Cost Model work provided for the redefinition of some of the original Jensen model parameters into SEER-SEM [57].”

The model first estimates the size of the project by either using an estimate or by using function-based sizing. Equation 21 shows how SEER-SEM predicts the

effective size of the project (S_e), where L_x is a language-dependent expansion factor; $AdjFactor$ is a combination of factors that consider phase at estimate, operating environment, application type, and application complexity; UFP is the function-based sizing, unadjusted function points required in the software to achieve the desired functions; and $Entropy$ ranges from 1.04 to 1.2 depending on the type of software being developed [57].

$$S_e = Lx \times (AdjFactor \times UFP)^{\frac{Entropy}{1.2}} \quad (21)$$

The effective size of the project is not necessarily the lines of code in a project, rather an internal size parameter within SEER-SEM. After being calculated, the effective size is used to predict the effort required to develop the software for the project. Equation 22 shows how SEER-SEM calculates the effort required to develop the software, where C_{te} is effective technology a composite metric that captures factors relating to the efficiency or productivity with which development can be carried out and D is staffing complexity a rating of the project's inherent difficulty in terms of the rate at which staff are added to a project [57].

$$E = D^{0.4} \frac{S_e^{1.2}}{C_{te}} \quad (22)$$

SEER-SEM is a proprietary tool which coefficients are not publicly available even though Galorath did release the functions and structure of the model it uses. Though the model itself is not usable without paying for the SEER license, reviewing the model shows there are methods that predict the size and effort of software development based on the functions required of the software.

Table 28 shows how the SEER-SEM Model meets the criteria set for a software development cost model when analyzing the product architecture. SEER-SEM calculates the effort of the project which can be translated to cost by multiplying the effort

Table 28: Evaluation of the SEER-SEM Model

	Ability to Calculate Software Development Cost	Connects Software Functions to Cost	Publicly Available (Optional)
SEER-SEM	✓	○	⊘

by a nominal computer scientist salary. Also, SEER-SEM shows there are was to calculate the size of the project to the number of functions the software must control. However, the model’s parameters have not been released, meaning the organization has to pay for a proprietary program.

Quality Software Management’s Software Life-cycle Management Estimate (SLIM-Estimate) and Putnam’s Model QSM’s SLIM-Estimate model is based off of Lawrence Putnam’s model he developed in 1978 [120]. The model predicts the time and effort (person-months) to develop software. Equation 23 shows how SLIM calculates the effort, where B is a special skills factor and a function of the project size, $SLOC$ is the project size in lines of code, P is a productivity parameter which accounts for the ability of the organization to produce software at a particular defect rate, and $Time$ is the time allotted for the software’s development [92]. Tables 29 and 31 provide reference values of the special skills factor (B) and the productivity parameter (P).

$$E = 12^5 B \left(\frac{SLOC}{P} \right)^3 \frac{1}{Time^4} \quad (23)$$

Since many firms might not have all the software development’s scheduling information, Putnam’s Model and SLIM uses the default formula in Equation 24 [120].

$$E = 56.4B \left(\frac{SLOC}{P} \right)^{9/7} \quad (24)$$

Table 2 shows general references of what the special skills factor (B) should be based on the project size ($SLOC$).

Table 29: Putnam Special Skills Factor [92]

Size (SLOC)	B
5-15K	0.16
20K	0.18
30K	0.28
40K	0.34
50K	0.37
>70K	0.39

Table 31 provides references for the productivity parameter. The table also provides specific examples of applications that allow the user of the model to have baselines of the parameter's values based on past projects. Putnam's model is easy to use and much of the parameters are publicly available. However, the model still relies on an estimate of the project's size to calculate the effort or time to develop the software.

Table 30 shows how the QSM SLIM-Estimate or Putnam Model meets the criteria set for a software development cost model when analyzing the product architecture. The Putnam Model calculates the effort of the project which can be translated to cost by multiplying the effort by a nominal computer scientist salary. Also, the model's parameters have been released allowing for the model to be implemented, even if the organization developing the product does not want to pay for a proprietary program. However, its inability to relate the software's functions to its costs fail to relate the software to the product architecture.

Table 30: Evaluation of the QSM SLIM-Estimate or Putnam Model

	Ability to Calculate Software Development Cost	Connects Software Functions to Cost	Publicly Available (Optional)
Putnam	✓	✗	✓

Table 31: Putnam Productivity Parameter [92]

Productivity Index	Productivity Parameter	Application Type
1	754	
2	987	Microcode
3	1,220	
4	1,597	Firmware (ROM)
5	1,974	Real-time embedded, Avionics
6	2,584	
7	3,194	Radar systems
8	4,181	Command and control
9	5,186	Process control
10	6,765	
11	8,362	Telecommunications
12	10,946	
13	21,892	
14	13,530	Systems software, Scientific systems
15	17,711	
16	28,657	Business systems
17	35,422	
18	46,368	
19	57,314	
20	75,025	
21	92,736	
22	121,393	
23	150,050	
24	196,418	
25	242,786	Highest value found so far
26	317,811	
27	392,836	
28	514,229	
29	635,622	
30	832,040	
31	1,028,458	
32	1,346,269	
33	1,664,080	
34	2,178,309	
35	2,692,538	
36	3,524,578	

Table 32: Evaluation of the Identified Software Models

	Ability to Calculate Software Development Cost	Connects Software Functions to Cost	Publicly Available (Optional)
COCOMO	✓	⊘	✓
SEER-SEM	✓	○	⊘
Putnam	✓	⊘	✓

2.3.4.2 Summary of Software Development Cost Estimation

After reviewing previously developed software development cost estimation, all of the identified models do not meet the criteria required to relate the software's development to the product architecture. Table 32 shows the results of the review. Since none of the models meet the criteria, either a new model must be created or the models must be combined.

Development of a new model is outside the scope of the research and could be costly and time consuming. As a result, either the COCOMO or Putnam model can be combined with a function-based sizing model creating the connection between the software's functions and the cost. Observing SEER-SEM, it used a function-based sizing model to predict the size of the project. So first, it is important to understand unadjusted function points.

Function point analysis is a way to translate any type of coding language to a standard way of calculating the size of a software project [84]. It takes complicated concepts of coding and translates them into five major components: external inputs, external outputs, external inquiries, internal logical files, and external interface files. An external input is a process where data flows across some sort of boundary into the project or software package in question. An external output is a process where data flows across some sort of boundary out of the project or software package in question. An external inquiry is a process where an external entities retrieves data

Table 33: Calculating the Total Number of Unadjusted Function Points [84]

Type of Component	Complexity of Components			
	Low	Average	High	Total
External Inputs	_ x 3 = _	_ x 4 = _	_ x 6 = _	
External Outputs	_ x 4 = _	_ x 5 = _	_ x 7 = _	
External Inquiries	_ x 3 = _	_ x 4 = _	_ x 6 = _	
Internal Logical Files	_ x 7 = _	_ x 10 = _	_ x 15 = _	
External Interface Files	_ x 5 = _	_ x 7 = _	_ x 10 = _	
Total Number of Unadjusted Function Points =				

from one or more internal logical files and external interface files. An internal logical file is “a user identifiable group of logically related data that resides entirely within the applications boundary and is maintained through external inputs [84].” Finally, an external interface file “a user identifiable group of logically related data that is used for reference purposes only [84]” to relate the package with others.

Breaking down the required functions into these components allows for the standardization of software development analysis. After breaking down the functions into components the engineer must bucket the components into three levels of complexity. Sorting the components allows the unadjusted function points (UFPs) to be calculated using the structure, found in Table 33.

After calculating the number of UFPs, they still need to be converted to the lines of code. Through the review of past methods, Table 34 was discovered. It provides estimates of the lines of code per UFP for the following coding languages.

After reviewing all of the concepts present in software development cost estimation, a process can be formulated combining certain concepts. As a result, the proposed process consists of the following steps:

1. Break down the product into its individual components and subsystems
2. Identify which components or subsystems have internal software packages
3. Identify which component’s functions are controlled by a central processor

Table 34: Converting UFP to SLOC [92]

Language	SLOC/UFP
Ada	71
AI Shell	49
APL	32
Assembly	320
Assembly (Macro)	213
ANSI/Quick/Turbo Basic	64
Basic-Compiled	91
Basic-Interpreted	128
C	128
C++	29
ANSI COBOL 85	91
Fortran 77	105
Forth	64
Jovial	105
Lisp	64
Modula 2	80
Pascal	91

4. Calculate all of the UFPs for the components or subsystems with internal software packages
5. Calculate the UFPs required to integrate all components and subsystem functions to the central processor
6. Convert the total-integrated UFPs into lines of code (Average Value 100 SLOC/UFP - Table 34)
7. Use any of the identified models to calculate the effort required to develop the software (Depending on preference on the use of proprietary models)
8. Convert effort into cost using a nominal salary of a coder or computer scientist (~\$80,000 US2017 [7])

This approach will give an estimate of the cost required to develop the software that controls many of the functions in a product. Chapter 4 provides more details on

the calculation of the UFPs associated with each component.

2.3.5 Additional Required Concepts Conclusion

This section defined additional concepts that are relevant to product architecture selection. By implementing product architectures, the designer is creating a complex problem. Therefore, this section introduced the definition and concepts of complex systems. Furthermore, the analysis of past product architecture selections shows designers often consider flexibility and complexity as essential concepts when selecting the appropriate product architecture. Flexibility is easily confused with robust design. Therefore, the comparison between the two was presented to clarify their difference. Finally, this section introduced the importance of software development due to UAVs' and other modern products' reliance on software to control many of the product's activities.

Now that the concepts that dominate product architecture selection have been introduced, the next section will introduce the research objective followed by the research question that will help formulate a new framework to facilitate systems engineers in the selection of a product architecture.

2.4 Research Objective

As stated in Section 1.4, products are traditionally sized to fulfill one primary mission. When multiple missions are involved, the designer makes compromises, to make the aircraft perform robustly in all conditions. Compromises tend to result in decreased performance. Thus, new product architectures have been introduced to reduce the losses in performance, maintain manufacturability, increase product flexibility, and reduce design complexity. Conventional product architecture selection tends to be a heuristically based process where experts down-select options without fully exploring the product architecture space. This dissertation shifts away from this paradigm. Furthermore, as stated in Section 1.4, it is important to select the most favorable

architecture early in the design process since it will influence subsequent steps. Therefore, a new framework must be formulated to facilitate the decision-making process. The framework must be able to compare many competing architectures, consider changes in design drivers (market, technology, and performance requirements), and aid system architects in performing trade-offs between competing architecture designs. Specifically, the framework will:

- Derive need of the product from the customer(s) or business strategy
- Implement a method to clearly define the product’s functional requirements from the needs
- Provide an understanding of how and which functional requirements drive the architecture
- Provide the architect with insights on what types of product architectures are favored
- Implement a way to characterize and explore the space of alternative product architectures
- Derive what qualifies a “good” product architecture

All these considerations provide the research objective stated below:

Research Objective:

Formulate a framework that aids the system architect in choosing the most appropriate product architecture when developing vehicles and planning their evolution.

The result of this dissertation is a transition from qualitative to quantitative analysis of product architectures. Furthermore, it will provide an understanding of key

performance and manufacturing requirements that drive design considerations and a traceable link to the impact of key product requirements that drive design considerations. The benefit of this dissertation is a method/framework that provides a means to produce and evaluate alternative product architectures concerning changing and fixed requirements and increase the traceability of the product architecture selection throughout the design process.

2.5 Formation of Research Questions

The research objective, as stated in Section 2.4, requires a new architecture selection framework to be formulated. The framework must satisfy the criterion listed in Section 2.2.11. Thus, it follows the generic engineering decision support process, which follows six steps outlined in Figure 32. The steps consist of establishing the need for the product, define the problem or functions the product must perform, establishing how the product will be evaluated, generate alternative product architectures, analyze these alternatives, and make a decision about which product architecture to implement.

The need for a new product arises from a problem or gap identified by potential customers or a business strategy developed by the manufacturer. General practices can be used to **establish a need** for a new product. Analyzing the industry and the manufacturer's capabilities can ensure the development of the product is in the manufacturer's interest and whether the product can accomplish the need.

The functional requirements help **define the problem** the product is hoping to address. Traditional systems engineering practices such as requirement analysis and functional allocation and analysis can set up the design of the system and drive the selection of the architecture. Here, the engineers set the functional requirements and select the configuration (physical architecture) of the system. This dissertation

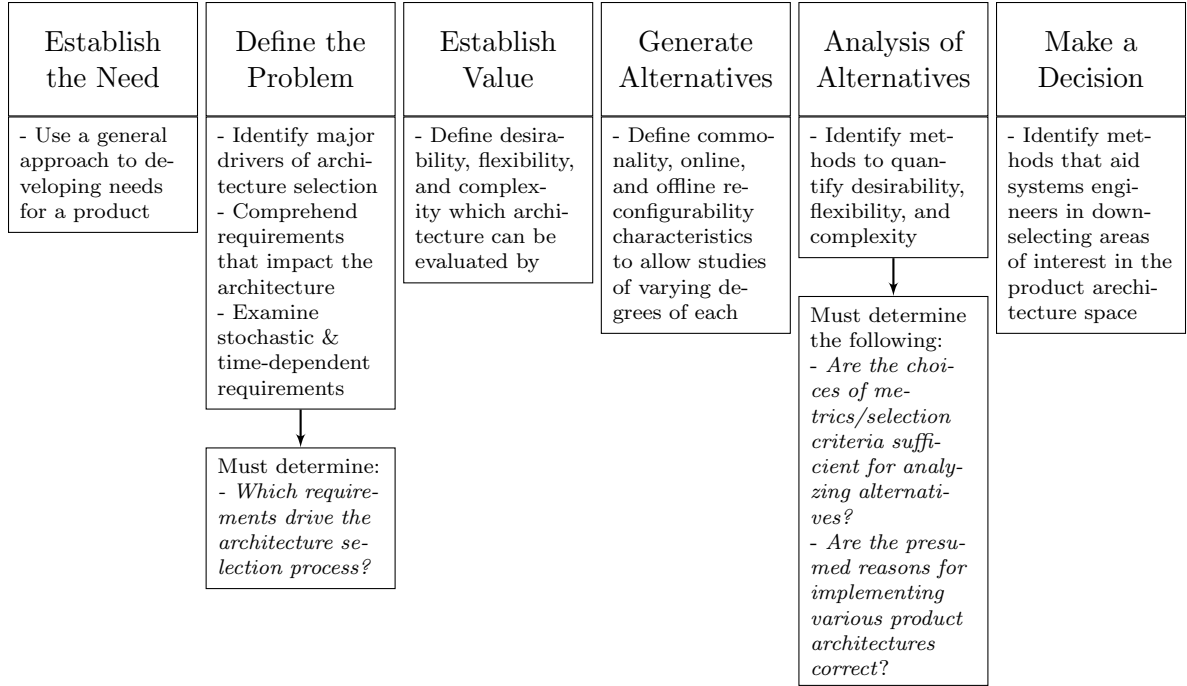


Figure 32: This Dissertation's Framework Overview

attempts to identify which of these requirements drive the product architecture implemented.

From the background research, specifically the investigation of industries' history and past architecture sections (Section 1.5.1), product architectures implementation trades among desirability, flexibility, and complexity. Before generating and analyzing various alternative product architectures, the system engineers must **establish value** (metrics) that can evaluate the product architectures.

Product architectures implement three characteristics among the interfaces of the components or subsystems. Therefore, designers can vary commonality, online and offline reconfigurability to **generate alternative** product architectures. A numerical representation of the product architecture space can be used to capture the impact of varying all three characteristics.

Desirability is the ability of the product to achieve performance and cost requirements; flexibility is the ability of a product to conduct tasks other than those initially

intended, and complexity is the difficulty in designing the product. **Analyzing Alternative** product architectures requires comparing the three terms. Therefore, all of these terms must be defined and given numerical representations to evaluate product architectures.

Once the method generates alternative product architectures, **analysis of the alternatives** provides the designer or system engineer with a wealth of knowledge about the problem. The designers must implement methods to quantify desirability, flexibility, and complexity.

Finally, the designer must **make a decision** on what levels of commonality, on-line, and offline reconfigurability the architecture should have. The designers must implement a method or techniques to help identify areas of interest in the product architecture space. However, before real analysis begins, the metrics determining the desirability, flexibility, and complexity of the product must be justified as sufficient. Then, the method of down-selecting regions of interest can be warranted as well.

The following subsections propose questions that drive the formulation of this framework.

2.5.1 Research Question 1: Establish the Need for a New Product

A manufacturer must have a reason to develop a new product. Development of a new product requires the manufacturer's understanding of the industry/market's state and trends, including opportunities, threats, risk factors, and constraints. Therefore, the need can be either derived or received from the external factors impacting the manufacturer. However, the manufacturer must also be able to recognize their internal strengths and capabilities, ensuring the manufacturer can produce the proposed product. This process of analysis will allow the manufacturer to formulate a product-based/customer-oriented strategy. The analysis and formulation of a need for a new product require structured methods or techniques, leading to Research Question 1:

Research Question 1:

Which methods can a manufacturer and its engineers utilize to establish a need for a new product and develop inputs to facilitate the formation of functional requirements of the product?

Section 3.1 describes the process of formulating a product-based/customer-oriented strategy. Once the manufacturer formulates its business strategy, requirements can be derived from the need, defining the product.

2.5.2 Research Question 2: Define the Problem or Requirements of the New Product

Once the manufacturer establishes the need for a new product, the manufacturer must derive requirements that define and constrain it. The derived requirements influence manufacturers to favor certain architectures over others. These requirements include functions, technology levels, life cycle costs, environmental considerations, and legal considerations. They often evolve with time, causing requirement drift. The future market directly affects the architecture selection decision since the manufacturer wants to dominate the market with the selected product architecture for the longest period, often requiring the architecture to change. In this dissertation, drivers are the requirements that drive the choice in a product architecture. These considerations lead to Research Question 2:

Research Question 2:

What are the typical design drivers that lead system architects towards different architecture implementation strategies (fixed, reconfigurable, or product family)?

Once the research identifies the drivers, designers must have the ability to determine trends among how the requirements or drivers are structured and the selected product architecture, leading to Research Question 2.a:

Research Question 2.a:

Is there some effect from how drivers are structured early in the design process that tends to favor one implementation strategy over another?

Finally, designers must have the ability to identify trends among the changing requirements/drivers and the selected product architecture, leading to Research Question 2.b:

Research Question 2.b:

How do we capture the impacts of changing drivers on the product architecture?

Section 3.2 describes the process of determining product architecture drivers. This dissertation conducts studies on the drivers impact on the product architecture, allowing for the formulation of a new framework.

2.5.3 Research Question 3: Establish Value of the Product Architecture

Due to highly complex nature of modern products and their architectures, this dissertation creates new evaluation criteria. Traditionally, metrics are used to evaluate configurations which included but were not limited to performance and cost. Because of the complication from adding new architectures and configurations, it is of even greater importance to compare alternatives ability to satisfy customer demands and manufacturer goals. These needs for new way to establish value of product architecture, leading to Research Question 3:

Research Question 3:

What quantifiable ways can engineers utilize to determine whether an architecture is “good” or favorable?

2.5.4 Research Question 4: Generating Alternative Product Architectures

Defining a system's architecture is not transparent due to qualitative definitions of architectures. Many systems contain elements of multiple architectures making a numeric definition of the architecture space difficult. Competing product architecture alternatives will vary regarding the following key component characteristics: online, offline reconfigurability and commonality, as identified in Section 1.3, resulting in various levels of modularity, customization, and reconfigurability. The vague properties of the architecture space lead to Research Question 4:

Research Question 4:

What methods can be used to aid in the generation of alternative product architectures?

A quantitative architecture space is defined in Section 3.4 to provide the architect with a means to map and compare different designs concerning each other. Defining the architecture space also simplifies any trade-off or optimization analysis. Without a quantifiable architecture space, creation of a framework is impractical.

2.5.5 Research Question 5: Analysis of Alternative Product Architectures

As stated in Section 2.3.3, there are many ways to evaluate architectures, such as maintainability, availability, reliability, manufacturability, and more. Depending on the design problem, it may be advisable to switch out or include other metrics. However, based on the background research, the focus of this dissertation will be on desirability, flexibility, and complexity, leading to Research Question 5:

Research Question 5:

How are product architectures evaluated in a way that determines a product architecture's ability to satisfy requirements, resilience to changes in the industry associated with time, and the internal difficulty of developing and producing the new product quantified (desirability, flexibility, and complexity)?

First, desirability relates to the architectures ability to achieve customer demands and manufacturer's goals. Second, flexibility is the system's ability to be used for purposes or tasks other than originally designed. Third, complexity relates to the system's difficulty to develop, produce, and support. The research in this dissertation provides quantifiable representations of all three that captures each defined quality.

The requirement to quantify desirability leads to Research Question 5.a:

Research Question 5.a:

How is the desirability of the product architecture determined?

More research is required to define flexibility applied to product architectures and provide a mathematical representation, leading to Research Question 5.b:

Research Question 5.b:

What is an appropriate definition and quantification of flexibility in the context of product architectures?

Once flexibility concerning product architecture selection has been identified, more research is required to define complexity concerning product architectures and provide a mathematical representation, leading to Research Questions 5.c:

Research Question 5.c:

What is an appropriate definition and quantification of complexity in the context of product architectures?

Section 3.5 describes the formulation of new criteria.

2.5.6 Research Question 6: Determining Areas of Interest in the Product Architecture Space

System engineers need to identify types of architectures and regions in the architecture space that are favorable. The research in this dissertation must describe a technique that allows for the analysis and down-selection of alternative architectures, leading to Research Question 5:

Research Question 6:

What techniques can facilitate the process of analyzing and down-selecting regions of interest in the product architecture space?

Section 3.6 describes the techniques that can facilitate the analysis of alternatives and decision-making process.

2.6 *Summary of Background Research*

Throughout the Background Research, this chapter reviews the systems engineering process (Section 2.1). It provided a benchmark of how the product architecture is selected. The system engineering process consists of analyses that break down the problem and logically develop a conceptual product that meets customer needs and manufacturing entity requirements. Following the review of the systems engineering process, Section 2.2 conducted an extensive review of existing methods for physical, system, and product architecture selection. Section 2.2 identified gaps in current architecture selection techniques and develops criteria a new framework must have.

The review of past methods and the product architecture selection problem identified a couple key concepts that influence the process. Section 2.3 reviews these concepts which were commonality and reconfigurability indices, complex system analysis, requirement flexibility and design complexity, and software development. The commonality and reconfigurability indices provide a way to represent a product architecture numerically. Also, implementing the product architecture has a direct impact on complex system analysis. Complex system analysis involve managing interactions between requirements and couplings between design variables. These concepts relate to the requirement flexibility and design complexity and Section 2.3.3 defines both. Finally, in the modern era all products contain some software that controls their functions. Thus, software development is an emerging discipline that influences the product architecture selection process. Section 2.3.4 introduces a way to calculate the cost of software development as it relates to the product architecture. Finally, with all of the concepts explored and reviewed, Section 2.4 defines the objective of the research which calls for a new framework that facilitates in the product architecture selection process. Next, Section 2.5 provides questions that serve to guide the research during the development of the new framework.

In the past, designers have down selected architectures before analyzing the architecture space. Furthermore, the prior methods lack analyses that determine the architectures' performance, cost, and resilience to design and requirement changes. The next chapter presents the formulation of the new framework.

CHAPTER III

FORMULATION OF FRAMEWORK FOR PRODUCT ARCHITECTURE ANALYSIS OF UNMANNED SYSTEMS AND TECHNOLOGIES: FA²UST

The proposed framework aims to satisfy the concerns and gaps of existing product architecture selection methods identified in Chapter 2. Most of the gaps presented in the previous chapter concern the inclusion of the business strategy in the analysis, full exploration of the product architecture space, and the ability to analyze the product architecture's impact on the interactions and couplings present in complex system design. Therefore, all of the concerns are incorporated into one framework: FA²UST. The framework consists of six stages:

1. Establishing a need for a new product line
2. Identifying requirements or drivers that influence product architecture selection relevant to the design problem
3. Establishing metrics and weightings that provide value to an alternative product architecture
4. Developing a numerical representation of the product architecture space allowing for the comparison of the alternative product architectures
5. Developing evaluation metrics that consider the interactions and couplings present in complex design problems
6. Utilizing or creating new methods that allow systems engineers to make informed decisions on which product architecture to implement

FA²UST ties all of these stages together by sending information gathered in one to inform the ones that follow. When a final decision is made, the systems engineers can trace the information back through each stage to ensure the selection meets the manufacturer's business strategy, customer's needs, and all the derived functional requirements. Also, the systems engineers will gain information about the risks associated with meeting multiple requirement and the complexity of the problem. From this information, the systems engineers can implement appropriate actions to offset these risks.

Figure 33 outlines how information within the framework flows. First, the management and engineers within a firm must identify an industry they wish to penetrate with a new product line. Here, they must develop a business strategy that reflects their strengths, while taking advantage of any opportunities the industry presents. The research conducted during this stage helps inform the engineers when they develop the functional requirements in the following stage. Also, this information helps the engineer determine a way to evaluate alternative architectures in the third stage. In the second stage, the systems engineers take the information acquired from the research conducted in the first stage to develop functional requirements of the new product line. The functional requirements help inform the overall evaluation criteria (OEC) used to evaluate the alternative product architectures in the third stage. Through functional decomposition, the engineer can identify feasible and infeasible combinations of components which therefore influence the feasible and infeasible product architectures that should be examined, providing information for the fourth stage. In the third stage, the systems engineers must derive the OEC from the customer's needs identified in the first stage and the functional requirements derived in the second stage. Also, the industry and internal research conducted in the first stage can help provide qualitative information to help set weightings on three critical metrics: desirability, requirement flexibility, and design complexity. The OEC and weightings

provide the information necessary to evaluate the alternative product architectures in the fifth stage. In the fourth stage, the feasible product architectures identified through the functional decomposition are generated in a numerical space. In the fifth stage, these alternatives are run through appropriate analysis to calculate each product architecture's desirability, requirement flexibility, and design complexity. The weightings are then applied to provide an overall evaluation metric for each alternative. In the sixth stage, the systems engineers analyze the results from the fifth stage and select a product architecture to implement. Finally, the systems engineers must compare the selected product architecture to the manufacturer's business strategy, internal capabilities, and customer's needs to validate it meets all of these criteria.

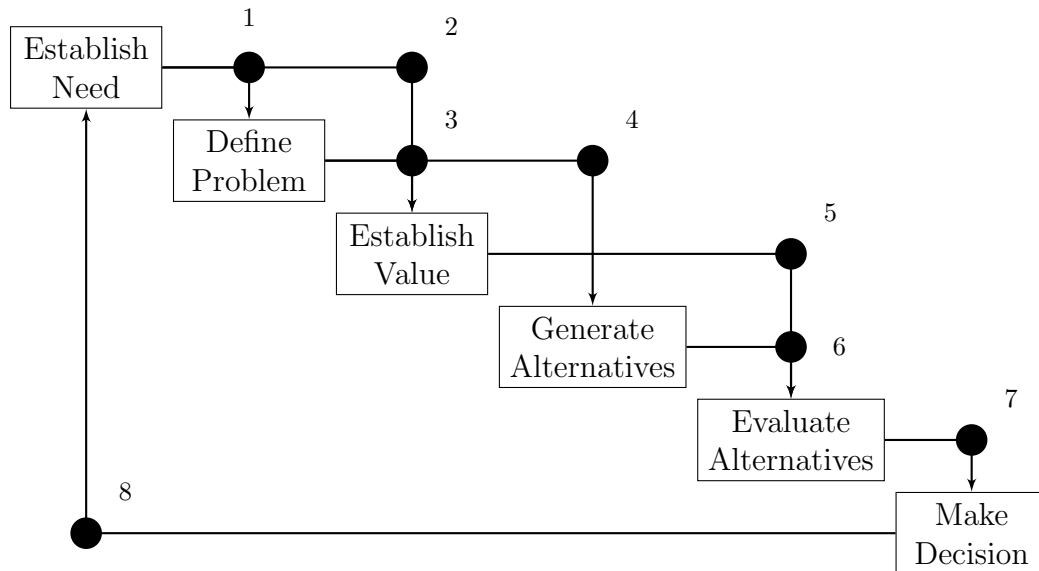
The rest of this chapter goes into detail of each stage, elaborating on the various methods and practices that will result in an appropriately selected product architecture.

3.1 Analyzing Customer Needs and Formulating a Product-Based/Customer-Oriented Business Strategy

The need to develop the new product comes from the request of customers, collaboration among end-users and the manufacturer, or the manufacturer's implementation of a business strategy. The resultant product should either be a solution to a problem, a filler to a gap in the market, or a new integrated solution that disrupts the current market paradigm.

This dissertation introduces a framework that utilizes multiple methods that help the manufacturer determine the desires of the customers and whether the manufacturer can satisfy them, answering the Research Question 1: *Which methods can a manufacturer and its engineers utilize to establish a need for a new product, and develop inputs to facilitate the formation of functional requirements of the product?*

Figure 34 displays the process of formulating a product-based/customer-oriented strategy. Many business academics believe product-oriented strategies are inflexible



Flow of Information:

1. Market, industry, and internal research to inform derivation of functional requirements
2. Market, industry, and internal research on internal processes and customer's desires, enabling the development of an overall evaluation criteria (OEC) and weights for desirability, requirement flexibility, and design complexity
3. Functional requirements enabling the development of an OEC
4. Functional analysis creating options of feasible and infeasible configurations and product architectures
5. OEC and weights for desirability, requirement flexibility, and design complexity allowing each alternative product architecture to be evaluated
6. The alternative product architectures to be evaluated
7. Results from the evaluations allowing for a decision to be made
8. Selected product architecture is validated ensuring it meets the manufacturer's business strategy, internal capabilities, and customer needs

Figure 33: Framework for Product Architecture Analysis of Unmanned Systems and Technologies: FA²UST

to a changing business environment [125] due to the manufacturer’s focus on a specific product, but with the proper analysis and innovative drive these product-based strategies can adapt and prove successful. The proposed framework addresses this issues by incorporating business strategy and systems engineering.

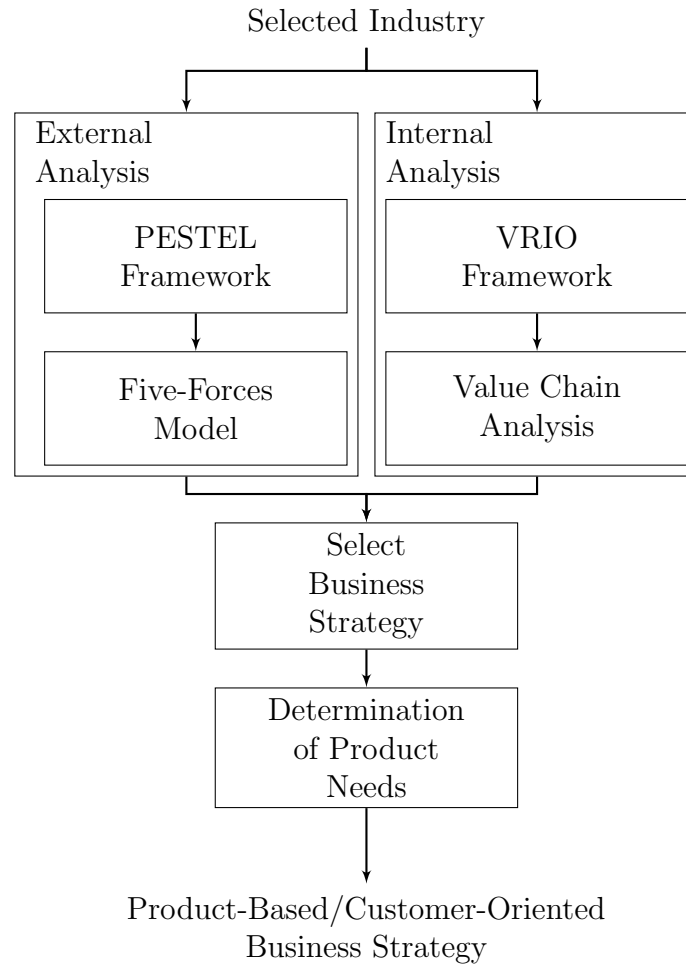


Figure 34: Process to Formulate a Product-Based/Customer-Oriented Business Strategy

First, the manufacturer selects the industry they wish to enter or pursue the development of a product. Then, the manufacturer must analyze that industry and their own internal business.

In the external analysis, a PESTEL Framework (Section 3.1.1) determines the factors that influence the industry, and a Five-Forces Analysis (Section 3.1.2) determines

the average bargaining and competitive strength the manufacturer has in the industry. In the internal analysis, the manufacturer must first revisit its mission, values, and vision, aligning the culture of the company with the entrance into the selected industry or development of a new product. Second, a VRIO Analysis (Section 3.1.3) allows the manufacturer to identify resources and capabilities unique to itself that are useful in the selected industry, and a Value Chain Analysis (Section 3.1.4) determines what resources and capacity are required to produce a product in this industry. At first, this analysis is an initial guess but receives feedback from the configuration and product selection. More frameworks can be utilized depending on the industry and global breadth of operations, but these two frameworks provide sufficient analysis for product development.

From the external and internal analysis, the manufacturer selects a business strategy (Section 3.1.5) that leverages its capabilities, resources, and insights about the industry. The management team must ensure the business strategy, determining how the firm will compete in the industry. Finally, guidelines setting the number to produce, price-points, and capabilities of the product can be configured, establishing the needs of the product (Section 3.1.6) and setting a product-based/customer-oriented business strategy (Section 3.1.7).

3.1.1 PESTEL Analysis

The external environment of an industry consists of the factors that influence it [125]. Analyzing these factors can provide key insights to the manufacturer, including trends, threats, and opportunities. The manufacturer can do little to influence these factors, subjecting the business to the whims of the of each. The factors that compose the PESTEL Framework are:

- *Political*: the impact of actions of governing bodies that impact the market
- *Economic*: the macroeconomic factors influencing global consumer and market

behavior

- *Sociocultural*: society's culture and values that embody the overall view of the industry
- *Technological*: the products or processes that affect the performance or cost of the goods offered in the industry
- *Ecological*: environmental issues that interact with the industry
- *Legal*: laws, mandates, regulations, and court decisions that constrain and direct an industry

These factors impact every industry. Many of the concepts explored using this framework are abstract. The concepts help determine the current and possible future states of the industry. The factors require research to obtain numerical trends or qualitative future options. Managers must consider the insights gained from the framework to determine the best course of action ensuring long-term success. Without knowledge of these factors in a product-oriented can lead to a failed release of a new product causing a drastic loss in invested capital. Also, the research conducted on these factors provides a great amount on information on the external factors that influence the functional requirements. The next stage can use this information provided by the PESTEL analysis to create benchmarks or regulatory constraints.

3.1.2 The Five-Forces Model

The Five-Forces Model derives the profit potential of the industry, the bargaining, and competitive strength of an entity within the industry [125, 118]. Most of the information pertaining to the analysis of the five forces comes from the external factors analyzed in the PESTEL Framework. The analysis provides a starting point for the Five-Forces model. The model consists of the forces:

- Bargaining Power of Buyers
- Bargaining Power of Suppliers
- Threat of New Entrants
- Threat of Substitute Products or Services
- Rivalry among Existing Competitors

The power of the buyers relates to the ability of the business entity's customers to demand lower costs and higher quality of a product, diminishing returns for the firm. Low prices reduce the revenue, and higher quality increases the cost per product for the business. Power of buyers relates to the number of customers, level of standardization, switching costs, and the threat of backward integration. Additionally, the type of clients, their budget structure, and quality control relate to the price sensitivity to the product. If economic factors change the price of the firm's product, then the price is directly affected.

The power of suppliers relates to the company's negotiating strength to purchase the goods to produce the business's product. Suppliers control the price and quality of the input materials or goods directly, impacting the firm's price or performance offered. The power of vendors relates to the number of suppliers, the supplier's dependence on the company's industry, the company's cost of switching suppliers, the supplier's level of differentiation of goods, the lack of substitutes available to the enterprise, and the supplier's ability to forward integrate.

The threat of new entrants is the possible risk competitors will enter the industry. Often, incumbent firms will lower the prices of their goods and spend on marketing or quality to maintain their current customer base, making it hard for the new competitors to gain a foothold in the industry. Companies take advantage of barriers to entry including economies of scale, network effects, customer switching costs, capital

requirements, government policy, and retaliation to reduce the risk of new entrants. At times, it is impossible for the new entrant to be profitable in the industry until the new competitor achieves a certain scale. The scale spreads the fixed cost over a larger number of sales. Network effects occur when the product or service's desirability is related to the number of customers using it, further requiring scale. Switching costs are the costs a customer incurs when they switch suppliers. These expenses can make it hard for new competitors to steal customers. Capital requirements are the price to enter the market, such as investing in infrastructure or machinery. Government regulations or standards create costs for firms. At times, these expenses can be too high for start-ups or new entrants. Finally, when a new competitor enters an industry, the incumbents will retaliate since they have the resources to incur losses until the new firm exits the market.

The threat of substitutes is the risk customers will move to alternative industries that meet the capabilities of current products. These industries often have an attractive performance vs. cost trade-off. Furthermore, the industry is at risk of losing customers if the switching cost is low.

Lastly, rivalry among existing competitors is the battle for market share amongst incumbent firms. When competition is high, firms often find themselves in price-wars, reducing the profitability of the industry. Firms tend to differentiate themselves by offering a "superior" product. The competitive forces relate to the size and number of competitors, industry growth, strategic investments, and exit barriers.

This analysis presents the average forces for the entire industry, and the forces are usually described as strong, moderate, or weak. All five forces relate to the profitability. An industry with strong forces has a low-profit potential. The manufacturer must either strengthen its position or leverage its current favorable position. The analysis allows the management team to devise a business strategy that reflects the

firm's current position and industry dynamics. Furthermore, the engineers and management can use this information to gather a database competitive designs which provide price-points and performance benchmarks. The next stage in the framework can use this information to form benchmarks of functional and economic (or price) requirements.

3.1.3 VRIO Framework

Any business strategy should leverage resources and capabilities unique to the company that provide the enterprise with a sustainable competitive advantage over its competitors [125]. These resources and capabilities make up the company's core competencies and can be tangible or intangible. The VRIO Framework identifies which are the most advantageous by evaluating their value, rareness, cost to imitate, and whether the company is organized to capture the resource or capability's value. Figure 3 displays the VRIO Framework.

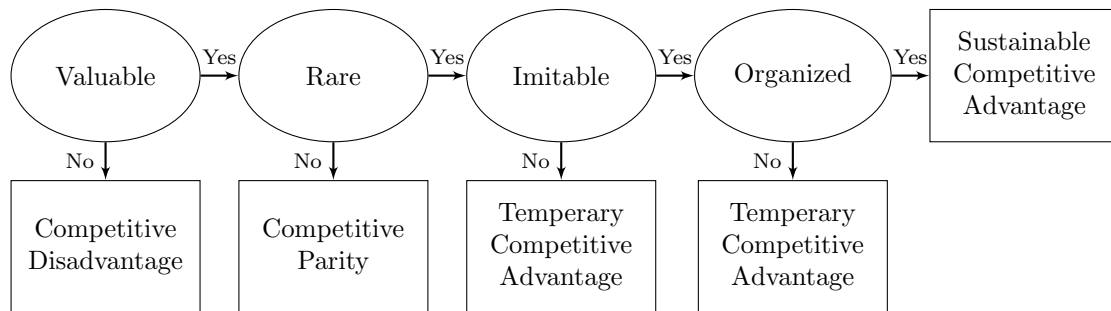


Figure 35: VRIO Framework [125]

The strategy should leverage the resources and capabilities that are valuable, rare, hard to imitate, and have the organization to capture their value. The ones that are valuable, rare, and not necessarily hard to imitate can be used to expand upon but not be the backbone of the strategy. In a product-oriented strategy, the VRIO Framework can determine if the manufacturer has the resources and capabilities to pursue the development of the product. If it does not, the manufacturer should reevaluate the selected industry or look to outsource/enter strategic alliances. The VRIO framework

provides a first check to determine whether the manufacturer can produce product's or compete in this industry. In the final part of the process, the final selected product architecture must fit with the current manufacturer's capabilities. If not, another product architecture must be selected.

3.1.4 Value Chain Analysis

The manufacturer must have the organization and capabilities to produce the proposed product, and the organization and capabilities of the manufacturing entity should relate to the developed product's architecture. Value chain analysis analyzes the disciplines, raw materials, and departments the business requires to develop the new product. Figure 36 displays the chain of activities involved in a businesss value chain.

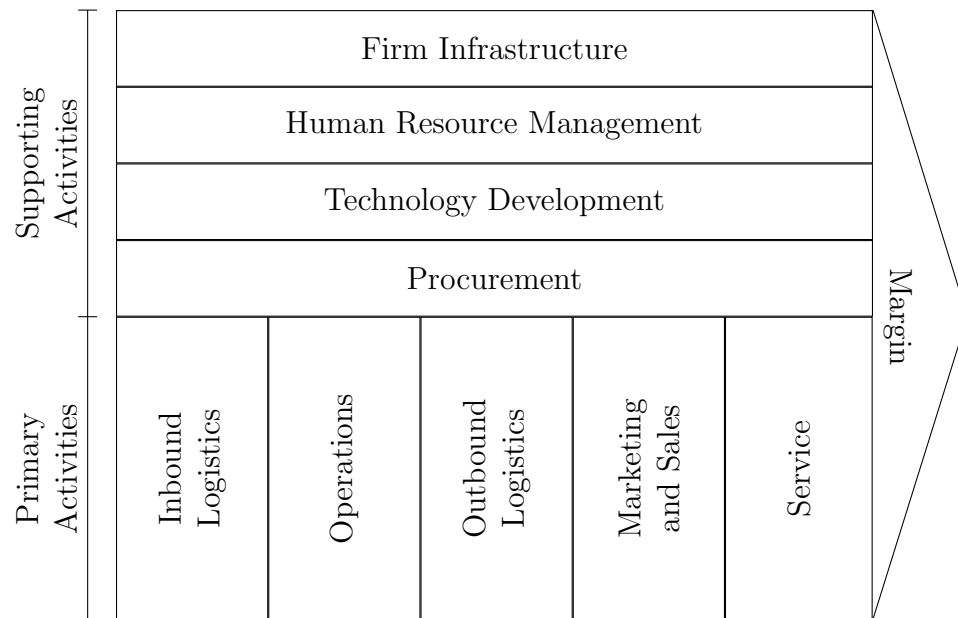


Figure 36: Value Chain Analysis [117]

During the value chain analysis, the company must analyze what value chain activities are required to develop the new product. Traditionally, a business must contain all the elements of the value chain identified in Figure 36 [117]. Containing all processes is considered vertical integration. However, modern IT technologies challenge this assumption. Data and communication between business units are transforming

entities from a vertical to a horizontal organization [55]. Businesses can outsource various value chain processes to external entities, including development, production, and distribution.

There are two primary alternatives to vertical integration. Taper integration utilizes external entities to supply goods, components, or subsystems or distribute the final product. Outsourcing is when another firm conducts an entire value chain activity.

When considering to outsource value chain activities, the hiring company needs to set standards and expectations of the service. Furthermore, the company is losing control of this activity but can focus more resources on the integrated product, reducing the fixed cost associated with the development of the product but increasing the complexity of the design and production of the product. Essentially, the product gets broken down into multiple value chains where transactions and relations occur between internal divisions or external entities. For each transaction relation, the bargaining power between entities must be analyzed to inform the cost and benefits of the tapering or outsourcing. If there is a high probability of design changes (relating to product complexity) and the suppliers power is high, then costs could possibly explode.

The value chains organization relates to the selection of product architecture. The configuration defines what components and subsystems are required, and the interfaces define its nature. For example, manufacturers tend to outsource the design and production of modular components. The initial value chain analysis is an initial guess at the required value chain activities. Throughout the development of the product, it must receive feedback as the configuration is selected. The business must ensure it can conduct or outsource all activities at a reasonable cost. At the end of the FA²UST process, the final selected product architecture must fit within the processes available to the manufacturer. If not, the management must explore outsourcing or

tapering options to complete the value chain activities. If no option is viable, the engineers must select another product architecture.

3.1.5 Selecting Business Strategy

After conducting the external and internal analysis, the manufacturer must choose a business strategy that reflects its internal strengths and the industry and market dynamics. There are two generic types of strategies most used in business [125]. All strategies impact the needs and define the capabilities required of the product. The first is differentiation, which tries to separate the offered product or business model from the rest of the competition. This strategy usually demands higher performance, quality, and technology-level of the product. The second is price-leadership, which attempts to offer the customers a equally valuable product but at a lower cost when compared to the competition. These two strategies depend on the focus of the strategy, which can be either narrow or broad. A narrow focus designs the product around the needs of a distinct group of customers, while a broad focus designs the product around the needs of the market as a whole. Any variation of these three strategies can be implemented. Figure 37 displays the four possible strategic positions. The position and scope relate to the insights gained from the external analysis but reflect the strengths identified in the internal analysis.

The manufacturer must decide how to pursue the market. Where the entity chooses to compete relates to the need for the new product and the functional requirements that define the product. Figure 38 displays the capability market space, which defines dynamics of the market/industry.

In Figure 38, there are two axes: capability and market size. The capability axis relates to the type of product the manufacturer is producing relative to the average of the market. A manufacturer has to deal with the trade-off between price and performance. Unless there is a paradigm shift in the industry, it is either unrealistic

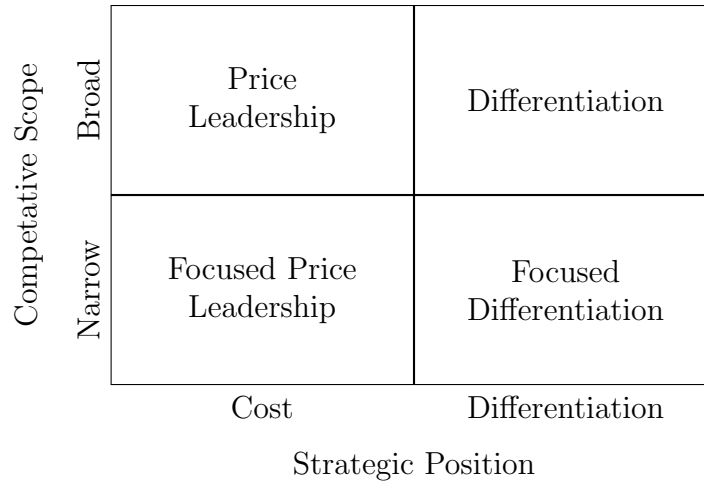


Figure 37: Strategic Position and Competitive Scope [125]

or impossible to produce a low-priced, high-performance product. The market size relates to the number of customers or demand for the new product. Notice, no numbers are presented in this space since the market space is dependent on the industry and time. The market space can be applied to any industry and at any point in time in the industries evolution. Social, economic, legal, ecological, political, and technological factors all impact the market space.

Cutting from the upper left corner to the lower right corner is the Active Product Alley. The Active Product Alley is where most companies compete. As the price and performance increase, fewer customers have access to the capital and resources to buy the product. Successful product-oriented strategies fall somewhere within this alley.

Figure 38 places three examples from the unmanned aerial vehicle industry in the space. The first is the Do-It-Yourself (DIY) drone industry. These drones are low-cost, mostly quad rotor products that can be designed specifically for the purposes the customer intends. This industry sells its product to anyone including amateurs in aviation. The second is the military unmanned aerial vehicle industry. Military UAVs require unmatched operational performance, and only a small number of customers desire them. These customers are primarily government militaries, where the product must fit in the entities overall operations. The third is the civilian operation

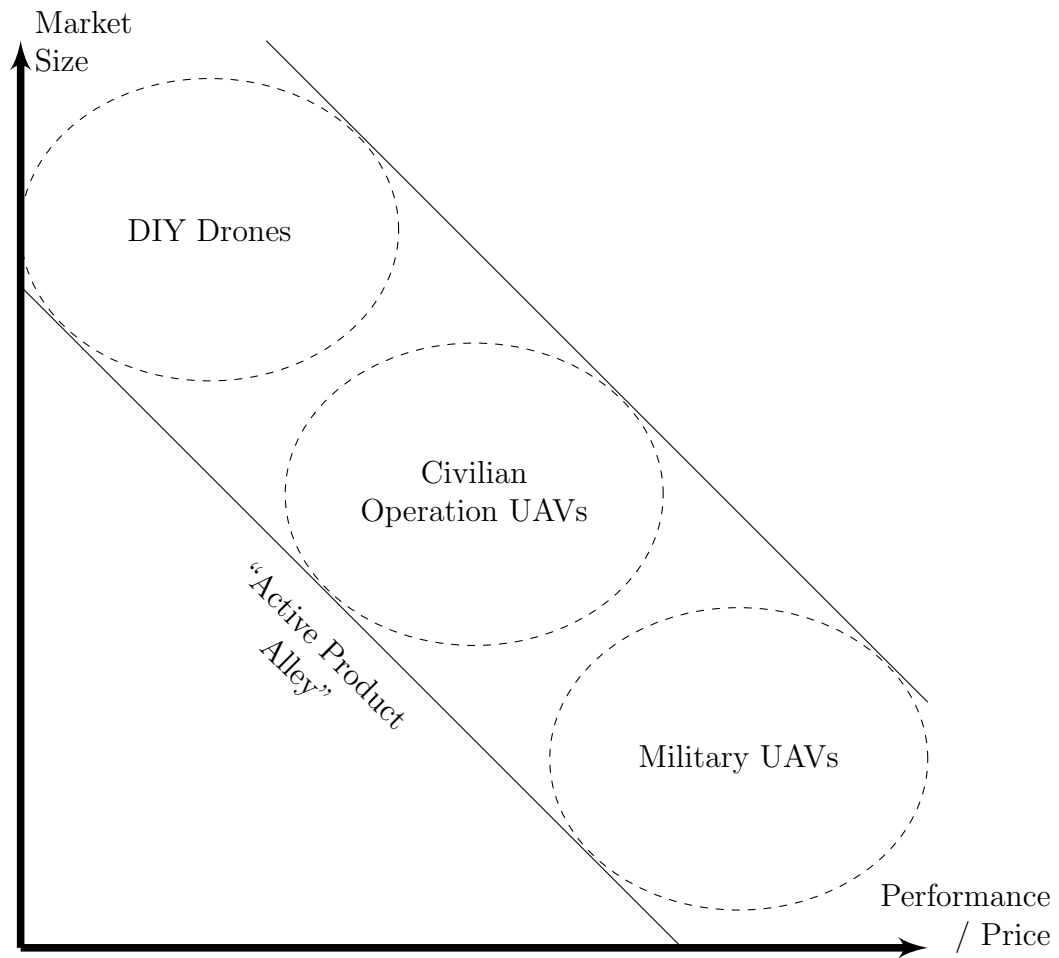


Figure 38: The Capability and Market Size Relational Space

unmanned aerial vehicle industry. These UAVs do not require the level of performance required of military UAVs. However, the industry sells to professional entities that require a level of performance to use in research or corporate-business related activities.

The five forces vary as a company's choice in market changes. As the location moves down the alley, the bargaining power of suppliers and buyers increases as the number of suppliers and buyers who supply and prefer high-performance products decrease. However, the threat of new entrants, threat of substitutes, and competition decreases since the capital requirements and required specialty increase. The opposite can be said as the location moves up the alley. The bargaining power of suppliers and buyers decreases as the number of suppliers and buyers who supply and prefer low-cost products increases. Furthermore, the threat of new entrants, the threat of substitutes, and competition increases due to the decrease in required capital and specialty. The low-cost, large market size is susceptible to disruption since low-cost competitors can emerge a change the dynamics of the industry.

The Market Space Analysis, allows the manufacturers to modify their Five-Forces analysis to reflect the market location the firm decides to compete, expanding the forces' strengths to very weak, weak, moderately weak, moderate, moderately strong, strong, and very strong. Furthermore, it provides information on the demand for the new product, price points, and sets the scope of the strategy. These insights can be used to develop the business strategy and customer needs.

Once the manufacturer selects its strategy, it should revisit its vision, mission, and values [125, 35]. The business's vision outlines the company's aspirations and goals. The vision provides its employees with a sense of purpose. The mission is a statement that describes what the business does. It can define the means it will accomplish its goals outlined in the vision. Managers can reinforce the mission by investing in long-term commitments that strengthen the means the company plans to utilize. Finally,

the values are a statement that describes the company's organizational structure, culture, bedrock principles, and moral compass. It defines the ethical standard the company wants to achieve, providing guidelines on employee behavior.

Vision statements can be customer-oriented or product-oriented. Customer-oriented statements tend to make the enterprise more flexible to a changing environment, and product-oriented statements tend to constrain the approach to the problem [125]. However, if upper management presents the product-oriented vision in a way that aims to solve the customer's problems it can be just as flexible. Instead of purely focusing on how to improve the existing products a hybrid statement will foster innovative thought and new product solutions, resulting in the expansion of the company's capabilities and offerings.

Though these concepts are fuzzy and play on the qualitative side of the engineers' and managers' analysis, defining business's culture can focus intentions, increase the flow of information among business units, and set the tone for the rest of the organization. The vision, mission, and values take queues from the internal analysis conducted earlier, ensuring the strategy is consistent with the analysis.

3.1.6 Determining Product Needs

Once the manufacturer sets its business strategy's position and scope, the manufacturer must determine the needs of the customer that drive the design of the product. Figure 39 displays the places and processes that determine the requirements of a new product. The needs tend to be the capability, performance, or price point requirements.

In Figure 39, the three origins of needs are broken down into three processes. The needs derived by the customer come from two sources. The first option is when the client decides a new product can satisfy the desired capability and will fit into their current operations. The customer determines during this process that their activities

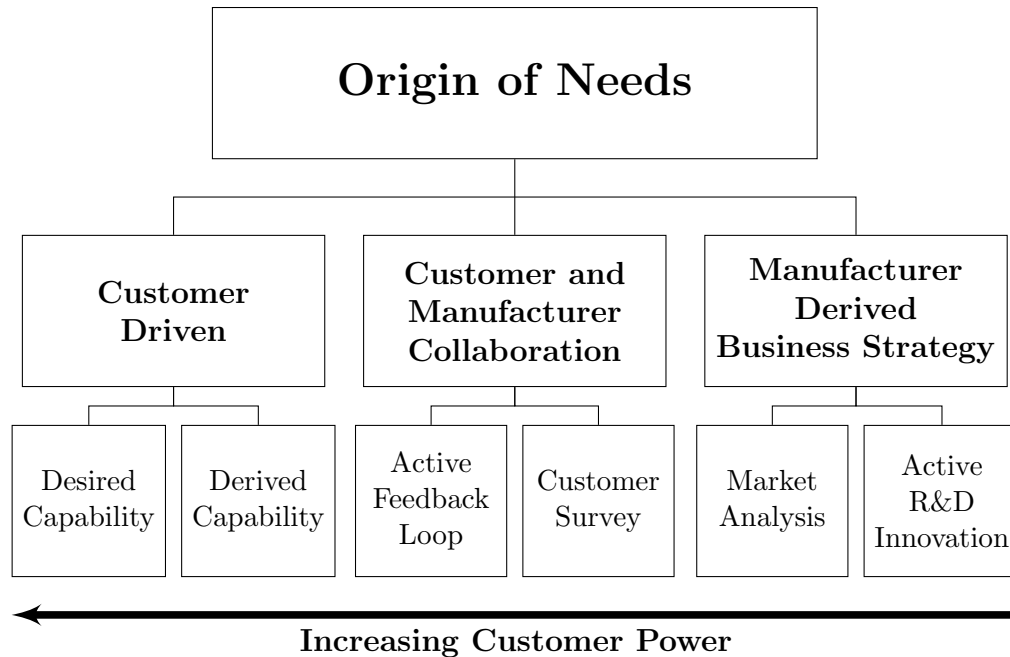


Figure 39: Origins of Needs for a New Product

or operations require no modifications. Instead, a new capability can expand upon them. The second option is when the customer conducts studies on their operations and realize it must implement a new operational strategy which requires an entirely new type of product or capability. These options usually occur when the customer's power is high, allowing the customer to set the needs without much feedback from the manufacturer. The manufacturer must be able to realize when it cannot profitably satisfy these requirements. The manufacturer has the option to back out of the deal.

At times, the customer and manufacturer will collaborate to derive a mutually beneficial product. This process is achieved either through an active feedback loop or a one-way customer survey. The active feedback loop requires the formation of a committee of representatives from the manufacturer and key clients who discuss the trade-offs for both parties achieving the set needs. When the product is not very complex, this approach can be proven very successful. However, with increasing product complexity comes the difficulty in predicting the relation among the product's

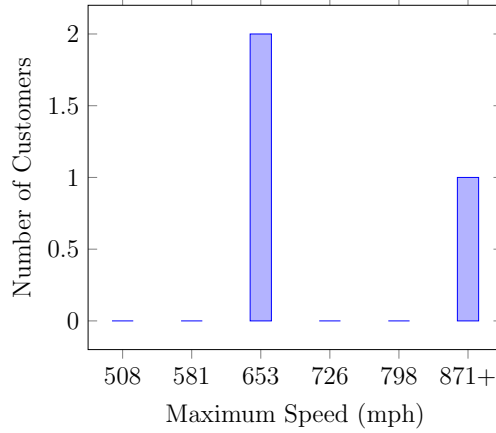


Figure 40: High Customer Power Market

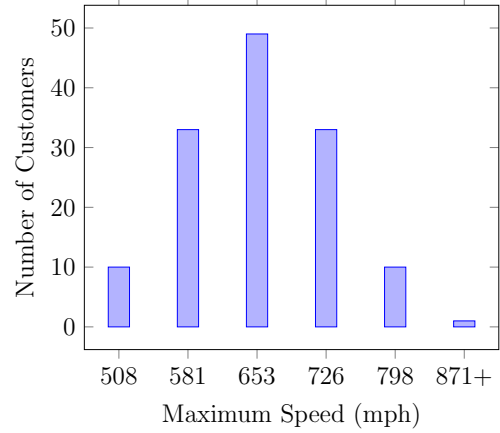


Figure 41: Low Customer Power Market

needs and the design [138]. The second option is a one-way survey, where the manufacturer asks key customers their desires for the product's capabilities to determine the distribution of needs [155]. These options usually occur when the client's power is about equal to the manufacturers. Therefore, the customer's and manufacturer's input is required to determine the needs of the product. The manufacturer in this position has more flexibility to determine its profitability.

When the manufacturer's power is high, or dependence on the customer is low, the manufacturer can determine or shape the market. The manufacturer can analyze the market and choose the most profitable route and set the needs of the product without much customer feedback. The analysis can look at similar products and determine where a lucrative gap is or where the new product could dominate. Another option is to shape the market by discovering a new way to implement and integrate new technologies, requiring innovative thinking and vision for the company. Shaping the market requires the ability to identify needs of the market that the industry has not previously identified. The redefinition of the market or industry can place the manufacturer in an unmatched position [39].

Customer power in this context is extremely dependent on the number of clients. For example, Figures 40 and 41 show two markets with high and low customer power.

In the high-power customer example illustrated in Figure 40, , there are three customers. Two of the customers want a UAV that can achieve a maximum speed of 653-mph and one wants a UAV that can reach a top speed of 871-mph. In this case, the manufacturer is at the whims of the customer. It has three options:

1. Produce a vehicle that can achieve a maximum speed of 653-mph
2. Produce a vehicle that can achieve a maximum speed of 871-mph
3. Produce two vehicles where one can achieve a maximum speed of 653-mph and the other 871-mph

If the manufacturer chooses the first choice, then it will lose the third client, but it should maintain the scale necessary to be profitable. If the manufacturer chooses the second option, then it will lose the other two customers since they probably won't be willing to pay the premium for the increased capability, losing scale and making the probability of profitability lower. If the manufacturer chooses the third option, the manufacturer might implement commonality or reconfigurable characteristics to the architecture to reduce the fixed cost and maintain scale. However, implementation of these features could change the flexibility and complexity of the product.

In the low-power customer example shown in Figure 41, there are 136 customers with a smoother distribution of needs. In this case, the manufacturer has more power and is not dictated by the client. It still has the same options as the former example, but the repercussions are different. If the manufacturer chooses the first choice, then it will lose 44 customers. If the manufacturer chooses the second option, then it will lose 92 customers, but it will be able to charge a premium for the increased performance. If the manufacturer picks the third option is can probably afford two production lines. The manufacturer must expand its margin by implementing commonality and reconfigurable characteristics. However, there is a risk of increased complexity or decreased flexibility by combining the production lines. Therefore, no

matter what strategy the manufacturer chooses it should analyze the product architecture to provide feedback and evidence supporting or challenging the business strategy as shown later in this chapter.

3.1.7 Checking the Formulated Product-Based, Customer-Oriented Business Strategy

As the manufacturer develops its product-based, customer-oriented business strategy it must ask the following questions to ensure the strategy is consistent with its analysis and assumptions:

- Do the business strategy's scope and position reflect the external analysis of the industry?
- Does the manufacturer have or have access to the capabilities to satisfy customer needs sufficiently?
- Which value chain activities should the business internalize, taper, and out-source?
- How does the organization of value chain activities impact the profitability of the product?
- Is pursuit of a business model in this industry a good idea?

Asking these questions checks the process and increases the probability of the strategy's success. Furthermore, as the framework moves into the requirements formulation and the architectural flexibility and complexity analysis, the producer should revisit the strategy and update the assumptions based on the analysis. Updating the assumptions may lead to a modification or at times abandonment of the strategy.

3.1.8 Summary of Analyzing Customer Needs and Formulating a Business Strategy

The results from this stage of the framework are a formulated business strategy that utilizes the development of a new product line to meet the customer's needs. It analyzes external factors relevant to the industry, the dynamics of the industry, internal capabilities and process, and creates abstract concepts of what the customer wants. The information provided by this stage informs the following stages by providing market data and concepts that influence the derivation of functional and economic requirements. Furthermore, it provides strengths and weaknesses of the manufacturer in question which allow the manufacturer to consider the risks present in complex system design. These risks will manifest themselves as requirement flexibility and design complexity. Once the management and engineers have formulated a business strategy and identified customer needs, the engineers must refine the requirements relevant to the new product line's design, leading to the next stage.

3.2 Identification of Product Architecture Selection Drivers and their Impact

After determining the customer needs, the manufacturer must derive requirements that identify the functional requirements and constraints that define the system. These requirements and limitations come from the analysis from the last stage. The research conducted by the last stage identifies internal and external factors that influence the derivation of functional, technical, performance, and economic requirements. Figure 42 shows the steps a system engineer should take to transform the customer needs to functional requirements.

During the formation of the manufacturer's product-based/customer-oriented business strategy, the industry analysis provides the expected demand, market performance and price benchmarks, and overall business goal of the new product. Also,

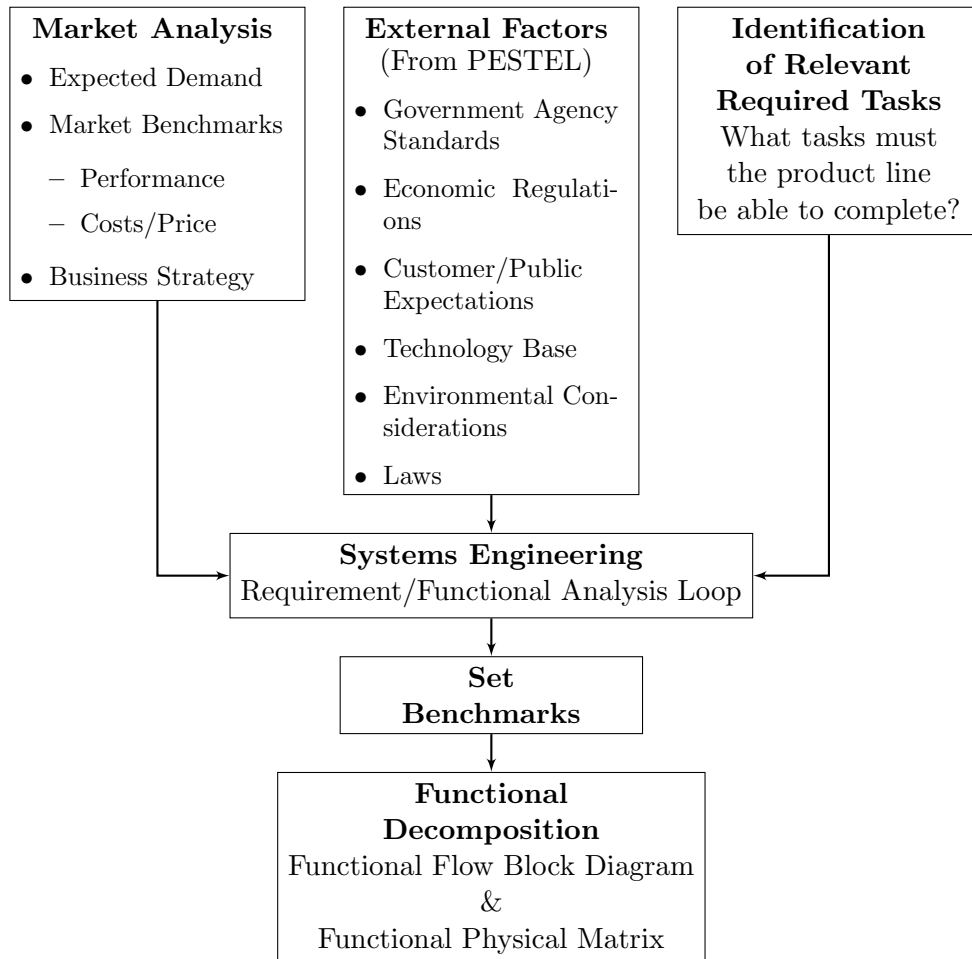


Figure 42: Process to Transform Customer Needs to Functional Requirements

the PESTEL analysis identifies external factors relevant to the industry. The research provides regulations and other requirements the engineers must consider in the development of the new product line. The customer's needs identified in the last stage provide abstract ideas of what tasks the product line must be able to complete. The systems engineers must create concrete definitions of these tasks. The system engineers and management must combine these three elements into quantifiable requirements with benchmarks set based on market data or expert analysis. The requirement and functional analysis found in Section 2.1 can provide the methods and processes that assist the systems engineers in the process. The requirements formed from the industry and internal analysis can be broken down into internal and external factors, shown in Figure 43.

From these three tasks, the engineers can set the product's technical, performance, and economic benchmarks. Regulating governmental organizations often provide standardized missions or tasks, but engineers can adjust these as they see fit. These benchmarks must be consistent with the business strategy and the standards defined by external industry actors. Finally, the engineers must functionally break-down all of the required tasks of the product using a Functional Flow Block Diagram (Section 2.1.3.1). Breaking down the tasks into individual functions allows the engineers to pair components or subsystems required to complete each function. Using a Functional-Physical Matrix (Section 2.1.3.2) which uses a list of possible components that can be integrated into the product line and the individual functions that compose the required mission or task, the functions can be matched with feasible components or component pairings. The feasible configurations can be fed into the fourth stage where the framework determines which feasible product architectures to consider.

Out of the requirements formed in this stage in the framework, the ones that drive the product architecture selection and a means to capture their impact must be identified. The following section explores methods to answer Research Question

External Factors	Internal Factors
<ul style="list-style-type: none"> • <i>Customer Expectations</i> (Capability, Economic, Sociocultural, Technological, and Ecological) • <i>Project Specific Constraints:</i> <ul style="list-style-type: none"> – Industry Specifications and Baselines (Technological) – Costs (Economic) • <i>Enterprise Constraints:</i> <ul style="list-style-type: none"> – Standards and Guidelines (Legal) – Policies and Procedures (Political and Legal) • <i>External Constraints</i> (Political, Technological, and Legal) • <i>Performance Requirements</i> (Capability) • <i>Modes of Operation</i> (Capability) • <i>Technical Performance Measures</i> (Capability) 	<ul style="list-style-type: none"> • <i>Project Specific Constraints:</i> <ul style="list-style-type: none"> – Team Assignments and Structure – Control Mechanisms • <i>Enterprise Constraints:</i> <ul style="list-style-type: none"> – Management Decisions – General Enterprise Specifications – Domain Technologies – Physical/Financial/Human Resources • <i>Operational Scenarios</i> • <i>Measures of Effectiveness and Suitability</i> • <i>System Boundaries</i> • <i>Interfaces</i> • <i>Utilization Environments</i> • <i>Life Cycle Process Concepts</i> • <i>Functional Requirements</i> • <i>Physical Characteristics</i> • <i>Human Factors</i>

Figure 43: Breakdown of Requirements

2: *What are the typical design drivers that lead system architects towards different architecture implementation strategies?* A couple of industries will be reviewed to provide a starting point when answering this question. The research will review the evolution of each industry and by analyzing reasons what forced designs to implement different product architectures over time.

3.2.1 Investigation of Past Industries

This dissertation selected four industries to analyze as case studies. The four industries investigated are the automobile, multi-role helicopter, US Navy carrier fighter, and unmanned aerial vehicle or system. These industries provide examples of a mass market, a robust performance, a high performance, and an emerging industry respectively. Each case study looks at key designs throughout its history analyzing their levels of commonality and reconfigurability, their product architecture’s qualitative label, and the reasons that drove engineers to implement the product architecture. At the conclusion of each case study, a list of drivers relevant to the industry.

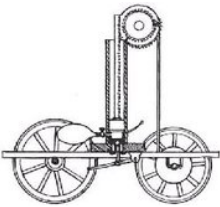






3.2.1.1 Automobile Industry: A Mass Market Industry

The automobile industry has matured over the last one hundred years. The automobile has become a commonly utilized system throughout the world. Its purpose varies with geographic market and owner. Since the auto industry’s market is large, it provides an excellent case study. Table 35 displays the auto industry’s evolution.

The first combustion engine automobile emerged in 1807. It was an innovative invention that was powered by hydrogen and oxygen combustion. Modern automobiles still possess the primary piston and spark-plug subsystem [33]. It was a “fixed” product architecture, implementing no commonality and reconfigurability. Since it was a prototype, the first combustion engine was more of a proof-of-concept that showed propulsion by this method was possible.

In 1897, Winton became the first major automobile manufacturer in the United

Table 35: Automobile Industry Architecture Evolution

Year	Model	Picture	Architecture
1807	First combustion automobile		Fixed
1897	Winton Motor Carriage Company Car		Fixed
1908	Ford Model T		Modular
1933	Pierce-Arrow Silver Arrow		Fixed
1928	Ford Model A		Modular
1960s	GM Pontiac Tempest, Buick Skylark, & Oldsmobile F-85		Modular
2000	Dower Ridek		Modular

States. At this point, automobiles were an emerging industry [17]. Designs were custom built for the wealthy. As a result, each vehicle was hand made with specific

components built for each order. Winton’s cars incorporated some online reconfigurability to allow the operator to control the vehicle in motion. This included a steering wheel for lateral control, a throttle for engine speed, and transmission for wheel speed [105]. However, they are still considered a “fixed” product architectures since they were a single product without any family members to share parts and without any major reconfigurable components.

In 1908, Henry Ford revolutionized the auto industry with the Model T. Ford implemented rigorous standardization which allowed Ford to produce many more vehicles at a lower cost. The low cost opened up the automobile market allowing people from all economic backgrounds to purchase a car. Ford offered multiple body styles that were all compatible with the same standard chassis and engine [134]. Its incorporation of mass production and standardization laid the groundwork for the evolution of the industry [38]. The automobile industry had appeal to the mass market. By implementing standardization in the design, Ford met the price point required by the larger market. Again, the Model T incorporated online reconfigurability so the operator could control the vehicle in motion. However, it qualitatively can be considered a “modular” product architecture since it shared common components across all versions with offline reconfigurable bodies.

During the roaring 1920s, the market changed. The market evolved to into multiple market segments. Each segment had individual expectations of capabilities and performance. Therefore, the first trucks, sedans, and high performance vehicles began to emerge. The demand for customization, or “modification[s] made to something to suit a particular individual or task [10],” caused manufacturers to focus on one market segment or implement product architectures that could satisfy all segments while managing costs. Due to the growth and separation of wealth, luxury designs became increasingly profitable in all industries. The Pierce-Arrow Motor Car Company capitalized on this trend by developing custom built vehicles to for the very wealthy [137].

Much like the Winton's cars in the early 1900s, parts were made specifically for each design. Though very profitable at first, the Pierce-Arrow Motor Car Company was unable to adapt during the Great Depression in the 1930s and went out of business [100]. Its custom-made, "fixed" architecture proved inflexible to the change in market demand and price.

During the Great Depression, Ford again dominated the market into the 1940s. The Ford Model A used mass production and implemented the module-based design. The module-based design allowed the product line to share common parts reducing costs and offline reconfigurable components to provide various type of cars specific to each market segment custom needs [106]. The combination of common and offline reconfigurable components made the Model A cheaper and easy to customize, making cars available to the mass market and upgradeable for those who could afford it [38]. Therefore, the "modular" product architecture allowed Ford to survive the economic depression while companies like the Pierce-Arrow Motor Car Company to go out of business. The standardization implemented by Ford in the early 1900s was the catalyst for the emergence of the "modular" product architecture. It kept costs low for Ford while still satisfying the diverse needs of the market. Therefore, the "modular" product architecture was flexible to the economic shocks that occurred during the Great Depression. Hereby ending manufacturers' use of "fixed" product architectures in the industry.

By the 1970s, most automobile manufacturers had adopted mass production and module-based implementation. However, during this time period Japanese manufacturers emerged with superior production practices, and the price of oil saw huge fluctuations as conflict in the Middle East caused shortages[144]. Furthermore, customers had come to expect a new model every year causing obsolescence, or "the process of becoming no longer in use or the condition of being nearly no longer in use [9]," to shorten to a few years. Thus, manufacturers saw their profits squeezed as

all customers demanded lower acquisition and operating costs associated with each product while still demanding high levels of customization. Most manufacturers produced multiple models which shared common chassis to reduce production cost [132]. However, the increased commonality among models caused some of the lower performing vehicles to cannibalize higher performing vehicles, resulting in the reduction of performance and popularity of the higher performing models [107, 27, 78]. Some automobile manufactures even incorporated smaller engines usually found in the economy (low-priced) cars into the higher performing cars. When customers found out, these manufacturers lost business. By reducing costs by using common components, the cars became similar missing the customization demanded by the customers.

Now in the post-2000 era, all manufacturers produce automobiles which contain some elements modular design where multiple designs share a common chassis and components can be swapped to modify the performance. The customer demands now look similar to the 1970s. Though the price of oil has stabilized it has been fairly high [23]. Therefore, customers still demand lower acquisition and operating costs associated with each product while still demanding high levels of customization. These demands are proven by the large number of models offered by numerous manufacturers, and the constant pressure to reduce fuel consumption in the industry [109, 79, 41]. Thus, automobile manufacturers use module-based product architectures to meet all of these requirements while using technology to reduce fuel consumption. The common use of modular product architectures in the industry has even seen the industry come up with interesting concepts. The Modek, an entirely modular design, was invented in 2000 [53]. The proof-of-concept shows the industry continues to increase the implementation of modular product architectures .

Market Size, customization, low-cost, and quick obsolescence influenced the automobile industry's implementation of product architectures in their designs. Eventually, the entire industry adopted the use of module-based or modular product architectures because they help the products meet all of the requirements demanded by the customers. When Pierce-Arrow Motor Car Company decided to pursue highly specialized cars for the wealthy in the 1920s, they were unable to adapt during the Great Depression as demand for luxury decreased. The Pierce-Arrow Motor Car Company could not cope with changes in the market resulting in the company's demise. Also, the industry always produces new models each year, causing previous ones to become obsolete quickly. Standardization and modular design provide a means for companies to stay market competitive by developing easily upgradeable sub-components.

From this case study, the list below identifies the predominant architecture selection drivers for the automobile industry.

Automobile Architecture Selection Drivers

1. Production Quantity
2. Diversity of Product Roles (Customization)
3. Acquisition Cost
4. Operation Cost
5. Obsolescence

The automobile industry investigation leads to Observation 1:

Observation 1:

Mass market industries that tend to focus on mass production, customization, and low costs leading designers to favor module-based product architectures.

3.2.1.2 Multi-Role Helicopter Industry: A Robust Performance Industry

The multi-role helicopter emerged after World War II. The systems were designed to provide medical evacuation, surveillance, personnel transportation, troop support, communications, and electronic warfare. This case study analyzes the two most dominant platform in the industry: the Bell UH-1 and Sikorsky UH-60. Table 36 displays the multi-role helicopter industry evolution.

The Bell UH-1 Iroquois (Huey) was the US Army’s first operational turbine engine helicopter. Bell developed the vehicle as a “medevac”-transport helicopter for the US Army in 1955 [135]. The UH-1 product line consisted of two models: the 204 and 205. Throughout the Vietnam War, Bell developed multiple variants of the models to meet the diverse needs the US Army encountered during the conflict. Bell delivered the first variant, the UH-1A (Model 204A), in 1959. During its deployment, the US Army outfitted the variant with machine guns and rocket launcher. After experiencing combat, the US Army requested Bell create a new variant that had greater passenger and cargo capacity. Thus, Bell developed the UH-1B (Model 204B) in 1961 which could accommodate seven passengers; three stretchers, two sitting casualties, and a medical attendant; or 3,000 pounds of cargo [135]. Soon thereafter, the US Army requested more variants to meet specific roles in their military strategy. Thus, Bell developed:

- The UH-1C (Model 204C) as a gunship in 1965
- The UH-1D (Model 205) in 1963 extending the UH-1’s passenger limit to fourteen

Table 36: Multi-Role Helicopter Industry Architecture Evolution

Year	Model	Picture	Architecture
1959	Bell UH-1A Iroquois (Huey)		Fixed
1961	Bell UH-1B		Product Family
1963	Bell UH-1D		Product Family
1979	Sikorsky UH-60A Black Hawk		Fixed
1984	Sikorsky SH-60B Seahawk		Product Family
1990	Sikorsky HH-60J Jayhawk		Product Family
1991	Sikorsky HH-60G Pave Hawk		Product Family

- The UH-1E (Model 204E) in 1964 as an US Marines assault support helicopter
- The UH-1F (Model 204F) in 1964 as an US Air Force utility helicopter
- The UH-1H (Model 205H) in 1967 as a subsystem and engine upgraded version of the UH-1D specifically designed to handle bad weather
- The HH-1K (Model 204K) in 1970 as an US Navy search and rescue helicopter
- Bell developed other variants of the UH-1 including the UH-1L, UH-1M, UH-1N, UH-1P, UH-1V, UH-1X, and UH-1Y which were primarily subsystems and engine upgrades of the previously developed variants as technologies in these subcomponents improved

Bell was able to produce the variants on two production lines, one for the Model 204 and another for the Model 205. The production coordination was possible due to the common air frame components amongst models with various engines and electronics paired with the specific role assigned to the vehicle. Though the UH-1 variants possessed some online reconfigurability to allow the operators to control the vehicle during flight, the combined vehicles are considered a “product family” architecture. These characteristics made the helicopter “affordable” and widely used throughout the Vietnam conflict [135]. The multiple roles, missions, and subsystem capabilities drove the evolution of Bell’s UH-1 product family. As the branches of the US military identified new roles or suppliers produced better subsystems for the helicopter, Bell met these needs with a new variant of an existing design. Technology evolution is the incorporation of new concepts and theories into existing systems to obtain a greater, practical solution [77]. Over the years, specifically the electronics and propulsion domains saw rapid technological innovation creating greater efficiency and power. Bell’s utilization of a product family architecture maintained costs by achieving a large economies of scale and offering a portfolio of diversified capabilities (customized

roles). Economies of scale “is the cost advantage that arises with increased output of a product [136].” It essentially diffuses the fixed costs of production over larger produced quantities. However, the US Military turned to a new manufacturer to produce multi-role helicopters following the Vietnam War.

The Sikorsky UH-60A Black Hawk replaced the Bell UH-1 Iroquois (Huey) in 1979. Sikorsky originally designed it to carry eleven combat troops plus three crew members. Customers can use it for utility, air assault, medivac, command and control, and reconnaissance missions [96]. Due to the Black Hawk’s success, Sikorsky developed the SH-60B Seahawk to replace the Kaman SH-2 Seasprite in 1984, a modified version of the Black Hawk specifically for the US Navy. Sikorsky designed it for missions including anti-submarine warfare, search and rescue, drug interdiction, anti-ship warfare, cargo lift, and special operations. [148]. Designers shortened the Seahawk, placed the rear wheel forward, and transformed the cargo area allowing it to carry mission dependent subsystems. Later, Sikorsky developed the HH-60J (later upgraded to the MH-60T in 2007) Jayhawk to replace the HH-3F Pelican in 1990. It was a modified version of the Seahawk specifically designed for the US Coast Guard. It incorporated robust and rugged design for maritime environments, and its design missions focused on enforcing maritime laws and search and rescue [98]. Also, Sikorsky designed the HH-60G Pave Hawk in 1991 for electronic warfare equipment as an upgraded version of the Black Hawk. Designers modified it, allowing the integration of electronic warfare equipment. All new electronics, avionics, and subsystems gave the Pave Hawk the capability to conduct special electronic warfare operations as well as the original Black Hawk missions [95]. Sikorsky implemented a similar production strategy as Bell, creating many more variants of the UH-60 that either satisfied a US Military defined role or incorporated more technologically advanced engines or subsystems. Though the UH-60 variants possessed some online reconfigurability to allow the operators to control the vehicle during flight, the combined vehicles are

considered a “product family” architecture. The product families use of common air frame components allowed Sikorsky to produce the variants on the same production line maintaining costs while offering a number of products that satisfy diverse roles.

Throughout the world, multiple countries, militaries, and entities use Bell’s and Sikorsky’s helicopters. Both companies still produce modified versions of the Iroquois and Black Hawk. As a result, the Iroquois and Black Hawk product families dominated the multi-role helicopter industry during their height of production because of the reduced manufacturing costs associated with product-family architectures. This example shows achieving scale in a highly niche market is essential to profitability over time and in some cases leads to market dominance or company survival.

From this case study, the list below identifies the predominant architecture selection drivers for the multi-role helicopter industry.

Multi-Role Helicopter Architecture Selection Drivers

1. Number and Diversity of Product Roles
2. Subsystem Capabilities and Technological Evolution
3. Production Quantity

The multi-role helicopter industry investigation leads to Observation 2:

Observation 2:

Industries that demand rugged and multi-mission vehicles where the missions and subsystem capabilities are diverse, lead designers to favor product-family architectures.

3.2.1.3 US Carrier Fighter Industry: A High Performance Industry

Since the end of World War I, militaries have realized the effectiveness of air superiority over land and sea. As a result, countries have demanded the highest performing aircraft, known as fighters, to fight battles for air superiority. To project air superiority abroad, the United States launches many of these fighters from naval carriers, extending the country's power and reach. These designs require higher performance, structural strength, and ruggedness. The high performance, small-niche industry provides an alternative case study. Table 37 displays the US carrier fighter industry evolution.

The Grumman F-6F Hellcat was a rugged, well rounded, carrier-based fighter designed to integrate well into carrier operations. Grumman specifically intended it to counter the Japanese Zero in 1943. The Hellcat was capable of greater speed and power, compared to the Zero. These capabilities proved superior to the Zero as long as the Hellcat pilots avoided lower speed dog fights [67]. Grumman developed the Hellcat during World War II. It was a single product produced on one product line specifically designed to outperform the Japanese Zero. Though it has online reconfigurable components to allow the operator to control the vehicle during flight, it was primarily considered a “fixed” product architecture. However, the Hellcat was soon replaced by higher technological jet fighters after World War II.

The Vought F-7U Cutlass added speed to the carrier-based aircraft due to its jet engine, swept wings, and tailless design. It also included hydraulic landing gear and flight controls. The addition of immature technologies, even though state-of-the-art, made the vehicle unreliable. The hydraulics leaked and the engine never produced the thrust expected, but it provided the first step introducing jet fighters into the US fleet [12]. Vought developed the Cutlass to replace the Hellcat in 1951. It incorporated modern electronics and subsystems and was the first jet-powered carrier-based fighter. Though it has online reconfigurable components to allow the operator

Table 37: US Carrier Fighter Industry Architecture Evolution

Year	Model	Picture	Architecture
1943	Grumman F-6F Hellcat		Fixed
1951	Vought F-7U Cutlass		Fixed
1957	Vought F-8 Crusader		Online Re-Configurable
1960	McDonnell Douglas F-4 Phantom II		Fixed
1972	Grumman F-14 Tomcat		Online Reconfigurable → Product Family
1978	McDonnell Douglas F-18 Hornet		Product Family
Future	Lockheed Martin F-35 Lightning II		Modular, Online Reconfigurable, & Product Family

to control the vehicle during flight, it was primarily considered a “fixed” product architecture. However, it was eventually replaced as the Cold War demanded more capabilities from the fleet and better technologies emerged.

The Vought F-8 Crusader added supersonic capabilities to the carrier-based aircraft. It had an online reconfigurable wing which could change its pitch to improve carrier takeoff capabilities [99]. Vought developed the Crusader in 1957 to incorporate subsystems and weapon systems required by the US Navy as a response to the growing arms race during the Cold War. The increased payload requirements increase the total weight of the vehicle. Therefore, Vought added a online reconfigurable wing that could increase the pitch of the wing during takeoff allowing the vehicle to have a small takeoff length requirement. The strong interaction between the required payload weight and short takeoff field length requirements required Vought to implement online reconfigurability. It also had online reconfigurable components to allow the operator to control the vehicle during flight, but its online reconfigurable wing made it considered as a “online reconfigurable” product architecture. However, the online reconfigurable wing increased the complexity of the fighter during the development, production, and operations of the vehicle, causing the aircraft to be much more expensive than its predecessors.

The McDonnell Douglas F-4 Phantom II, developed in 1960, provided the US Navy a long range fighter/bomber. Its multi-mission capability was a response to reduce the overall acquisition costs for the US Navy. McDonnell Douglas replaced traditional gunnery (machine guns) with a stockpile of radar guided missiles and state-of-the-art electronics and subsystems. These changes hoped to provide the US military with a “fighter of the future” [72]. The new subsystems and weapon systems were lighter than their predecessors, causing the older fighters to become obsolete. Also, engine technology improved over time providing the industry with more powerful engines. Thus, McDonnell Douglas could design a fighter without online reconfigurability.

Though it has online reconfigurable components to allow the operator to control the vehicle during flight, it was primarily considered a “fixed” product architecture. However, the Cold War created the need to expand the capabilities of the carrier-based fighters into the supersonic regime.

The Grumman F-14 Tomcat replaced the aging F-4 Phantom II in 1972. It was the first fighter to achieve supersonic speeds which caused the vehicle to be capable of overcoming huge amounts of drag. The F-14’s online reconfigurable architecture allowed it to be carrier-launched and still achieve supersonic speeds. The aircraft was designed to complete air-to-air, precision air strike, and naval air defense missions. It included the integration of advanced electronics to support the new weapon systems and radar interceptor capabilities paired with the specific mission the vehicle would conduct. The electronics, engines and structures were upgraded twice, producing a product family with the two F-14B and F-14D variants [13]. Before Grumman developed the two variants, the Tomcat was considered an “online reconfigurable” product architecture. As new technologies became available, Grumman updated the electronics, engines, and structures creating the two variants and a “online reconfigurable” and “product family” architecture emerged. The new variants were more efficient, powerful, and capable. However, the online reconfigurable wing made the aircraft heavy and complex. As the technologies incorporated in the electronics, engines, and structures continued to improve, the Tomcat’s product architecture began to become obsolete.

The improvement in technologies allowed McDonnell Douglas to develop the F-18 Hornet as a fighter for the US Navy and Marines in 1978. McDonnell Douglas designed it to conduct air-to-air and air-to-ground missions. Users could adapt it to perform photo-reconnaissance and electronic countermeasure missions. The technologies integrated into the design were carbon-fiber wings and digital fly-by-wire controls. These technologies drastically reduced the weight, removing the need for the online

reconfigurable wing implemented in the F-14. At first, McDonnell Douglas developed the A and B variants together. Variant A had one seat and variant B had two, primarily for training purposes. Variants (C, D, E, F and G) of the Hornet were introduced later on in production as upgraded versions. The variants included reducing the radar signature, improving engines, and increasing maneuverability, range, and payload capacity [24]. When McDonnell Douglas developed the E and F variants, the manufacturer increased the size of the vehicle to accommodate larger engines and more sophisticated avionics. The variants were produced on a common product line and shared many of the same airframe components, implementing a “product family” architecture. Implementing the product architecture allowed McDonnell Douglas to manage costs through economies of scale.

With ever-expanding requirements for modern fighters and long-time operation of the F-18, the US Military decided it was time to upgrade the fleet in 2006. Lockheed Martin is concurrently developing the Lightning II for the US Navy, Air Force and Marines. It is a combination of modular, online reconfigurable, and product-family architectures. They share module-based components. The non-carrier variants possess the ability to reconfigure its engine orientation for vertical takeoff and landing (VTOL) or short takeoff and vertical landing (STOVL). The F-35 combines advanced stealth with fighter speed and agility, handles fully fused sensor information, and is capable of network-enabled operations and advanced sustainment [83]. The highly interacting requirements have caused production overruns and missed deadlines [138]. The F-35 provides a concept of interest for this dissertation. The F-35 provides an example where designers implemented reconfigurable and commonality characteristics into the architecture to increase the performance and decrease the manufacturing cost. However, the common components reduce more than the reconfigurable increase the performance, and the reconfigurable components increase more than the common components reduce the complexity and cost.

The US carrier fighter industry shows companies tend to choose architectures that provide the highest performance sometimes with an increase in cost, and architecture selection depends heavily on technology. Therefore, it is crucial to understand the impact of technology on the architecture. Considered technologies are electronics, structures, propulsion, materials, and manufacturing. Many of the fighters mentioned replaced existing ones. The designer must understand when a vehicle and its technology might become obsolete and how the architecture will be affected. As a result, most of these high performing vehicles started as fixed architectures and were later upgraded, causing product families to emerge.

From this case study, the list below identifies the predominant architecture selection drivers for the US carrier fighter industry.

US Carrier Fighter Architecture Selection Drivers

1. Performance
2. Technology Evolution
3. Obsolescence
4. Requirement Evolution

The US carrier fighter industry investigation lead to Observations 3 and 4:

Observation 3:

Industries driven by technologies, obsolescence, and rapidly evolving and high-performance requirements lead designers to favor fixed product architectures which can be upgraded, forming product families.

Observation 4:

High-performance industries lead designers to consider online reconfigurable product architectures.

3.2.1.4 Unmanned Aerial Vehicles or Systems Industry: An Emerging Industry

The unmanned aerial vehicle or system (UAV or UAS) industry is emerging, motivated by the advancements in electronics and computer science. UAVs have existed since the American Civil War but have not become universal until the last quarter century. The absence of humans in the cockpit increases its mission capacity and reduces its cost. UAVs have historically conducted for military operations, but as regulations shift, civil uses continue to grow. Therefore, the UAV industry provides a case that has not fully matured. Table 38 displays the UAV industry evolution [56].

During the American Civil War, the northern forces developed the first unmanned aerial vehicle to drop bombs with a timing mechanism. Perley's Aerial Bomber, developed in 1863, was a balloon that contained a basket full of explosives. The concept was mostly ineffective. As a result, developing UAVs dissipated until technology improved [112]. The vehicle was an experimental concept with only one made to try to break the deadlock of the American Civil War. It was nearly impossible to control and was about as "fixed" of a product architecture as one can find.

Due to improved electronics and aeronautical engineering, Nazi Germany developed the Fieseler Fi 103 (V-1) in 1944 during World War II. It was a jet-propelled

Table 38: Industry Architecture Evolution

Year	Model	Picture	Architecture
1863	Perley's Aerial Bomber		Fixed
1944	Fieseler Fi 103 (V-1)		Fixed
1955	Teledyne-Ryan Aeronautical AQM-34 Firebee		Fixed
1982	IAI Scout		Fixed → Modular
1995	General Atomics RQ-1 / MQ-1 Predator		Fixed → Product Family
1998	Northrup Grumman RQ-4 Global Hawk		Fixed → Product Family
2003	AeroVironment RQ-11B Raven		Modular
Future	UCLASS	N/A	Modular

cruise missile that could hit a target from long distances. These vehicles used a primitive jet engine and analog electronics to navigate the bomb to a general target area. The cruise missile was not very accurate but provided the ground work for many of the UAVs built from 1940 to 1980 [101]. Though it has online reconfigurable components to allow the operator to control the vehicle during flight, it was primarily considered a “fixed” product architecture. Its low accuracy was due to its infantile analog controls. Not until digital and improved computational technologies emerged did unmanned systems become practical.

In 1955, the US Air Force started using the Teledyne-Ryan Aeronautical AQM-34 Firebee as a target and stealth reconnaissance drone. For reconnaissance missions, the US Air Force used a modified version of the vehicle with stealth material for the skin. Improved electronics and propulsion made the vehicle much more controllable [140]. Though it has online reconfigurable components to allow the operator to control the vehicle during flight and its modular subsystem package, it was primarily considered a “fixed” product architecture, partly because the term modular was not widely used in the industry. As a result of the Firebee’s success, the uses of UAVs expanded.

In 1979, the Israeli military introduced the IAI Scout designed to provide intelligence and reconnaissance. The development of a small, reliable UAV required all departments of IAI to work together to develop custom subsystems for the vehicle. It consisted of composite materials, microwave signal communications, and new stabilized electro-optical sensors. In the 1990s and 2000s, IAI developed variants of the Scout where modular subsystems could allow the vehicle to jam enemy communications and conduct electronic warfare [70]. Though it has online reconfigurable components to allow the operator to control the vehicle during flight and its modular subsystem package, it was primarily considered a “fixed” product architecture at first. However, as the mission portfolio for the vehicle expanded, IAI developed modular subsystem packages which operators could swap out depending on the mission the

vehicle was to conduct. Thus, the product architecture evolved into a “modular” product architecture.

The General Atomics developed the RQ-1/MQ-1 Predator as a new generation of reconnaissance and surveillance UAV in 1995. It also spurred the modern growth of the UAV industry and a realization of UAVs’ potential. It satisfied the US requirements of a medium-altitude, long-endurance (MALE) UAV. In 2001 the RQ-1 was armed with AGM-114 Hellfire missiles creating the MQ-1. The MQ-1 can conduct armed reconnaissance and interdiction missions. The RQ-1 and MQ-1 functioned similarly to the IAI but were more efficient and less detectable [14]. Though it has online reconfigurable components to allow the operator to control the vehicle during flight and its modular subsystem package, it was primarily considered a “fixed” product architecture at first. Similar to the IAI Scout, as the mission portfolio of the vehicle expanded, General Atomics developed a second variant that could operate with weapon systems. Thus, the product architecture evolved into a “product family” architecture.

The Northrup Grumman RQ-4 Global Hawk is a high-altitude long-endurance (HALE), reconnaissance, and surveillance UAV. Its primary purpose is to provide high-resolution images in all weather and over large geographical areas, day or night. Northrup Grumman built two variants of the Global Hawk in 2010 and 2011: the MQ-4C Triton for Naval missions and the RQ-4E Euro Hawk for NATO related missions [97]. Since the program started in 1998, cost overruns grew until 2010, due to ineffectiveness, unreliability, and changing performance requirements [30]. In 2006, the Global Hawk unit cost was 25% over its baseline estimate, and the US Congress almost canceled the program [11]. However, by 2013 from higher usage, costs were reduced 50%, making it a more viable design [128]. The trend in cost also relates to the maturation of the high tech UAV design process. Though it has online reconfigurable components to allow the operator to control the vehicle during

flight, it was primarily considered a “fixed” product architecture at first. Similar to the IAI Scout and General Atomics Predator, as the mission portfolio of the vehicle expanded, General Atomics developed multiple variants that had varying subsystems and capabilities. Thus, the product architecture evolved into a “product family” architecture.

The AeroVironment RQ-11B Raven is a lightweight reconnaissance and surveillance UAV with a modular frontal subsystem compartment. It is small enough to be man-portable allowing users with varying levels of experience to use it throughout the world. It also is rugged and reliable making it easy to use in tough environments. The detachable front section provides a compartment for various subsystems [147]. The Raven is one of the first modular designs in the industry, but with DARPA’s continuing request for new modular architectures, providing the expectation of more to come [119]. Though it has online reconfigurable components to allow the operator to control the vehicle during flight and its modular subsystem package, it is primarily considered a “modular” product architecture due to its modular frontal subsystem compartment.

The UCLASS is a UAV that is currently under development. It will be able to launch and operate autonomously from an aircraft carrier and conduct long-endurance ISR&T (intelligence, surveillance, reconnaissance, and targeting) and precision strike missions. It will inherit stealth and other technologies from the F-22 and F-35 programs, allowing it to integrate in the battle space of the future. Finally, acknowledging DARPA’s request for modular architectures, the UCLASS is being developed to have interchangeable components [102]. Since the program is still under development, it is unclear what type of product architecture the competing manufacturers will implement.

The unmanned aerial vehicle is an emerging industry. It combines elements of the automobile, multi-role helicopter, and US carrier fighter industries. It not only

has the potential appeal towards a mass market, but also it must be reliable no matter what the task or environment, and some designs require high performance. As the industry matures, the architecture selection trends will change as seen other industries. A manufacturer must choose a flexible and competitive architecture that will be able to sustain itself in the future markets. Therefore, the future architectures this industry will implement is uncertain.

Due to the immature nature of the UAV industry; this case study cannot identify any drivers or observation. Therefore, this dissertation attempts to determine the drivers of architecture selection for the UAV industry.

3.2.1.5 Observations from Previous Industries

Combining all of the drivers identified from the case studies, a list of architecture selection drivers emerges. The items can be broken down into four categories: design, market, life cycle duration, and technologies.

Architecture Selection Drivers

1. Design Requirements
 - (a) Performance
 - Speed
 - Range
 - (b) Mission Diversity
2. Market
 - (a) Size
 - (b) Acquisition and Operating Costs
3. Life Cycle Duration
 - (a) Requirement Evolution
 - (b) Obsolescence
4. Technologies
 - (a) Impact
 - (b) Evolution

Identification of drivers provides a system architect with what factors impact architecture selection. The ones identified above provide a starting point determining what drives architecture selection. However, the drivers have been identified qualitatively but need to be verified quantitatively. The combination of these drivers leads to Assertion 1:

Assertion 1:

The design, market, life cycle, and technological requirements drive the product architecture selection process.

Understanding how these drivers impact architecture selection is important and challenging. Even so a designer or manufacturer, without fully understanding the complicated nature of product architecture selection, tends to make these decisions heuristically. Furthermore, it is complicated by the use of qualitative terms to describe the product architecture. As shown in the case studies, many of the products offered in all of the industries contain some common and reconfigurable elements, yet there are given vague qualitative labels. Therefore, the product architecture must be converted from a qualitative label to a numerical representation. This dissertation formulates a data-driven, quantitative framework to assist the designers in selecting the most appropriate architecture.

3.2.2 Observing the Design Drivers' Impact on the Product Architecture Selection

It is important that the system architect understand the impact and characteristics of each driver on the architecture selection. Section 3.2.1 identified possible drivers. Again, Table 39 displays the drivers identified and breaks them down into four categories: design, market, life cycle duration, and technologies.

Currently, there is no way to understand a drivers' impact on the product architecture selection numerically and visually. Understanding the drivers' impact will aid the architect in strategic road mapping and in performing trade-offs between alternative architectures. A test case presented in this section captured the impact of the identified drivers, using historical data. The test case provides a means to find answers to Research Question 2.a: *Is there some effect from how drivers are structured early in the design process that tends to favor one implementation strategy over*

Table 39: Product Architecture Selection Drivers Identified in Section 3.2.1

Design	Market	Life Cycle Duration	Technologies
1. Performance	1. Size	1. Requirement Evolution	1. Impact
2. Mission Diversity	2. Customization	2. Obsolescence	2. Evolution
	3. Manufacturing Costs		

another?

Multiple industries and markets were analyzed to get the full picture of architecture decision making. These industries are mostly mature with various driver values and choice in architectures. Industries considered were the automobile, truck, race car, commercial aircraft, bombers, multi-role helicopters, fighters, and unmanned aerial vehicles industries.

For each industry, the test case placed a relevance or magnitude ranking ranging from one to five on a factor: five being high, three medium, and one low. Furthermore, it identified the predominant architecture(s) for each industry. If the industry implemented a type of architecture, an index received a value of one. Table 40 displays the results.

The automobile industry has a large market size. Its customers demand customization and quickly want new products making the original products obsolete. Thus, the industry implements modular design techniques.

The truck industry is like the automobile industry. The only difference is the demand for greater payload capability.

The F1 race car industry is a niche, high-performance market. The industry requires the cars to be fast, highly customizable, and able to perform on multiple

Table 40: Test Case and Driver Impact Framework

Industry	Design				Market		
	Range	Speed	Payload	Mission Diversity	Market Size	Cust.	Manuf. Cost
Automobile	2	2	3	1	5	5	1
Truck	3	1	5	1	5	5	1
Race Car	1	5	1	4	1	5	1
Commercial Aircraft	5	4	5	2	3	2	5
Bombers	5	4	5	1	1	1	5
Multi-role Helicopters	3	3	4	5	3	4	4
Fighters	1	5	2	3	1	1	5
Commercial Helicopters	3	3	3	3	2	3	4

Req.	Life Cycle			Technology			Product Arch. Charact.		
	Evolution	Obsolescence	Impact	Evolution	Fixed	Reconfigurable	Common		
Automobile	5	5	2	5	0	1	1		
Truck	3	5	1	5	0	1	1		
F1 Race Car	4	5	5	5	0	1	0		
Commercial Aircraft	2	3	4	1	1	0	1		
Bombers	1	1	5	3	1	0	0		
Multi-role Helicopters	3	3	4	3	1	0	1		
Fighters	2	1	5	2	1	1	0		
Commercial Helicopters	1	3	4	1	0	0	1		

tracks. The rapid introduction of new technologies makes the cars obsolete quickly. Thus, engineers implement modular design techniques.

The commercial aircraft industry essentially produces buses that can fly. Therefore, the industry is driven by the payload, manufacturing costs, and technological impact on the design, though the evolution of technologies is slow. Engineers in this industry implement scale-based designs.

The military bomber industry develops aircraft that must carry large payloads over a long distance. Stealth technologies are usually implemented to increase the odds the aircraft can achieve its objective. The manufacturing costs to produce these specialized aircraft are extremely high. Therefore, engineers design each aircraft for each specific mission, implementing fixed designs.

The multi-role helicopter industry requires systems that can take off vertically and carry nominal payloads over nominal distances. The need for robust multi-mission performance and high customization drive the industry. Therefore, engineers design product families.

The military fighter industry requires high-performance vehicles to complete combat objectives. These systems must have a high power to thrust ratio to achieve high speeds. Furthermore, new technologies are continually implemented to increase the vehicles' performance. Thus, engineers implement online reconfigurable characteristics to contribute to achieving these high-performance requirements.

Finally, the commercial helicopter industry is like the multi-role helicopter industry. However, the market does not demand the same customizable and multi-mission requirements. Therefore, engineers design product families.

Regressions were fit to the data to relate the product architecture indices and driver values. Due to the limited number of cases and the assumed threshold behavior of the drivers, a neural network was fit to the data. After fitting neural networks to the architecture driver data, the test case ran a Monte Carlo which varied all the drivers'

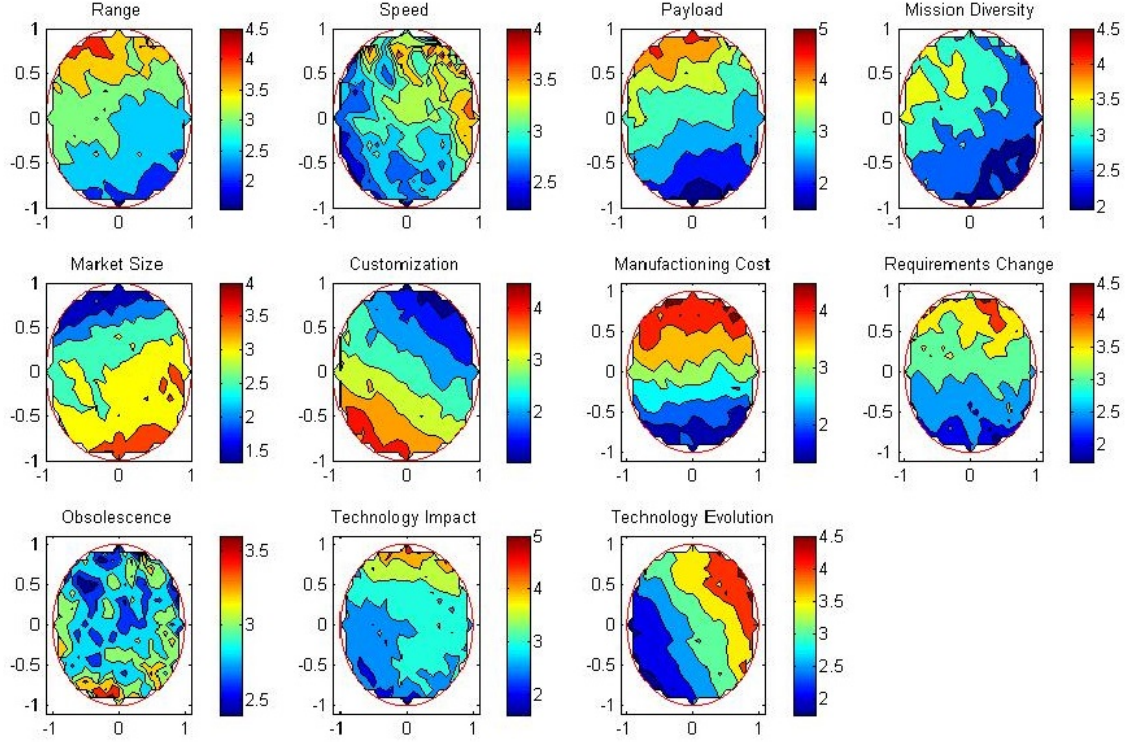


Figure 44: Architecture Driver Magnitude in the Space

values from one to five. The data was transformed from its three-dimensional form to two dimensions. The results are observations and conclusions of the drivers impacts on the product architecture selection. The transformation is given by Equation 31.

Figure 44 displays the average driver value for each region in the architecture space, allowing the determination of where certain drivers dominate. In Figure 44, the circular space represents the various product architectures which include fixed at the top, reconfigurable on the left, and commonality on the right. As a location in the space moves towards the edge of the circle, that product architecture becomes more predominant.

The results from the test case show how the various drivers influence the implemented product architecture. The following observations were drawn from the results:

- As the **performance** required (range, speed, and payload) of the product increases, the implemented architecture tends to be more fixed.
- A product that must conduct **multiple missions** tends to favor more reconfigurable product architectures.
- When the **market size** is greater, engineers have historically implemented commonality and reconfigurability characteristics.
- When the market demands highly **customizable** products, engineers implement reconfigurable characteristics.
- Historically, as the **manufacturing costs** increase, engineers tend to develop a fixed product.
- Industries whose **requirements change** frequently tend to build one-time fixed products.
- If implemented **technologies'** impact and evolution are more relevant in the industry, then engineers create fixed/common product architectures.

3.2.3 Observing the Product Architecture Selection's Sensitivity to Design Drivers

After determining the impact of the drivers on the product architecture selection, it is important to understand how changing drivers influence the process, answering Research Question 2.b: *How do we capture the impacts of changing drivers on the product architecture?*. Time-dependent differential equations can be utilized to understand the impact of changing drivers over time. In these equations, the intrinsic variable's (\vec{x}) rate of change to a extrinsic variables (\vec{t}) can be represented by the function: $\dot{\vec{x}} = f(\vec{x}, \vec{t})$. For this problem, \vec{x} is the product architecture, and \vec{t} is a vector of drivers. To analyze the path architecture will take based on initial conditions, a phase portrait is useful.

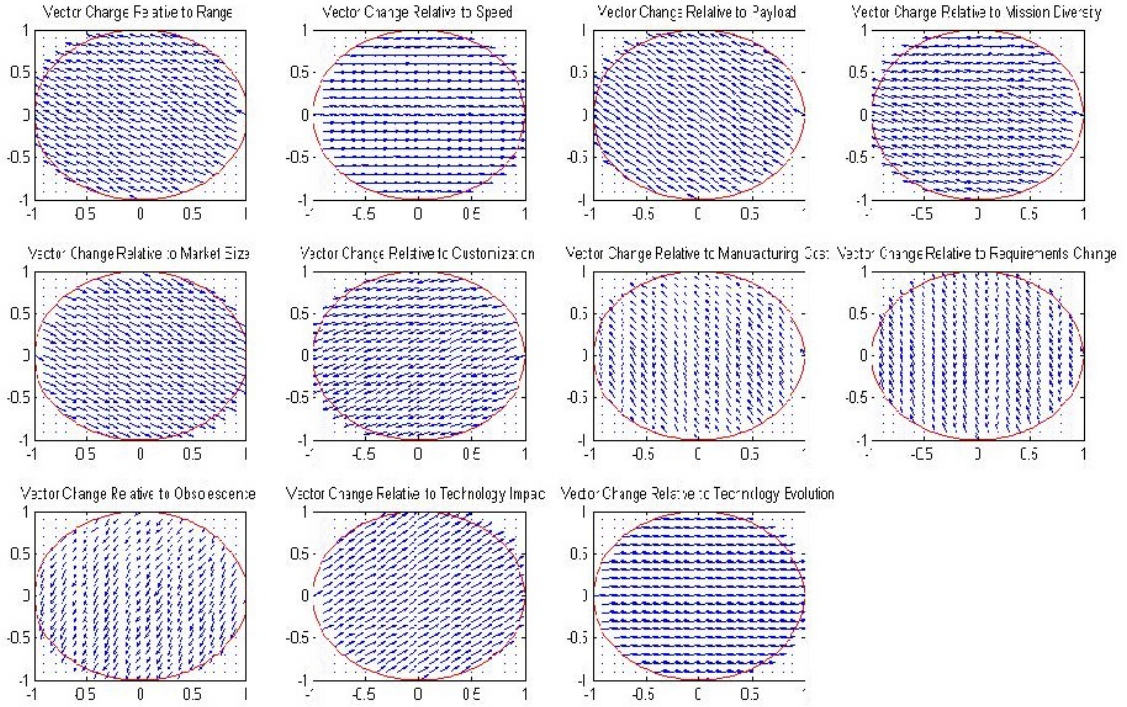


Figure 45: Driver Impact Mappings

Since drivers of an industry change throughout time, it is important to understand their changes' influence on the selected product architecture. Partial derivatives of the neural nets in the previous subsection provide the equations to create the phase portraits. This dissertation refers to these phase portraits as Driver Impact Mappings and are discussed in Section 3.6.2. Figure 45 displays the results from the test case.

The phase portraits show how the favorable product architecture reacts to a changing driver. These flows trace the derivative of the magnitudes found in the last subsection, usually pushing the product architecture towards one edge of the circle.

3.2.4 Conclusions from Test Case Determining Drivers Influence on the Product Architecture

The case study shows there are relationships among the drivers and the selected product architecture. Furthermore, the relations can identify the regions where certain drivers dominate and how their change influences the decision-making process.

System engineers hope to choose a product architecture that is resilient over time, satisfying the customer and ensuring profits for the manufacturer.

The results from the test case show how an engineer can choose favorable or a product architecture resilient to changing drivers. The favorable architecture is the one that satisfies each of the drivers. For a flexible product architecture, the choice is not as obvious. The flow diagrams show the relationships between intrinsic and extrinsic variables. Theoretically, when these flows are combined, nodes emerge where the flow is attracted or repelled, showing the flexible architectures.

3.2.5 Formation of Hypothesis 1

The case studies given in this section suggests certain requirements drive the product architecture selection more than others. An experiment must be formulated to justify the selection of these drivers. Furthermore, the experiment should be able to identify any other factors or requirements that drive the architecture selection.

These considerations and observations from this section help form Hypothesis 1:

Hypothesis 1:

If the product architecture selection drivers identified truly drive the decision, then they must significantly impact the process's results by influencing the levels of commonality and reconfigurability in the product line.

3.3 *Establishing a Valuable Product Architecture*

System engineers prefer product architectures that facilitate a product's ability to satisfy manufacturer goals and customer demands. These are concrete metrics defined by performance or cost analysis. Furthermore, the product architectures that reduce complexity and increase the flexibility of the product, relating to cost reduction and ensure long-term profitability. Flexibility and complexity are abstract concepts that

are not as easily defined. Therefore, a trade-off exists between flexibility and desirability (concrete and abstract concepts). This section analyzes how to capture this trade-off answering the third research question: *How do we develop a quantitative means to determine whether an architecture is “good” or valuable?*

The three metrics can be defined as follows:

- Requirement satisfaction or desirability relates to a products ability to achieve customer demands.
- Flexibility “implies an ability of the design to be changed to track requirement changes [127, 126],” suggesting if a product is flexible and an operator uses it for a purpose other than originally designed for, then it will still perform well.
- Complexity is the combination of the number of domains, functions, or disciplines; the level of interdependency among the domains, functions, or disciplines; the novelty of project; and level constraints stringency [21].

The engineer must identify which of the three concepts are most important or have the most relevant. Therefore, the engineer must ask the following questions to determine how important each of the metric. These questions attempt to add the more qualitative elements of product architecture selection. Each metric is assigned questions that relate to the “-ility” in a sense that can not be answered through numerical analysis. Therefore, here are some examples of what qualitative questions one might ask to assign weightings. All answers can be the following responses: high, moderate, or low, representing a score from 1 to 3.

- Desirability

1. *How much power do the customers have?* - This question relates to how important it is to achieve the maximum desirability. If the customers have

more power, the less the manufacturer can deviate from the most desirable design of the customers.

2. *How many requirement thresholds must the product achieve?* - This question relates to how much influence the customers have on the product and suggests the designer must satisfy several constraints. As a result, by achieving maximum desirability, the manufacturer is meeting the most requirements as possible.

- Flexibility

1. *How short is a product's traditional life span in the industry?* - Life span responds to the sensitivity of market to changes in requirements over time. If the product's life span is short, then the product is expected to be replaced quickly, meaning flexibility is not as pressing as say desirability and complexity.
2. *What is the cost to develop and produce a new product?* - If the cost is high to produce a new product, the product must be flexible to changing requirements. Flexibility will relate to the product's ability to stay relevant over time and will not need to be replaced. If the cost is low, the product can be easily replaced and flexibility is not a major concern.

- Complexity

1. *What is the manufacturer's novelty producing a product?* - The producer's experience in the industry and producing the product relates the company's ability to predict and mitigate mistakes during the design, development, and production of the product. Mistakes are a symptom of complexity and if an experienced team works well together, then there will be less mistakes, meaning complexity is not a pressing concern.

2. *How many domains are associating with developing a new product?* - The number of domains or disciplines involved in designing a product relate to the number of experts and trades between groups. Even if coupling among disciplines is not apparent, scale increases the chances of mistakes occurring, increasing the need to consider complexity.

Each metric can get a score ranging from two to six, and the total can vary from six to eighteen. Taking each score and dividing it by the total provides the relative importance of each metric. Though engineers should consider all of the metrics when evaluating the architecture, the engineers should realize which of the three metrics are more important or relevant.

The use of qualitative weightings can be called into question due to the lack of numerical support in asserting these questions impact the choice in product architecture selection. Therefore, a hypothesis can be formed to test the relevance of the weightings:

Hypothesis 2:

If the qualitative weightings have significant influence on the decision of which product architecture to implement, then the most favorable product architecture should be vastly different at various weightings.

Cases can be conducted where extreme cases of weightings can be compared against each other. These sensitivities studies will determine the weightings impact on the product architecture and whether the magnitude of change makes sense.

The framework can apply this method to any “-ity,” such as maintainability and availability. These “-ities” look at the supply chain dynamics of the product value chain. Therefore, the questions can relate to distances goods must travel to reach customers and quality/customer support of the product.

3.4 Identification of Methods to Facilitate Generating Alternative Product Architectures

Section 1.3 defines the three primary product architectures: fixed, reconfigurable, and product family. Furthermore, it identifies three that are a combination of two main architectures: online reconfigurable, modular, and scale-based. Though definitions clarify the problem, they do not provide a good way to quantify the product architecture space. The following section answers the second Research Question 4: *What methods can be used to aid in the generation of alternative product architectures?*

A product architecture is dominated by three characteristics of components: common, online, and offline reconfigurable. The fixed, reconfigurable, and product family definitions can be modified to component characteristic dimensions using this representation of the space. A fixed product architecture lacks any of these features. Figure 46 shows the transition from qualitative definitions to quantitative axes. The planes defined by two of the axes are reconfigurable, modular, and customizable products. However, products are not limited to the planes or axes. A product architecture can possess all characteristics of varying degree. The purpose of creating the three-dimensional space provides a means to understand architectural features of a certain product, allowing the engineers to perform trade-offs between alternatives in architecture selection problems.

When transforming the space from three to two dimensions, the product architecture space can be represented by Figure 47, providing a simplified means to visualize the space. Product architectures that primarily share two characteristics are online reconfigurable, modular, or customizable product architectures, represented by the planes formed by two axes in Figure 46.

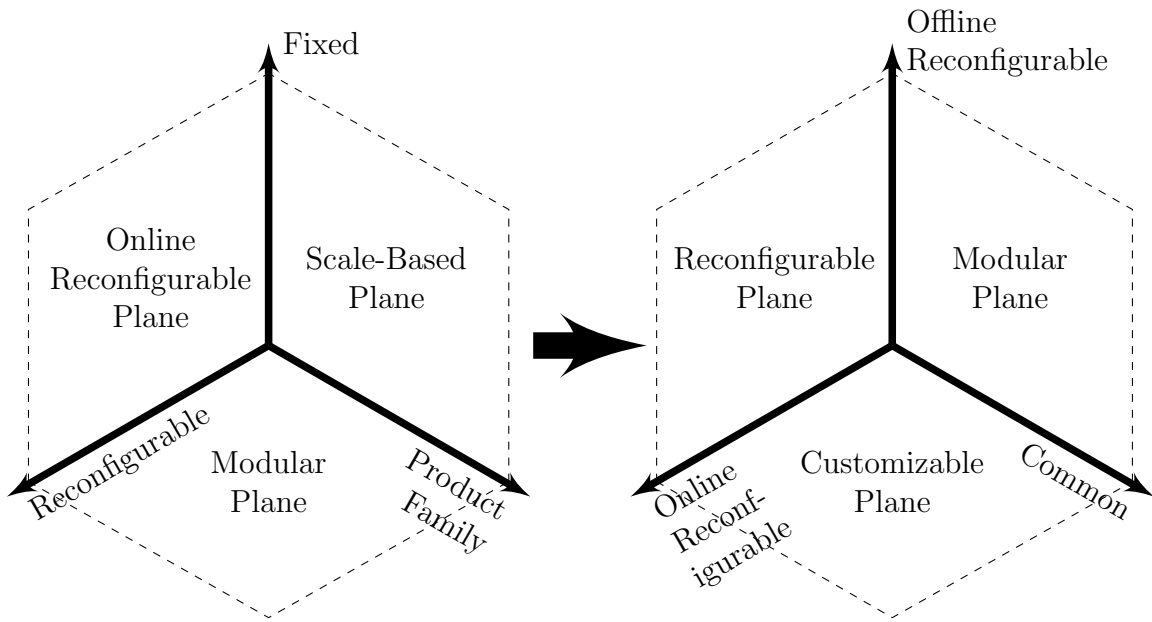


Figure 46: Transformation of Product Architecture Space

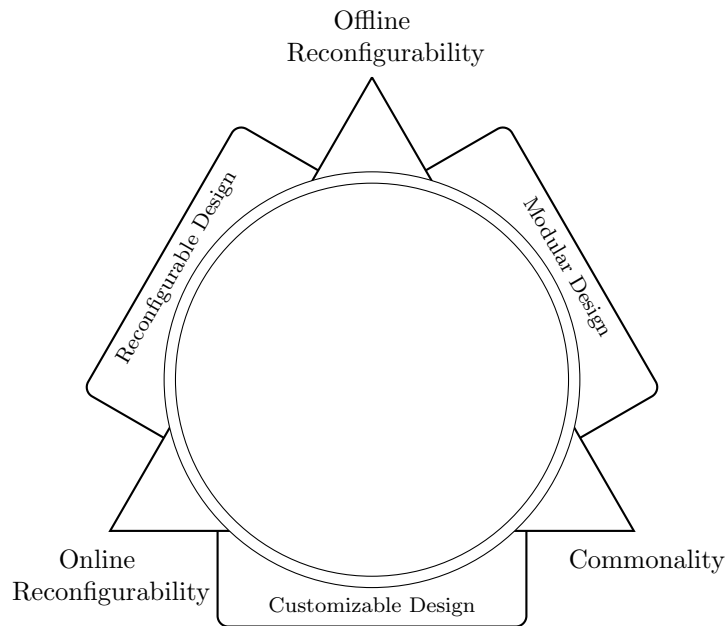


Figure 47: Two Dimensional Representation of Quantitative Architecture Space

3.4.1 Commonality Index

After defining the axes, the axes must be defined by indexes. The commonality index (CI) defined in Section 2.3.1 can represent the common axis [142]. The metric ranges from zero to one. One implies the product line shares all components and zero suggests there is no commonality in the product line. Equation 25 displays the index formulation. Where, J_p is the total number of designs that belong to the product architecture, $\#Comp_{Tot_j}$ is the total number of elements in a single family member, $\#Comp_{Tot_{Unique}}$ is the total number of **unique** components in the product architecture, and j represents a family member index.

$$CI = \begin{cases} 0 & \text{if } \sum_{j=1}^{J_p} (\#Comp_{Tot_j}) = \max(\#Comp_{Tot_j}) \\ 1 - \frac{\#Comp_{Unique} - \max(\#Comp_{Tot_j})}{\sum_{j=1}^{J_p} (\#Comp_{Tot_j}) - \max(\#Comp_{Tot_j})} & \text{otherwise} \end{cases} \quad (25)$$

3.4.2 Online Reconfigurability Index

For the reconfigurable axes, a literature review found an index that represents the architecture's ability to change or swap components [62, 64]. The possible reconfigurable indices reviewed did not meet the criteria outlined in Section 2.3.1. However, offline and online reconfigurability indices can still be derived from the research of previously developed indices. For example, the Percent Commonality Index (%C) includes the interfaces or connections to calculate commonality. The interfaces are critical in reconfigurability. Online reconfigurable components require the interface between the vehicle and the component to be manipulable, adding a degree of freedom to the product's controls and dynamics. Therefore, to calculate the online reconfigurability index, the number of degrees of freedom added by a reconfigurable interface are divided by the total number of interfaces between components in the product line.

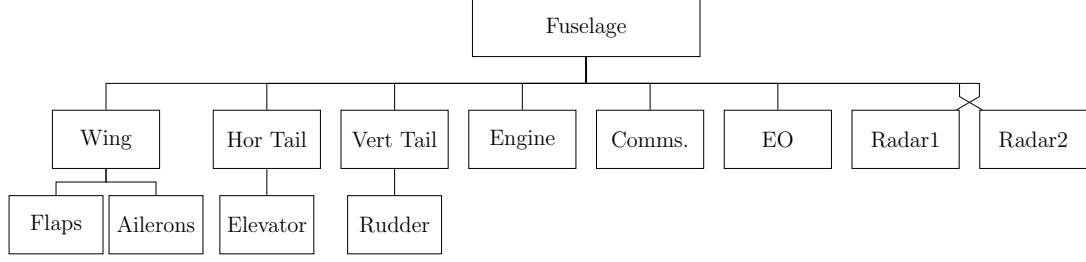


Figure 48: UAV Tree Diagram Outlining Physical Connections of Components and Subsystems (Fuselage Platform)

Equation 26 displays the proposed formulation of the online reconfigurability index.

$$OnRI = \frac{\#Inter_{On}}{\#Inter_{Tot}} \quad (26)$$

3.4.3 Offline Reconfigurability Index

Offline reconfigurable components rely on standardized interfaces that can handle different but the same type of components. In the automobile industry, a car can use various wheels because the interface between the car axes and the wheel has been standardized. For offline reconfigurability, if two components of similar type (two radars or two subsystem-packages) share the same interface with a parent component, then the interface is reconfigurable. The designer should organize the components in a tree diagram that relates physical connections between components and subsystems. Figure 48 shows a tree diagram of a basic tube-wing configuration. It outlines the physical connections among components. The configuration consists of a fuselage, a wing with flaps and ailerons, a horizontal tail with an elevator, a vertical tail with a rudder, an engine, a communications subsystem, an electro-optical sensor, and two radars. The radars share the same interface, suggesting both cannot be on the vehicle at the same time. Therefore, the radars' interface is reconfigurable.

When determining offline reconfigurability, the designer should choose a component that serves as the product's platform. The platform should reduce the number of levels in the physical-connection tree diagram, making it closest to all components

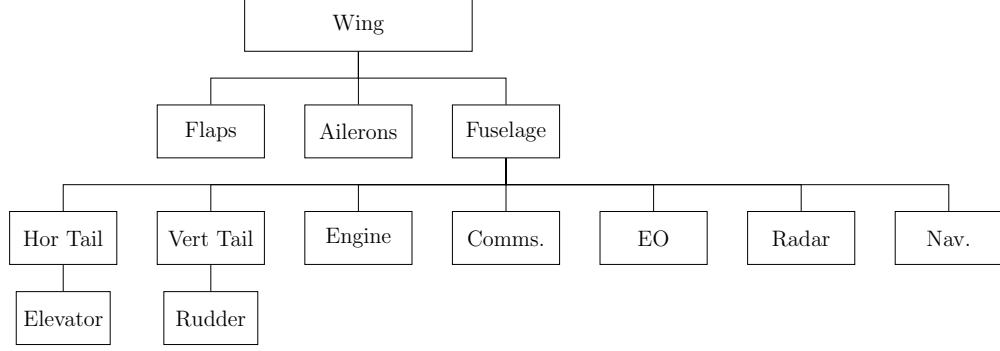


Figure 49: UAV Tree Diagram Outlining Physical Connections of Components and Subsystems (Wing Platform)

and subsystems. However, reducing the number of levels is not a rule but rather a general guideline. Figure 49 shows another example of a tree diagram of a basic tube-wing configuration. In this example, the wing is made to be the platform, but as Figure 49 shows, there are four levels compared to Figure 48's three. The difference between the two tree diagram shows the fuselage in a tube-wing configuration is the closest to all other components.

In summary, a offline reconfigurable interface can be defined as a connection that two components of the same type that share an interface with another single parent component. Therefore, the offline reconfigurability index should be the number of physical connections shared by components or subsystems of the same type divided by the total number of physical connections in the product line. Equation 27 shows the formulation of the offline reconfigurability index.

$$OffRI = \frac{\#Inter_{Common}}{\#Inter_{Tot}} \quad (27)$$

The creation of the quantitative architecture space leads to the assertion: Competing product architecture alternatives can be represented by common, online, and offline reconfigurable component characteristics. Therefore commonality and reconfigurability indexes can be used to generate alternative architectures numerically. Section 3.4.4 will demonstrate the formulation of these indices with respect to past

implemented product architectures in various industries.

3.4.4 Example F-14 Tomcat Product Family Indices Breakdown

The first case is the Grumman F-14 Tomcat. Over time, Grumman developed three variants of the vehicle. The variants were primarily upgrades as new technologies matured [115]. The fighter's first variant (F-14A) first flew in December 1970. The second variant (F-14A+ or F-14B) was developed in 1987. It replaced the engine with a higher thrust alternative. Also, the upgrade replaced the radar with a higher range alternative. The third variant (F-14D) was introduced in 1991. Again, the upgrade replaced the engines with more powerful and more efficient versions. Also, the new variant upgraded the electronics and avionics to include digital sensors and computers. The new engines and electronics allowed for better control, safety, and responsiveness.

An F-14 consists of fifteen primary components (components primarily used in the conceptual design phase): a fuselage, a cockpit, a radar, avionics, a main wing, flaps, ailerons, a horizontal tail, an elevator, a vertical tail, a rudder, two engines, and landing gear. Figure 50 shows the break down of these major components.

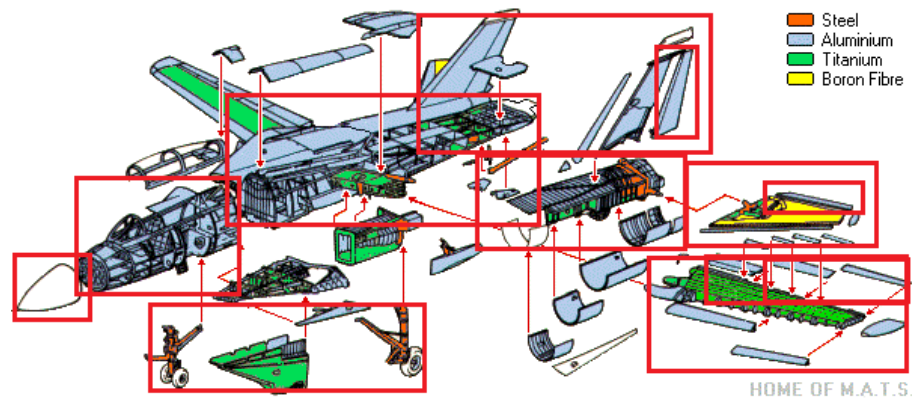


Figure 50: Grumman F-14 Tomcat in Architecture Space

Taking the information the breakdown of components provides, the commonality and reconfigurability indices (Section 3.4) for the F-14 Tomcat's product architecture

Table 41: Breakdown of Unique Components in the F-14 Product Line

Component	Variant						
	A	B	C	D	E	F	G
Fuselage	1	1	1	1	2	2	2
Cockpit	1	2	1	2	3	4	3
Radar	1	1	2	2	3	3	4
Avionics	1	1	2	2	3	3	4
Main Wing	1	1	1	1	2	2	2
Flaps	1	1	1	1	2	2	2
Ailerons	1	1	1	1	2	2	2
Horizontal Tail	1	1	1	1	2	2	2
Elevator	1	1	1	1	2	2	2
Vertical Tail	1	1	1	1	2	2	2
Rudder	1	1	1	1	2	2	2
Engines	1	1	2	2	3	3	4
Landing Gear	1	1	1	1	2	2	2

can be defined. First, there are seven variants of the F-14 (A, B, C, D, E, F, and G). Of these seven variants, the first four shared the same air frame but the radar, avionics, and engines were upgraded. The last three variants were larger in size than the first four. Variants B, D, and F all sat two pilots while the rest sat one. From this breakdown of components, there is sufficient conceptual information to formulate the commonality index for the F-14 product family. Each family member had 15 components, creating 105 total components. Table 2 shows the breakdown of unique components across the product line. The number associated with the variant and component relates to which variation of a component was used on the variant. For example fuselage 1 or 2.

Table 41 shows there are two unique fuselages, four unique cockpits, four unique radars, four unique avionics, two unique main wings, two unique flaps, two unique ailerons, two unique horizontal tails, two unique elevators, two unique vertical tails, two unique rudder, four unique engines, and two unique landing gears. The count of unique components therefore totals thirty-four. Plugging the results into Equation 25

gives Equation 28, and the commonality index for the F-14 Tomcat product family is 0.79.

$$1 - \frac{\#Comp_{Unique} - \max(\#Comp_{Tot_j})}{\sum_{j=1}^{J_p} (\#Comp_{Tot_j}) - \max(\#Comp_{Tot_j})} = 1 - \frac{34 - 15}{105 - 15} = 0.79 \quad (28)$$

After creating the commonality index for the F-14 Tomcat Product Family, the online and offline reconfigurability indices can be calculated. Taking the breakdown in components from Figure 50, the components can be arranged in a tree diagram to outline parents of each component. The best tree diagram minimizes the number of levels. As a result, Figure 51 emerges, where the red lines highlight the online reconfigurable interfaces and the wrapped lines represent the shared interfaces between components.

Figure 51 shows there are twelve online reconfigurable interfaces and twenty-four total interfaces in the product line. Plugging these results into Equation 26 produces Equation 29, and the online reconfigurability index for the F-14 Tomcat product family is 0.5.

$$\frac{\#Inter_{On}}{\#Inter_{Tot}} = \frac{12}{24} = 0.5 \quad (29)$$

Furthermore, Figure 51 shows there are twelve common interfaces and twenty-four total interfaces per design. Plugging these results into Equation 27 produces Equation 30, and the offline reconfigurability index for the F-14 Tomcat product family is 0.33.

$$\frac{\#Inter_{Off}}{\#Inter_{Tot}} = \frac{8}{24} = 0.33 \quad (30)$$

After calculating the three indices, the F-14 Tomcat can be placed in the product architecture space introduced in Figure 47. Equation 31 can translate the three dimensional indices into the two dimensional coordinates required for the space. The result is Figure 52.

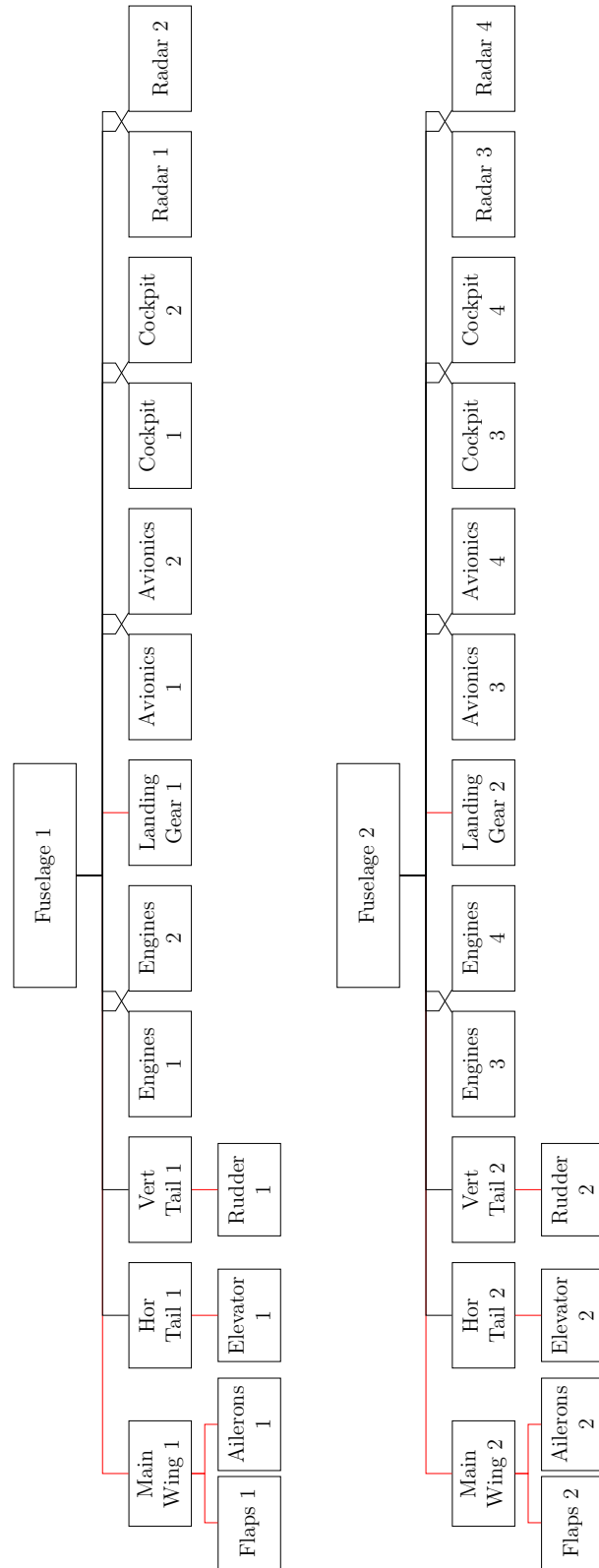


Figure 51: Tree Diagram Outlining F-14 Tomcat Physical Interfaces between Components and Subsystems

$$\begin{Bmatrix} X \\ Y \end{Bmatrix} = \begin{bmatrix} 0 & \cos \frac{\pi}{6} & -\cos \frac{\pi}{6} \\ 1 & -\sin \frac{\pi}{6} & -\sin \frac{\pi}{6} \end{bmatrix} \begin{Bmatrix} \textit{OfflineReconfigurability} \\ \textit{Commonality} \\ \textit{OnlineReconfigurability} \end{Bmatrix} \quad (31)$$

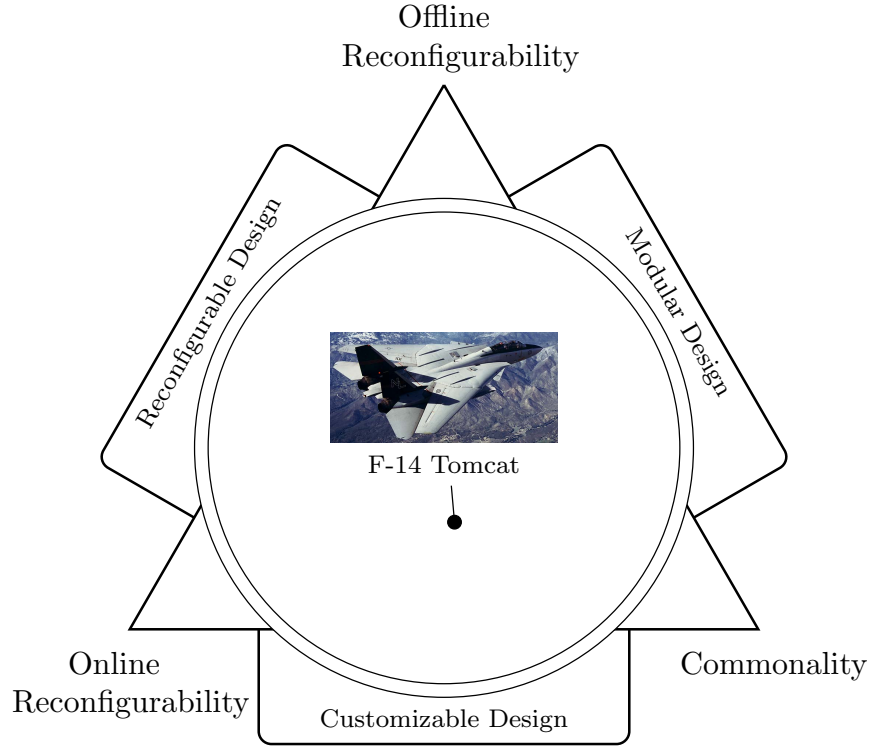


Figure 52: Grumman F-14 Tomcat in Architecture Space

3.5 *Evaluating Alternative Product Architectures by Quantifying Desirability, Flexibility, and Complexity*

Designers implement product architectures to reduce cost, increase performance, and satisfy customer and manufacturer desires. However, as observed in Section 1.5.1, systems engineers implement product architectures to reduce interactions among requirements and coupling among design variables, and as explained in Section 2.3, the interactions and couplings relate to a system's design complexity and requirement flexibility. This section looks to provide some numerical values to the terms that could be useful in evaluating the alternative product architectures.

3.5.1 Formation of Design Problem

When designing a product, system engineers try to maximize an overall evaluation criterion which are functions of the design space (\bar{x}) and requirements (\bar{R}). However, the design must satisfy constraints (\bar{g}, \bar{h}) formed by the customer needs and functional requirements and are functions of the design space and requirements. By setting up the design problem, it allows this dissertation to answer research question 5: *How are product architectures evaluated in a way that determines a product architecture's ability to satisfy requirements, resilience to changes in the industry associated with time, and the internal difficulty of developing and producing the new product quantified (desirability, flexibility, and complexity)?*

Traditionally, the desirability of a product is given by an Overall Evaluation Criterion (OEC), as explained in Section 3.5.2. However, the product must consider the constraints. Therefore, the design problem is given by Equation 32.

$$\text{minimize } [OEC = f(\bar{x}, \bar{R})] \text{ w.r.t. } [\bar{g}(\bar{x}, \bar{R}) \leq 0, \bar{h}(\bar{x}, \bar{R}) = 0] \quad (32)$$

Usually, to help with the optimization of the product, designers use a pseudo-objective function as seen in Equation 33, where ϕ_g and ϕ_h are penalty functions [60]. The summation of all the functional constraints creates the pseudo-objective constraint function ϕ .

$$\text{minimize } \left[\Phi(\bar{x}, \bar{R}) = f(\bar{x}, \bar{R}) + \sum_{i=1}^N \phi_{g_i}(\bar{x}, \bar{R}) + \sum_{j=1}^M \phi_{h_j}(\bar{x}, \bar{R}) \right] \quad (33)$$

Another representation of the pseudo-objective function is a Taylor series second order expansion. Using the Taylor series expansion, an application of Newton's Method minimizes the pseudo-objective function, as seen in Equation 34 where H is the Hessian of the pseudo-objective constraint function. Understanding this concept will become useful in the following sections.

$$\begin{Bmatrix} \Delta \bar{x} \\ \Delta \bar{R} \end{Bmatrix} = -H^{-1} \nabla \phi(\bar{x}, \bar{R}) \quad (34)$$

Equations 32 through 34 define the design problem. However, Equation 34 brings to light the how the design is sensitive to the change in design variables and requirements. The sensitivities relate to the resultant product's flexibility and complexity.

3.5.2 Creation of Desirability Metric

This dissertation needs to create a metric that captures the architectures ability to satisfy customer demands to answer Research Question 5.a: *How is the desirability of the product architecture determined?* The trends, from the historical data analysis mentioned in Section 3.2.2, provide a predicted or “preferred” architecture. If the systems engineers select a product architecture that differs from the “preferred” one, then the trade-off moving away from the predicted must be captured. Desirability (D) is a proposed metric that depicts the relation among the predicted and alternative architectures. Figure 53 displays desirability in the architecture space, where A , B , and C are different architectures, \hat{a} is the predicted or “preferred” architecture, and the arrows represent the difference between the alternative and “preferred” architectures desirability.

The desirability metric must satisfy the following criteria:

- Show ability of system to meet requirements
- Display preference based on technical or functional performance measures
- Clear and traceable logic for assigning preference

Traditionally, engineers represent desirability by an overall evaluation criterion (OEC). The OEC defines a metric that combines performance and cost values based

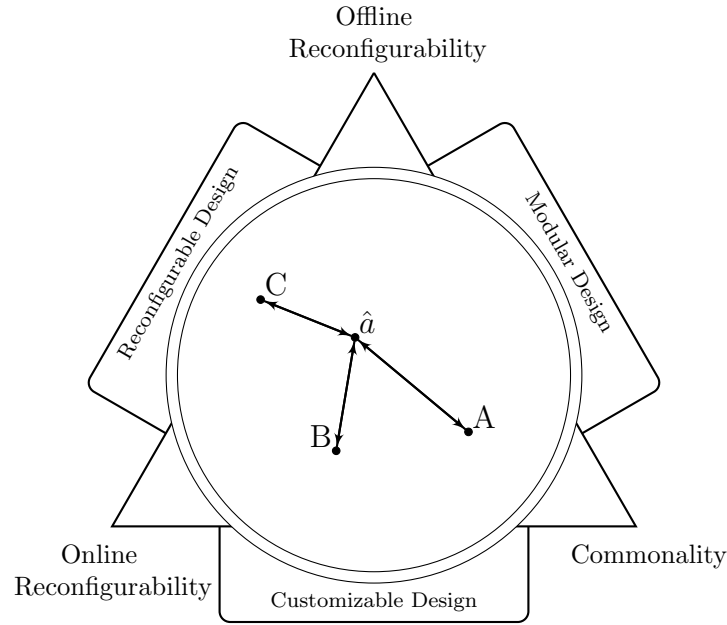


Figure 53: Desirability

on preference. Often, the OEC is normalized to prevent any of the metrics dominating the product architecture selection process. It is traditionally defined utilizing a Quality Function Deployment (QFD), which combines customer needs and functional requirements to create a weighted value equation. A desirable product architecture maximizes this metric with respect to any additional performance or cost constraints. An OEC captures requirement considerations; how well the system performs the mission, and its development is coherent and comprehensible. Therefore, desirability is the value of this OEC.

Section 2.2.1.1 explains the utility of a QFD. There the customer needs and functional requirements can help the systems engineers formulate preference of various technical performance metrics. For each product in the product line, the systems engineers must create an OEC. Each product must be able to conduct specific tasks and possess specific technical performance metrics.

The system engineers should break down the list of requirements by market and pair each requirement with a specific product family member. Then, the engineers

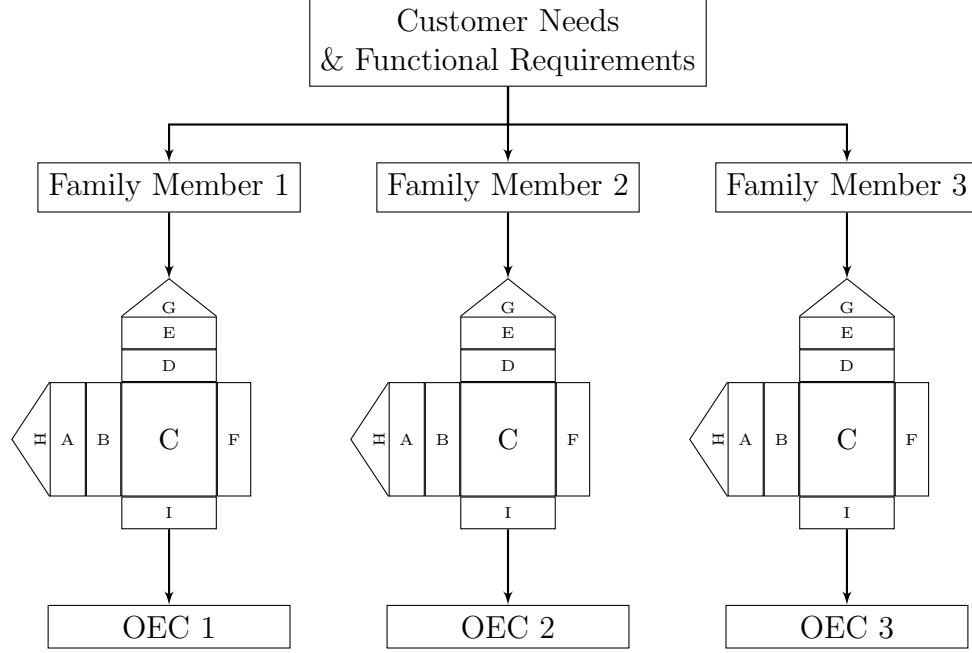


Figure 54: Overall Evaluation Criteria Creation

can conduct a QFD for each family member, defining each's OEC. Figure 54 outlines this process.

Therefore, the overall desirability is the sum of all OECs minus the penalty functions of all constraints (refer to Equation 33) in the product family divided by the number of product line members, creating a metric that varies from zero to one. For more detail on the formulation of desirability refer to Section 4.3.

3.5.3 Creation of Requirement Flexibility Metric

Since requirement flexibility plays a considerable role concerning product architecture selection, quantifiable equations of requirement flexibility (F_R) must be created. Hence, Research Question 5.b will be answered: *What is an appropriate definition and quantification of requirement flexibility in the context of product architectures?*

Products are flexible if they can perform more than designed. Furthermore, the roles the product conduct might change over time. A product can be profitable in the market as the market's preferences change. However, once the drivers reach a certain

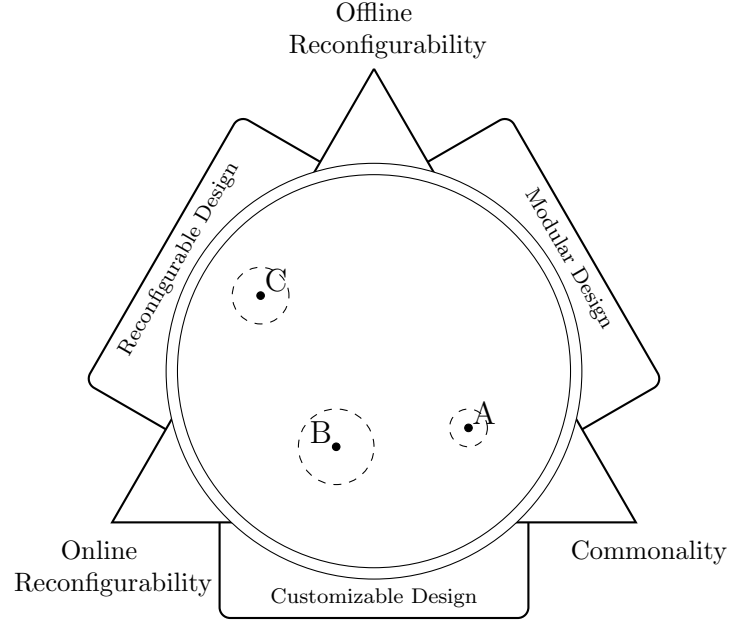


Figure 55: Flexibility

threshold, the product is no longer viable. Therefore, the manufacturer would have to develop a new product. Flexibility is a way to relate the drivers to the change in the product architecture. In Figure 55, the dashed circles represent the ability of the product architectures A , B , and C to perform tasks meant for other product architectures. Therefore, product architecture B is the most flexible out of the three.

A requirement flexibility metric must satisfy the following criteria:

- Capture the sensitivity of the product architecture to the requirements/drivers

Since requirement flexibility is the resilience of a product architecture to changing capabilities. Therefore, the product's sensitivity to changing requirements can produce its quantification. High requirement flexibility relates to high resilience and low sensitivity.

As set forth at the start of the section, the physics-based and disciplinary-based equations, showing the sensitivities of the product to changes in its design and allow for the development of sensitivity-based requirement flexibility. The derivation of requirement flexibility uses elements of the inverse Hessian and the gradient of the

product line's desirability function (Equation 37 and Equation 36).

$$H[D(\bar{x}, \bar{R})] = \left[\begin{array}{c|c} \frac{\delta^2 D(\bar{x}, \bar{R})}{\delta \bar{x}^2} & \frac{\delta^2 D(\bar{x}, \bar{R})}{\delta \bar{x} \delta \bar{R}} \\ \hline \frac{\delta^2 D(\bar{x}, \bar{R})}{\delta \bar{R} \delta \bar{x}} & \frac{\delta^2 D(\bar{x}, \bar{R})}{\delta \bar{R}^2} \end{array} \right] \quad (35)$$

$$\nabla D(\bar{x}, \bar{R}) = \left[\begin{array}{c} \nabla D_{\bar{x}}(\bar{x}, \bar{R}) \\ \nabla D_{\bar{R}}(\bar{x}, \bar{R}) \end{array} \right] \quad (36)$$

Equation 37 represents the inverse Hessian matrix of the product line's desirability as three sub-matrices: a , b , and c . The matrix a is a $n \times n$ matrix, b is a $n \times m$ matrix, and c is a $m \times m$ matrix, relating to the n design variables and m requirements.

$$H[D(\bar{x}, \bar{R})]^{-1} = \left[\begin{array}{c|c} a & b \\ \hline b^T & c \end{array} \right] \quad (37)$$

Equation 38 displays the formulation of requirement flexibility (F_R), combining the upper right con of the inverse Hessian matrix and the lower half gradient of the the product line's desirability function. This shows how a change in the requirements can impact the change in the design of the system. The more sensitive the design is to the requirements the less flexible it is.

$$F_R = \exp(-\| [b] \nabla D_{\bar{R}}(\bar{x}, \bar{R}) \|) \quad (38)$$

The metric is designed to vary from zero to one. Since there can be many designs associated with a single product architecture, a design space exploration can be conducted creating a distribution of requirement flexibility values. Systems engineers can compare the distributions of various product architectures to analyze the product architecture space.

3.5.4 Creation of Complexity Metric

The next step is to understand the complexity of the design problem. This dissertation observed seven specifically selected products that present different cases in the

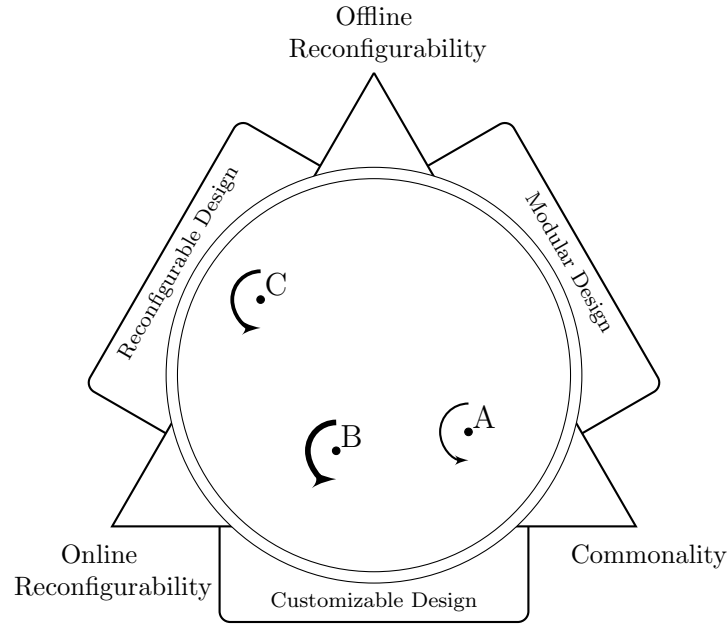


Figure 56: Complexity

architecture space in Section 3.4.4. From the analysis, systems engineers implement product architectures to reduce the couplings or interactions among the requirements, components, or disciplines, increasing the product’s flexibility and complexity. The couplings and interactions relate to the number errors made during design and development. The number of error directly relates to number of design changes and the cost of product development and production. Since the previous subsection already defined flexibility, this subsection answers research question 5.c: *What is an appropriate definition and quantification of complexity in the context of product architectures?*

Figure 56 displays complexity in the architecture space. The points *A*, *B*, and *C* are different architectures and complexity is represented by the curved arrows, the thicker the arrows, the more difficult the design problem associated with a product architecture.

The complexity metric must be able to satisfy the following criteria:

- Include Number of domains, functions, or disciplines
- Account for the level of interdependency among the domains, functions, or

Table 42: Components Broken Down by Discipline

Discipline	Component
Propulsion	Engine and/or Propeller
Aerodynamics	Wing, Empennage, Fuselage, and Engine Interaction
Structures	Wing and Fuselage
Controls	Wing, Empennage, and Engine
Manufacturing	Structures and Controls Results
Economics	Manufacturing Results

disciplines

- Consider the novelty of project
- Include the level of constraints' stringency

Traditionally in complex system design, an N^2 structure identifies the number of domains, components, and disciplines and their interactions and couplings. The N^2 structure identifies half of complexity's factors. For example, an N^2 diagram can show the number of components, their disciplines, and relations. Table 42 and Figure 18 show the disciplines and their associated components. Therefore, multi-disciplinary analysis can reveal the interdependencies or connections between the components and disciplines.

In the past, the Contact and Channel Model (C&CM) explained in Section 2.2 gave the connections weights. However, there are analytical and physics-based methods to represent the relationships. For this dissertation, the relations are considered constraints, objective functions, or relations. Some examples include Mattingly equations, cost approximations, and Breguet range/endurance equations. Derivatives of these mathematical functions provide the sensitivities.

From a review of literature and analysis of the problem, there are many ways to calculate complexity. The five methods identified in Section 2.3.2 provide various ways to calculate complexity. Out of the five, the complexity metric should capture

the difficulty of the design problem not necessarily the difficulty of achieving the requirements. Thus, the best analytical method to analyze complexity is by size. Complexity as size (C_{size}) analyzes the number of independent (idv) and dependent design variables (ddv), design requirements (dr), and measures of “goodness” (mg). Furthermore, it adds the number of modules (M^0) and connections among them (C^0) to show the additional magnitude of the problem. Equation 39 shows its formulation.

$$C_{size} = (M^0 + C^0) \ln |idv + ddv + dr + mg| \quad (39)$$

The representation of complexity as size provides an abstract method for determining the difficulty of the problem. However, the metric does not include the novelty of the project and the level of constraints’ stringency. Therefore, this dissertation uses a new metric that uses the information provided by conducting multidisciplinary analysis.

Complexity (C) is the ability of a design to be modified, which relates to the product architecture’s impact on the constraints, objective functions, or relations. These relations are design (performance) and process (manufacturing and costs) formulations and are usually nonlinear, meaning the first and even second derivatives are not constant. However, this dissertation does not focus on a specific design but rather the architecture’s relation with the development and production domains. Therefore, C is a combined measure of architecture’s impact on the sensitivity of design variables to each other.

From the beginning of this section, the physics-based and disciplinary-based equations, showing the sensitivities of the product to changes in its design, allow the development of sensitivity-based complexity. Equation 40 displays the formulation of complexity (C), combining the upper left corner of the Hessian matrix and the top half of the gradient of the product line’s desirability function.

$$C = 1 - \exp(-\|A\nabla D_{\bar{x}}\|) \quad (40)$$

The metric is designed to vary from zero to one. Since there can be many designs associated with a single product architecture, a design space exploration can be conducted creating a distribution of complexity values.

3.5.5 Formation of Hypothesis 3

From the review of past product architecture selections, systems engineers implement various characteristics to achieve multiple goals. The following are some reasons why designers implement product architectural characteristics:

- Designers use **fixed components** to design elements specifically for one role, thus increasing the performance of a single product in the product line.
- Designers use **offline reconfigurable components** to change the role of the system, thus increasing the requirement flexibility of the product without drastically increasing the cost.
- Designers use **online reconfigurable components** to enhance the performance of the system in various conditions, thus increasing the flexibility of the product sometimes at expense of the cost of the system.
- Designers use **common components** to reduce costs by sharing manufacturing processes, thus reducing the design complexity of the product.

Designers implement product architectural characteristics making these assumptions without fully understanding the consequences. The number of family members in the product line, types of requirements, or other factors can add unforeseen complexity to the product. This dissertation hopes to explore why, when, and how to implement product architectures. Therefore, it must test the validity of the assumptions listed above, leading to Hypothesis 3 and its sub-hypotheses:

Hypothesis 3

If implementation of product architecture characteristics (commonality and reconfigurability) have unforeseen or hard to predict consequences, then systems engineer require a new product architecture selection method or framework.

By testing these predisposed assumptions, they can be proven to be true, or new relations can be drawn to help designers understand the implications of implementing various product architectures. Section 3.7 describes how the approach to testing these hypotheses.

3.6 Formulation of Method to Identify Architectures of Interest

The new framework must identify regions of interest to facilitate an informed architecture selection. The process must be able to down-select architectures that are equally preferred or incomparable. Visualization of this process provides the architect with a greater understanding of the results. Therefore, a need exists to develop a visualization analysis to support quantitative decision making. This section answers Research Question 6: *What techniques can facilitate the process of analyzing and down-selecting regions of interest in the product architecture space?*

3.6.1 Definition of Pareto Identification

Pareto identification, often implemented in multi-objective/criteria design, identifies non-dominated systems as possible alternatives in the design process. Multi-objective/criteria problems tend to be complicated, and designers perform trade-offs between criteria. Pareto identification provides all possible solutions depending on the criteria's weightings. Non-dominated designs are solutions that are "better than all designs to which they can be compared, but that are incomparable to each other [60]."

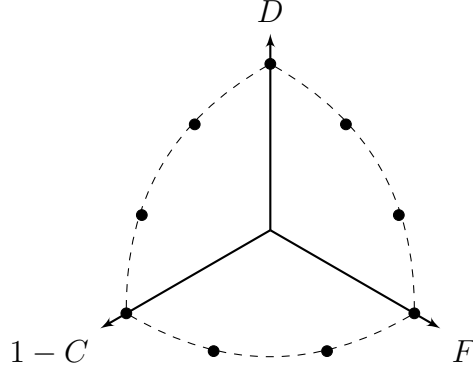


Figure 57: Theorized Product Architecture Pareto Frontier

For this dissertation, the evaluation metrics of product architectures are desirability, flexibility, and complexity. Therefore, the Pareto Frontier maximizes desirability and flexibility while minimizing complexity. As a result, Pareto frontier should look something similar to Figure 57.

3.6.2 Utilization of Impact Mappings to Form Flexible Pareto Frontier

Another way to determine architecture of interest is to find product architectures that are preferred over time, meaning they are resilient to changing drivers. The proposed method is to utilize phase portraits. “Phase portraits provide functions with an individual face and deepen our intuitive understanding of basic and advanced concepts in complex analysis [153].” A phase portrait shows how a state will change, based on initial conditions, if extrinsic variables change. Designers can use phase portraits to show nodal points where time independent preferred product architectures exist. Phase portraits of product architecture desirability, flexibility, and complexity can be combined to create Driver Impact Mappings. The nodal points that occur show stable points where the product architecture will see an equal trade between all of the evaluation metrics when a driver changes.

Section 3.2.3 introduced Driver Impact Mappings. For example, a common constraint in aircraft design is the Mattingly equation, which ensures the design has

enough thrust/power to achieve a mission segment. Equation 41 shows how the drivers (\bar{R}), design variables (\bar{x}), and product architecture (\bar{a}) impact the constraint. By taking the first and second derivatives of the Mattingly equation with respect to the design variables and requirements, quickly it becomes apparent how changing drivers impact desirability, flexibility, and complexity. Therefore, by impacting the evaluation metrics, it will impact the preferred product architecture.

$$\frac{T}{W} = \frac{K_1(\bar{x}, \bar{R}, \bar{a})}{\frac{W}{S}} + K_2(\bar{x}, \bar{R}, \bar{a}) + K_3(\bar{x}, \bar{R}, \bar{a}) \frac{W}{S} \quad (41)$$

Theoretically, superimposing the phase portraits of desirability, flexibility, and complexity concerning the drivers produces nodes where stable choices of product architectures exist. These solutions are assumed to be Pareto efficient, creating the product architecture Pareto frontier. The impact of drivers create different flows and behave differently. Therefore, the drivers behavior must be categorized, which occurs in the next subsection.

3.6.3 Categorization and Properties of Architecture Drivers

During this dissertation insights and contributions can be made providing value. One of them is categorizing the drivers' influence on the architecture space, including the drivers' preference and impact on architectures.

Two possible categories theorized are radial and tangential drivers. Radial drivers are ones whose partial derivatives are parallel to the index vectors, meaning as the driver changes, the architecture prefers to be dominated by that product characteristic. Tangential drivers are ones whose partial derivatives are tangent to the index vectors, meaning as the driver changes the architecture prefers to move towards another dominant product characteristic. Figure 58 displays the two driver categories: radial and tangential, respectively.

Categorization of drivers provides the architects with insights on strategic road

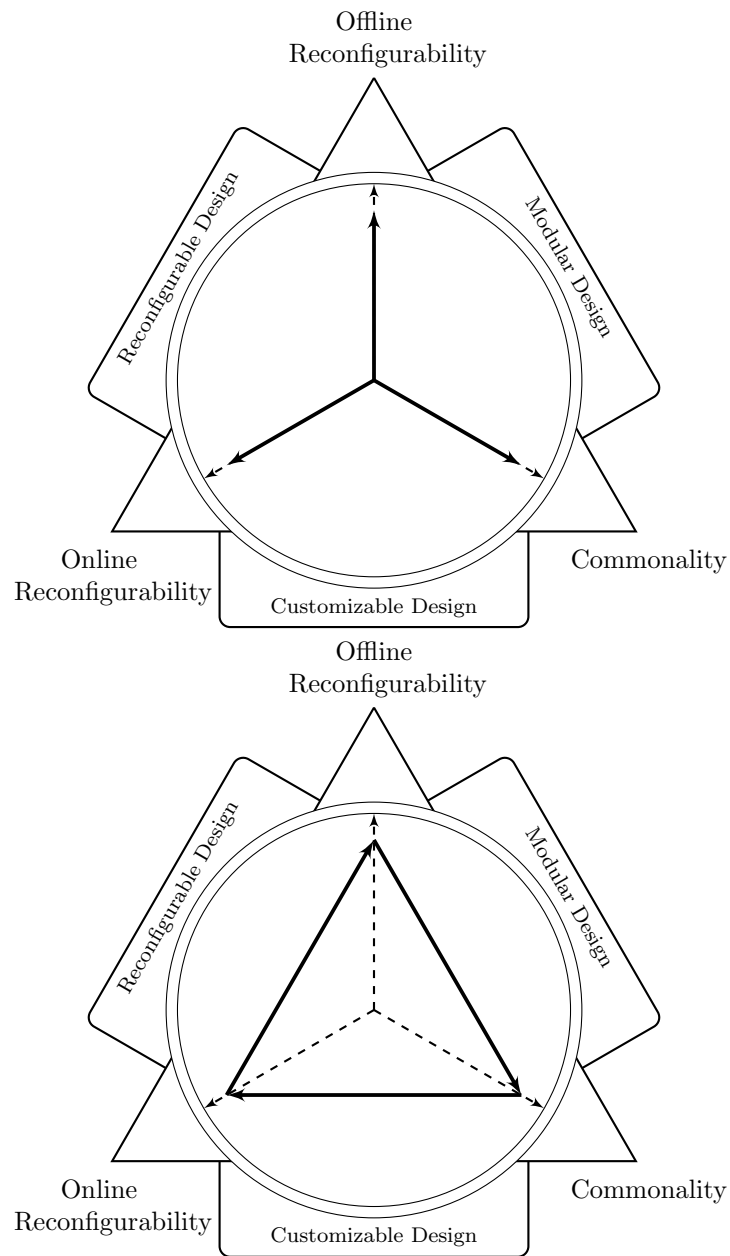


Figure 58: Theorized Driver Impact Categories: Radial and Tangential

mapping if requirements change, implying designers can predict the preferred product architecture's evolution.

3.7 Experimental Plan

The framework formulated in this dissertation must be able to achieve all the tasks laid out in Chapter 3, specifically Section 2.4. The framework must:

- Derive need of the product from the customer or business strategy
- Implement a method to clearly define the product's functional requirements from the needs
- Provide an understanding of how and which functional requirements drive the architecture
- Provide the systems engineers with insights on what types of product architectures are favored
- Provide a way to characterize and explore the space of alternative product architectures

Also, it must be able to compare many competing architectures, considering changes in design drivers (market, technology, and performance requirements), and aiding system architects in performing trade-offs between competing architecture designs.

Section 3.4 defines the architecture space which represents the space as indexes of component characteristics: common, online, and offline reconfigurable. The continuous space shows there are many architecture alternatives and attempts to simplify a complex qualitative problem. The conversion to a continuous domain gives the architect an approximation of what component characteristics the product architecture should possess and allows system engineers to compare many alternatives.

Section 3.2.1 identified drivers that influence architecture selection. Understanding the requirements and elements that impact the decision give architects greater knowledge of the process or problem's properties. Some industries' requirements or drivers vary over time, so product architectures evolved to meet these changes. The ability to analyze stochastic drivers is inherent to the architecture selection problem. Therefore, the systems engineers must possess the tools to pick a product architecture that best satisfies the requirements (desirability - Section 3.5.2) and is flexible (Section 3.5.3). The proposed framework must account for these properties.

Given that an inherent feature of a product architecture is the interactions and couplings among various disciplines, components, and subsystems, a data-driven methodology is needed to assess the impact of those interactions and couplings rigorously. The proposed way of understanding these features is through complexity, defined in Section 3.5.4.

Finally, the proposed framework must contain a means to identify product architectures or regions of interest in the product architecture space that the systems engineers should further explore (Section 3.6).

Combining the elements defined in Chapters 2 and 3 enables the formulation of a framework to facilitate the process of product architecture selection.

After answering the research questions, introduced in Section 2.5, the process begins to emerge. Figure 59 shows the finalized experimental plan.

The case study for this dissertation is a manufacturing firm's entry into the UAV industry. The firm must be able to understand what product architecture it should implement to enter and be successful in the market. The firm will go through the framework outlined in this chapter. Demonstrating the framework can validate the research. For example, experiments must be conducted to test past product architecture implementation assumptions (Section 3.7.1), test the sensitivity of the selected product architecture to the qualitative weightings (Section 3.7.2), and to confirm the

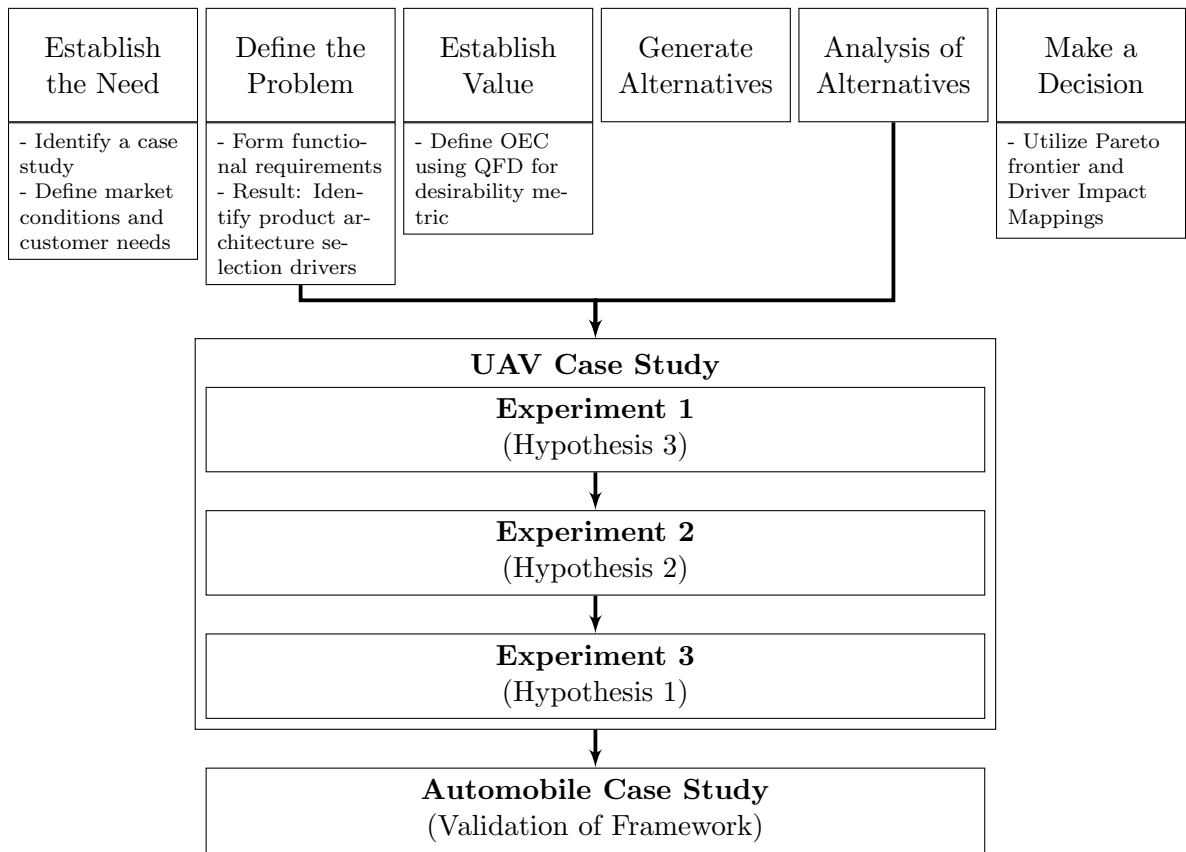


Figure 59: Experimental Plan

proposed product architectural drivers (Section 3.7.3). The first experiment tests the predisposed assumptions, listed in Section 3.5.5, verifying a need for numerical analysis of product architecture selection. The second experiment tests the validity of using the qualitative weighting system, outlined in Section 3.3. The final and third experiment verifies the drivers identified in Section 3.2, adding relevant and removing irrelevant ones. The drivers are essential to understanding the dynamics of the product architecture space. Finally, this dissertation applies the framework to a historical automobile industry case to prove the framework's utility. It will compare the decisions made in the industry against the recommendations provided by the framework.

3.7.1 Experiment 1: Testing Predisposed Assumptions in Product Architecture Selection

From the literature review there are assumptions systems engineers make that justify their reasoning to implement various product architecture characteristics. As a reminder, the most common are:

- Designers use **fixed components** to design elements specifically for one role, thus increasing the performance of a single product in the product line.
- Designers use **offline reconfigurable components** to change the role of the system, thus increasing the requirement flexibility of the product without drastically increasing the cost.
- Designers use **online reconfigurable components** to enhance the performance of the system in various conditions, thus increasing the flexibility of the product sometimes at expense of the cost of the system.
- Designers use **common components** to reduce costs by sharing manufacturing processes, thus reducing the design complexity of the product.

Repeating Hypothesis 3: *If implementation of product architecture characteristics (commonality and reconfigurability) have unforeseen or hard to predict consequences, then systems engineer require a new product architecture selection method or framework*, the goal of Experiment 2 is to test the validity of these assumptions. Most engineers implement product architectures without fully understanding the unforeseen consequences of their choice. Experiment 2 hopes to illuminate the process by analyzing the relationship between the product architecture and the evaluation metrics. In effect, Experiment 2 hopes to answer the following question:

- What are the reasons designers should implement product architectures, and what are the consequences of implementing their characteristics?

The required steps for this experiment and its sub-experiments are:

1. Determine the customer needs
2. Set the number of products (UAVs)
3. Derive the functional requirements for each product line member
4. Formulate the OEC for each product line member
5. Vary the product architecture by varying the levels of commonality, online, and offline reconfigurability
6. Conduct a design space exploration of the design variables to capture as much of the space as possible
7. Observe the trends in the product architecture characteristics and the evaluation metrics' distributions
8. Compare results with the assumptions

Experiment 1 should observe the impact of varying levels of commonality and reconfigurability on the performance, cost, requirement flexibility, and design complexity of the product line. By observing these characteristics of the product line, the experiment can determine if the four assumptions traditionally made during the implementation of product architectures hold true. Furthermore, if any unforeseen or hard-to-predict then the need for a new framework to facilitate the product architecture selection process is justified. Specifically, the experiment should determine if the percentage of the unique components is correlated with desirability, if both reconfigurable indexes are correlated with flexibility, and if the commonality index is negatively correlated with complexity. If all conditions are true and hold in all possibilities of the design, then hypothesis 3 will be proven as untrue and traditional methods can be used when selecting a product architecture.

If hypothesis 3 is proven to be true, the rest of the experiments can proceed. Experiment 1 will identify new trends and insights on the behavior of the problem. These results are essential to the dissertation and explain why these assumptions are not always valid. Furthermore, the results from Experiment 1 can support the following experiments. Therefore, if Experiment 1 formulates the OEC for each product, then the data produced in Experiment 1 is valid in Experiment 2 and 3.

3.7.2 Experiment 2: Testing the Validity of using Qualitative Weightings for various Metrics used to Evaluate the Product Architectures

This dissertation must test the validity of using the qualitative weighting technique presented in Section 3.3. Section 3.3 looked at the qualitative characteristics that surround requirement flexibility and design complexity. These characteristics or concepts can be hard to quantify and require some knowledge of the business case and internal processes and capabilities. Due to this stage's lack of numerical analysis, a way to test the validity of the stage is required.

Repeating Hypothesis 2: *If the qualitative weightings have significant influence*

on the decision of which product architecture to implement, then the most favorable product architecture should be vastly different at various weightings. The goal of this experiment is to test the impact of each of the weights on the selected product architecture. Furthermore, it should determine if the magnitude of the impact is proportional to the implied qualitative weight proposed in Section 3.3. In effect, Experiment 2 hopes to answer the following question:

- Is the proposed weighting system an appropriate approach in determining which product architecture should be the most favorable?

The required steps for this experiment are:

1. Determine the customer needs
2. Set the number of products (UAVs)
3. Derive the functional requirements for each product line member
4. Formulate the OEC for each product line member
5. Vary the levels of the weights providing extreme cases
6. Observe the impact of varying the weightings on the final selected product architecture
7. Compare the magnitude of change of commonality and reconfigurability of each case
8. Determine if the magnitude of change corresponds to the qualitative insight that provides the weightings

Experiment 2 will be conducted as a sensitivity study of the weightings' impact on the finally selected product architecture. Extreme cases where only one or two

metrics are important will be compared against each other and the original weightings determined by the case study found in Chapter 5. If the change in weightings correspond well to the results found in Experiment 1 and their corresponding final selected product architecture, then Hypothesis 2 will be proven true.

3.7.3 Experiment 3: Validation of Product Architecture Drivers

This dissertation must validate the drivers of product architecture selection. Section 3.2 investigated past industries to determine what factors influenced the product architectures adopted by each industry. Table 39 lists the resulting factors. Though the list provides a diverse set of requirements, this dissertation must test the assumption that these drivers are sufficient.

Repeating Hypothesis 1: *If the product architecture selection drivers identified truly drive the decision, then they must significantly impact the process's results by influencing the levels of commonality and reconfigurability in the product line.* The goal of this experiment is to determine which and how the identified drivers impact the product architecture selection process. Furthermore, the experiment should determine which drivers, previously identified in Section 3.2, are irrelevant and what other requirements might be possible drivers. In effect, Experiment 3 hopes to answer the following question:

- What are the requirements that drive the product architecture selection process?

The required steps for this experiment are:

1. Determine the customer needs
2. Set the number of products (UAVs)
3. Derive the functional requirements for each product line member
4. Formulate the OEC for each product line member

5. Conduct a design space exploration of the design variables, product architecture characteristics, and requirements to capture as much of the space as possible
6. Determine which of the identified drivers have statistically significant impact on a product architecture's desirability, flexibility, or complexity
7. Determine what alternative requirements have statistically significant impact on a product architecture's desirability, flexibility, or complexity

This dissertation will conduct a screening test/ANOVA to determine which drivers impact the product architecture selection process. An ANOVA is a statistical method that determines whether the variance in an input variable drives the variance seen in the output metrics. If all of the drivers influence either the desirability, flexibility, or complexity of the product architecture, then Hypothesis 1 holds true. If not, the conclusions must drop the drivers that are irrelevant. If any of the other requirements impact the evaluation metrics, the conclusions must deem them as drivers.

3.7.4 Validation of Framework

Finally, the proposed method must be verified and validated throughout its formulation. Figure 60 displays the verification and validation of the proposed method through the experiments.

The framework follows the generic engineering decision support process. The process establishes the needs of the customer and manufacturer, defines the problem, establishes a value for a product architecture, generates alternatives, evaluates the alternatives, and provides a final decision. However, there is still some questions or uncertainty about the framework. The two experiments will verify the steps in the process, and the automobile case study will provide the overall validation of the framework.

The automobile industry will provide a historical case studies where data is easily attainable to the public and product architectures are easily discernible. The case

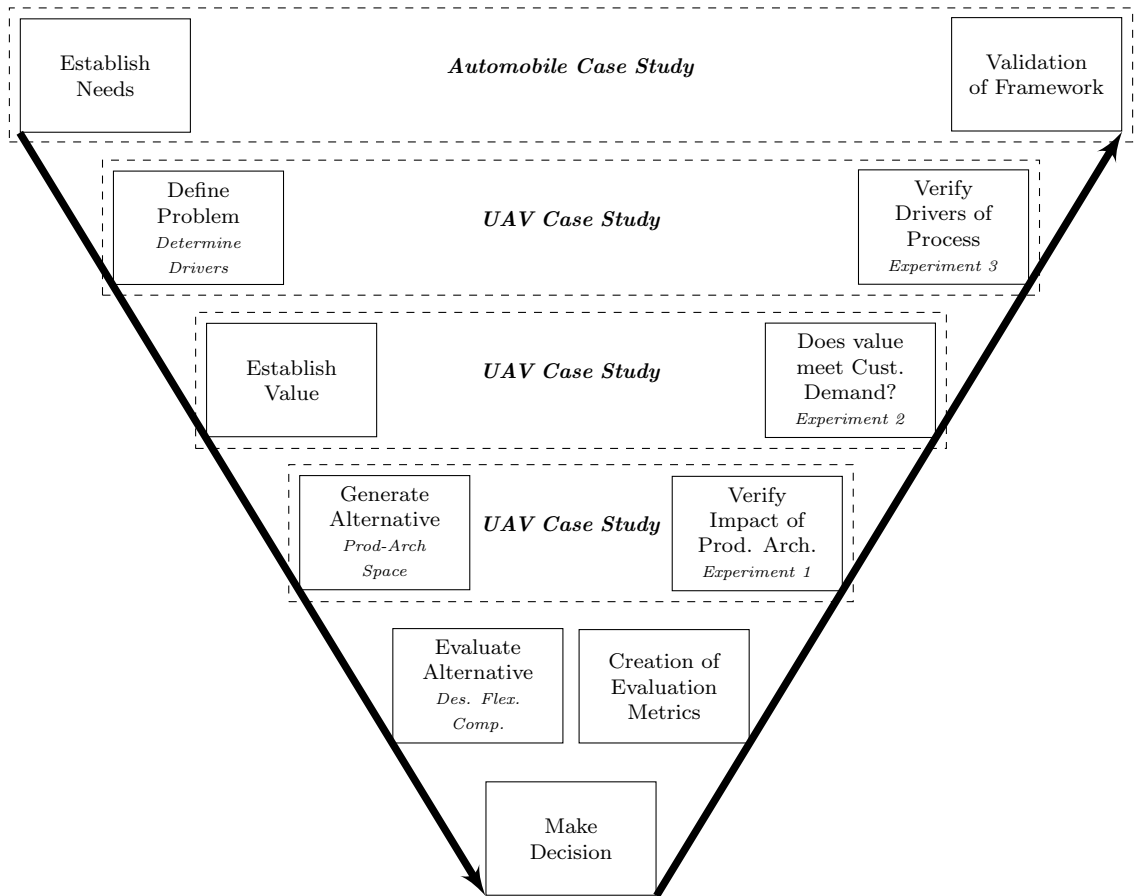


Figure 60: Verification and Validation Plan for Proposed Method

study will test the validity of the framework by comparing the historical decisions made throughout the process against the results derived from the framework.

3.8 Formulation of Framework Conclusion

Though this dissertation analyzes cases from other industries during the background research, it is not its focus. This dissertation utilized these cases to draw observations necessary for the creation of the new framework. The observations and insights gained from the other industry are relevant to the unmanned aerial vehicle industry because the UAV industry implements a broad range of product architectures. Therefore, a wide variety of industries must be observed to understand the problem architecture selection problem.

The next chapter outlines the modules and simulation models required to conduct the experiments. Then, the following chapter tests the framework using a case study of a UAV manufacturer. The case study will provide the data required to run the experiments and verify the framework. The following chapter validates the framework by using an automobile case study taking key points in the industry to use the framework. The results from the framework should match the decisions that promoted companies' success. The validation case study will show the proposed framework's utility.

CHAPTER IV

DEVELOPMENT OF THE ANALYTICAL MODULES REQUIRED BY FA²UST MODULE

The framework and experiments outlined in the last chapter require modules integrated together to allow for product architecture analysis. The modules required were incorporated into a module named after FA²UST. The module consists of an outer layer that defines and provides rules for the product architecture and an inner layer that primarily consists of the sizing and synthesis models used in traditional design practices. Before going into detail of the layers, it is important to provide some definitions of the objects and their roles within the FA²UST module. The FA²UST module was built in Java which is an object-based scripting language and operators can tie any type of product to the outer layer. Therefore, abstract objects were created to give the operator this flexibility. Figure 61 shows these objects and their relationships as they pertain to product architecture analysis.

Figure 61 shows there are seven objects that make up the abstract elements of FA²UST. Each has specific roles, behaviors, and structure allowing for an abstract construction of a product. Some definitions are provided to give clarification to each object:

- Values
 - Variables
 - * are the minimum number of values that can describe the properties of an entity.
 - * are assigned to one or multiple common entities.

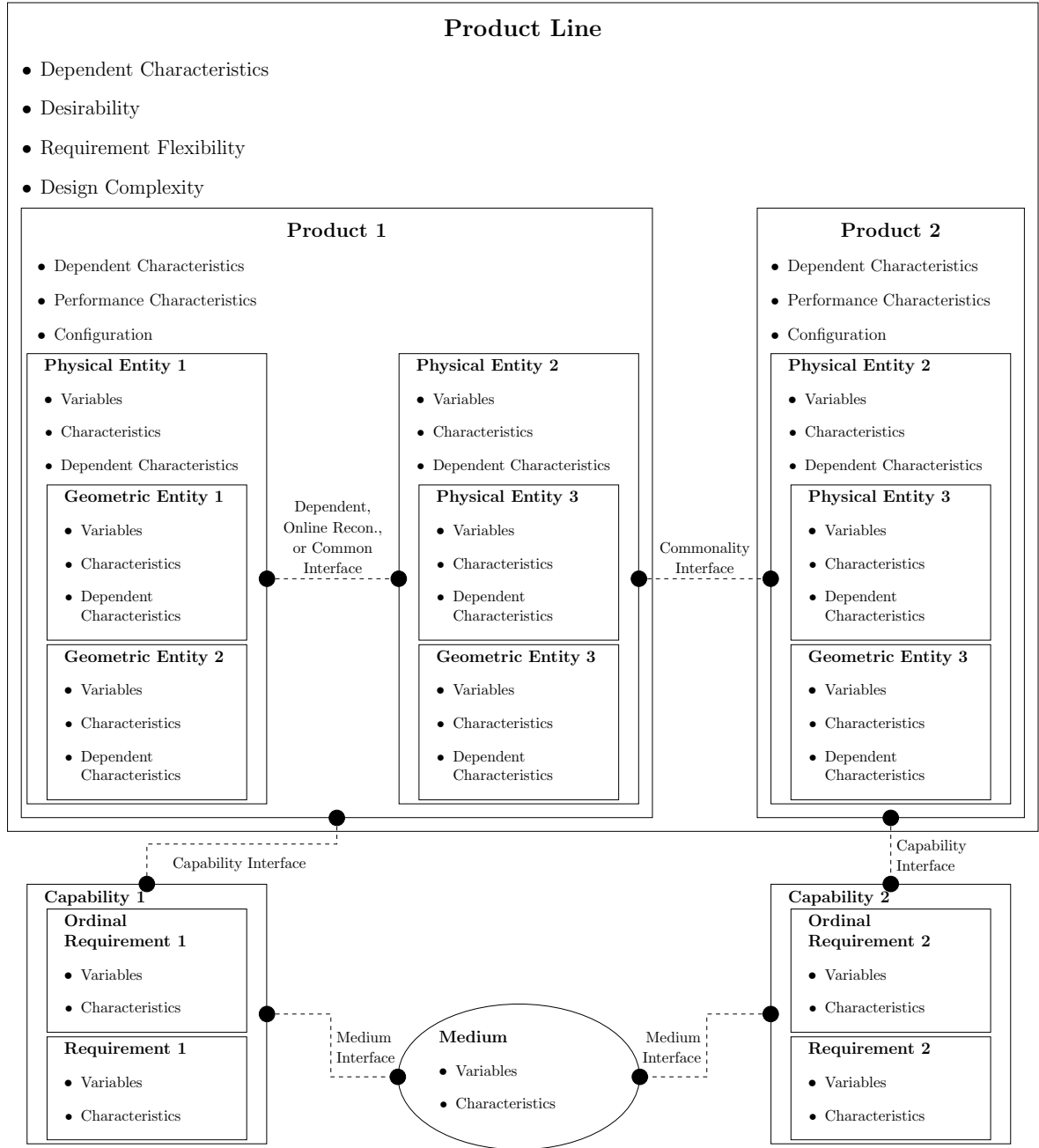


Figure 61: Definitions of Entities and Objects used within FA²UST

- Characteristics
 - * are functions of an entity's own variables without any external dependence.
- Dependent Characteristics
 - * are functions of an entity's own and external entity's variables.
 - * are dependent of another or multiple entities' variables.
- Performance Characteristics
 - * are functions of how a product, its subcomponents and its subsystems interact with the environment.
- Entities
 - Physical Entity
 - * is a physical object, usually a component or subsystem, which is a part of the composition a product or parent component.
 - * must have some geometric entity (explained later) to outline its shape.
 - * can be composed of other subcomponents or subsystems (other physical entities) to provide its full description and capabilities.
 - * consists of variables; characteristics; dependent characteristics; and its internal geometric and physical entities variables, characteristics, and dependent characteristics.
 - Geometric Entity
 - * is a shape that describes the shape of a physical entity.
 - * consists of variables and characteristics.
- Configuration
 - outlines the arrangement of physical entities within a product.

- Product
 - is a composition of physical entities and the interfaces between them creating a system that has its own emergent characteristics that emerge when operating in a medium.
 - has one or multiple capabilities paired with it in order to size and determine the performance characteristics.
 - the performance characteristics are determined from the product's interaction with the environment (medium) during operations.
- Requirements
 - Lone Requirement
 - * is an independent requirement demanding certain capabilities of a product.
 - Ordinal Requirement
 - * is a requirement or mission segment that has some order in the overall operations of the product.
 - * is dependent of other ordinal requirements.
- Capabilities
 - are compositions of lone and ordinal requirements that a product can achieve.
- Medium
 - is an environmental object which the product or products interact with.
 - is used to determine a product's or products' performance during operations.

- consists of variables, characteristics, and performance characteristics.
- Interfaces
 - Dependent Interfaces
 - * is a connection between two physical entities that defines the dependent characteristics of itself, its geometric entities, or its physical entity
 - Commonality Interface
 - * is a connection between two physical entities that makes the two common meaning the variables, geometric entities, and physical entities are always the same.
 - Online Reconfigurable Interface
 - * adds a degree of freedom to the control of the aircraft allowing the two components to change its orientation between each other.
 - * multiple online reconfigurable interfaces can be applied to the same two components.
 - Common Interface
 - * shows that there may be different operational modes where one component is preferred over another of the same type (two types of wings).
 - * ensures the dimensions, information (data or energy) flow, and connection point of the interface is common, standardizing the interface.
 - Capability Interface
 - * pairs a product with a capability.
 - Medium Interface
 - * determines the environment during a requirement or operations of a product.

Now that the abstract objects have been defined, the outer layer of the FA²UST Module can be explained. The outer layer ensures commonality, online reconfigurability, and offline reconfigurability during the sizing and synthesis of the product line. Also, it allows the calculation of a product architecture's desirability, requirement flexibility, and design complexity. With this information, the operator of the module will have enough information to conduct the experiments described in the previous chapter and conduct product architecture analysis.

4.1 FA²UST Module Outer Layer Inputs

The FA²UST Module's outer layer consists of two elements: managing the inputs from the module's operator and calculating the product architecture evaluation metrics. The first part reads the user's inputs to the module and sets forms of commonality, online reconfigurability, and offline reconfigurability. Also, it sets the designs and their composition of components and subsystems based on variable inputs defining the design of each product in the product line.

When the FA²UST Module loads the input file, it generates vectors that define the number of products within the product line, the design missions associated with each product, any additional requirements associated with each product, cost analysis parameters, and a series of components and their values for their variables.

During the initialization, the module reads in a list of each type of component. For example, the list will include engine 1, engine 2, and so on. As it creates this list, links specific design variables to each. For example, engine 1 has 100 horsepower. The module does this for all components provided by the input file. Then it creates a list of products that make up the product line. Each product has an index list associated with each type of component which allows the module to figure out which components are included in the product's configuration. Once the list has been generated, the module will either receive the orientation of components from the input

file or use a default orientation. The orientation provides the structure (parent and child component) of the product, as Figures 65 through 68 show. Also, the input file provides some switches that can turn on and off some online reconfigurable interfaces. The module compiles vectors of these switches with the lengths equaling the number of products in the product line. A switch is represented with a 1 for on and a 0 for off. For example in the UAV analysis, these include flaps ($Flaps$), variable sweep main wing ($VarSweep$), variable pitch main wing ($VarPitch$), variable pitch propeller ($VarProp$), and thrust vectoring ($VecThrust$). With all of this information, the module can generate the commonality, online reconfigurability, and offline reconfigurability of the product line.

4.1.1 Inputing the Commonality of a Product Architecture

The index lists from all components can be combined into a two dimensional array of indices ($[Comp]$), where the row vectors represent a component type and column vectors represent a product. It is important to note some of the elements of the two dimensional array might be empty, implying the product does not have this type of component. The number of indices in a column represent the number of components in that product, and the number of indices in the entire array represent the total number of components in the product line. When the redundant indices are removed from each row, the array becomes a list of unique components within the product line. These values provide enough information to calculate the commonality index found in Equation 25.

4.1.2 Inputing the Online Reconfigurability of a Product Architecture

The orientation of components provides the total number of interfaces in a product. Combining the total for each product provides the total number of interfaces in the product line. The vector sum of all the online reconfigurable switches provide the number of online reconfigurable interfaces in the product line. These values provide

enough information to calculate the online reconfigurability index found in Equation 26.

4.1.3 Inputing the Offline Reconfigurability of a Product Architecture

The orientation of components defines the parent child relationship between all components. If a parent has two children of the same component type then the two children share a common interface. Using this relationship, the module iterates through each interface within the product line and counts the number of common interfaces. The total number of interfaces and the number of common interfaces provide enough information to calculate the offline reconfigurability index found in Equation 27.

4.2 FA²UST Module Inner Layer

The inner layer of the FA²UST Module depends heavily on the type of product the engineers are analyzing. Models can be built up and added to show individual component characteristics and performance. However, it is also important that models are incorporated to show the interactions and emergent behavior that occurs when the components are combined. This section will look at the two products analyzed: UAVs and automobiles.

4.2.1 UAV Sizing and Synthesis Models

Models from existing aircraft sizing and synthesis modules and data from existing UAVs were combined to create a component based UAV sizing and synthesis module. The module has the ability to size and optimize certain characteristics of the aircraft relative to requirements or constraints placed on the design. It consists of geometric entities, physical entities, a medium entities, configuration rules, and integrated system architecture behaviors specific to UAV design. Using this information, the module can predict the development and production cost of a UAV product family. The information provided by the module is important in analyzing the product

architecture of the product line in the outer later of the FA²UST Module.

4.2.1.1 UAV Geometric Entities

The FA²UST Module uses three main geometric entities to describe the shape of the components used in UAV design. These geometric entities include the planform, airfoil, and cylinder.

Planform: The aerodynamic surfaces, explained later in this section, require a planform to describe their flat, two-dimensional shape.

Planform Variables: The FA²UST Module explains a planform with four variables: area (S), aspect ratio (AR), taper ratio (λ), and quarter chord sweep ($\Lambda_{1/4}$). From these four variables the rest of the characteristics of the planform can be explained.

Planform Characteristics: The planform characteristics important in the aerodynamic analysis of a UAV are leading edge sweep (Λ_{LE}), span (b), root chord (c_r), tip chord (c_t), and average chord length (\bar{c}). Equations 42 through 46 show how the four variables can define of the characteristics.

$$\Lambda_{LE} = \tan^{-1} \left[\tan (\Lambda_{1/4}) + \frac{1 - \lambda}{AR \times (1 + \lambda)} \right] \quad (42)$$

$$b = \sqrt{S \times AR} \quad (43)$$

$$c_r = \frac{2 \times S}{b \times (1 + \lambda)} \quad (44)$$

$$c_t = c_r \times \lambda; \quad (45)$$

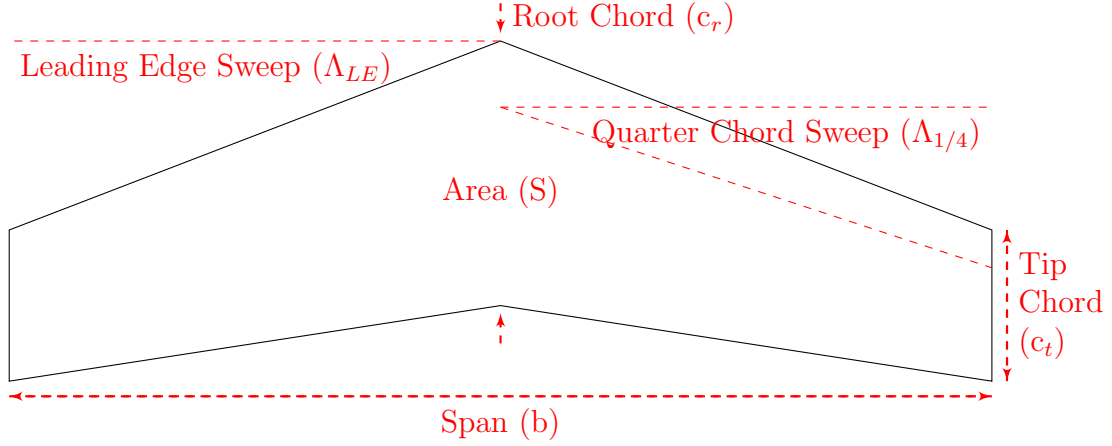


Figure 62: Example Planform Shape for Aerodynamic Surfaces

$$\bar{c} = \frac{2 \times c_r \times (1 + \lambda + \lambda^2)}{3(1 + \lambda)} \quad (46)$$

All of the variables can be combined together to define the shape of the planform as shown in Figure 62.

Airfoil: The aerodynamic surfaces, explained later in this section, require a planform to describe their flat, two-dimensional shape and an airfoil profile to provide the three-dimensional shape. The airfoils included in the FA²UST Module are displayed in Table 43. The airfoil defines some key aerodynamic parameters that help define the overall performance of the aerodynamic surface. These include maximum thickness to chord ratio (t/c), location of the maximum thickness (x/c), the two-dimensional lift coefficient of the airfoil at zero angle of attack ($C_{l_{\alpha=0}}$), the two-dimensional lift coefficient derivative with respect to angle of attack ($C_{l_{\alpha}}$), the maximum two-dimensional lift coefficient ($C_{l_{max}}$), the minimum two-dimensional drag coefficient ($C_{d_{min}}$), the two-dimensional lift coefficient's value at minimum drag ($C_{l_{atC_{d_{min}}}}$), the two-dimensional moment coefficient at zero angle of attack ($C_{m_{\alpha=0}}$), and the two-dimensional moment coefficient derivative with respect to angle of attack ($C_{m_{\alpha}}$). All of these parameters are important in determining the three dimensional forces on the aircraft.

Table 43: Options of Available Airfoils in the FA²UST Module

Airfoil	t/c	x/c	$C_{l_{\alpha=0}}$	$C_{l_{\alpha}}$	$C_{l_{max}}$	$C_{d_{min}}$	$C_{lat}C_{d_{min}}$	C_{m_0}	$C_{m_{\alpha}}$
NACA-X-4-0009	0.09	0.309	0	5.73	1.3	0.0055	0	0	0
NACA-X-4-0012	0.12	0.3	0	5.73	1.4	0.023	0	0	-0.353
NACA-X-4-2415	0.15	0.295	0.2	6.02	1.6	0.065	0.3	-0.2	-0.358
NACA-X-4415	0.15	0.309	0.4	5.73	1.65	0.065	0.4	-0.38	-0.573
NACA-X-5-23015	0.15	0.3	0.1	6.30	1.7	0.062	0.2	-0.02	-0.220
NACA-X-6-64-415	0.15	0.3	0.35	6.21	1.6	0.042	0.6	-0.21	-0.143

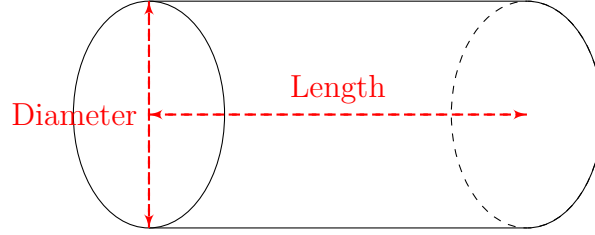


Figure 63: Example Cylindrical Geometric Entity

Cylinder: Many of the components in the FA²UST Module use a cylindrical geometric entity to represent their shape. The cylinder is a simplistic representation but provides enough information to inform the conceptual design of the system. A cylinder only requires two variables to define its shape: diameter and length. Figure 63 outlines a general cylindrical shape. As explained later in this chapter, components such as the subsystems, engine, and fuselage will be explained geometrically by this elementary shape.

4.2.1.2 UAV Physical Entities: Components and Subsystems Breakdown

The FA²UST Module incorporated many types of components and subsystems into its capabilities. The models consist of previously established historical or experimental-based regressions that help explain the characteristics or performance of a component.

Subsystems and Payload: Specifically, the FA²UST Module considers five types of subsystems: radar, EO-IR sensors, global positioning systems (GPS), inertial navigation systems (INS), and communications. Data was collected on the subsystems

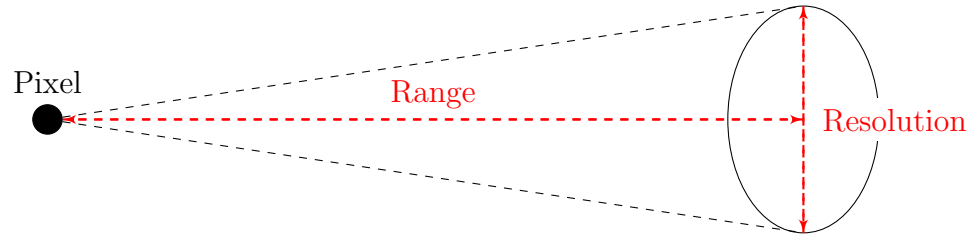


Figure 64: Range-Resolution Model Incorporated into Radar and EO-IR Sensor Subsystems

existing UAVs used, allowing the subsystems to be sized based on mission requirements and capabilities. The database created regressions which made this sizing possible. The subsystems had many different variables that effect the sizing. These new regressions reflect a better understanding of how the subsystem is designed and integrated into the vehicle.

Radar: The radar subsystem sizing consists of variables and characteristics. The operator of the module has the option to make some of the characteristics dependent on some of the mission segments.

Radar Variables: The driving capability or variables for the radar was the range (miles) per resolution (feet). These variables are based on how a radar functions. When a radar scans the surroundings, the radar takes images and transforms the images into a two or three dimensional image the operator or computer can process. The image is presented as a composition of pixels which represent a portion of the greater surroundings. Figure 64 shows how the surroundings are projected onto a pixel. The radar has a cone of vision that relates to one pixel the ratio of range versus resolution, defining the capability of the subsystem. A database of existing radar used on existing UAV platforms was compiled. From the data, regressions for radar characteristics were derived from the variables.

Radar Characteristics: The range (miles) and resolution (feet) variables define the characteristics of a radar subsystem. The characteristics are the radar's weight,

energy consumption, diameter, and length. These characteristics help define the geometric entity of the radar which is a cylinder. Equations 47, 48, 49, and 50 displays the radar's weight, energy consumption, diameter, and length respectively. The weight is in pounds, power is in Watts, diameter in feet, and length in feet.

$$W_{Radar} = 16.8 \times \frac{Range^{0.877}}{Resolution} : R^2 = 0.913 \quad (47)$$

$$\begin{aligned} P_{Radar} &= 254 + 56.3 \times \frac{Range}{Resolution} \\ &\quad + 0.755 \times \left(\frac{Range}{Resolution} - 19.1 \right)^2 \\ R^2 &= 0.968 \end{aligned} \quad (48)$$

$$Diameter_{Radar} = 0.743 \times \frac{Range^{0.220}}{Resolution} : R^2 = 0.922 \quad (49)$$

$$\begin{aligned} Length_{Radar} &= 0.428 + 0.0793 \times \frac{Range}{Resolution} \\ &\quad + 1.18 \times 10^{-3} \times \left(\frac{Range}{Resolution} - 19.1 \right)^2 \\ R^2 &= 0.999 \end{aligned} \quad (50)$$

When it comes to unadjusted function points for a radar, it is assumed the radar is connected to the processor which requires a high-complexity external input and output. It will be transferring images back to the processor which will process them and send commands back to the radar. Since it will wait for some commands from the processor, it will require an external inquiry which outlines what functions the radar wants to perform. Also the interfaces between the processor and the radar require two external interfaces that determine how the two subsystems will communicate with each other. Finally, the radar will require at least one logical file to interpret

Table 44: Calculating the Total Number of Unadjusted Function Points for a Radar [84]

Type of Component	Complexity of Components			
	Low	Average	High	Total
External Inputs	0 x 3 = 0	0 x 4 = 0	1 x 6 = 6	6
External Outputs	0 x 4 = 0	0 x 5 = 0	1 x 7 = 7	7
External Inquiries	0 x 3 = 0	0 x 4 = 0	1 x 6 = 6	6
Internal Logical Files	0 x 7 = 0	0 x 10 = 0	1 x 15 = 15	15
External Interface Files	0 x 5 = 0	0 x 7 = 0	2 x 10 = 20	20
Total Number of Unadjusted Function Points =				54

commands and process its own processes. As a result from using Table 44, a radar is approximated to have 54 unadjusted function points.

Radar Optional Dependent Characteristics: The radar must be able to have a certain resolution over a range. Therefore, the FA²UST Module has the ability to connect the variables with a dependent interface to a mission segment. The operator of the FA²UST Module can define a ratio for which a cruise segment's range is proportional to the range versus resolution: $\frac{Range}{Resolution} = K \times Range_{Cruise}$.

EO-IR Sensor: The EO-IR sensor subsystem sizing consists of variables and characteristics. The operator of the module has the option to make some of the characteristics dependent on some of the mission segments.

EO-IR Sensor Variables: The variables for an EO-IR sensor are the same as a radar: Range (miles) per resolution (feet). These variables are based on how an EO-IR sensor functions. When an EO-IR sensor scans the surroundings, the radar takes images and transforms the images into a two or three dimensional image the operator or computer can process. The image is presented as a composition of pixels which represent a portion of the greater surroundings. Figure 64 shows how the surroundings are projected onto a pixel. The EO-IR sensor has a cone of vision that relates to one pixel the ratio of range versus resolution, defining the capability of the

subsystem. A database of existing EO-IR sensors used on existing UAV platforms was compiled. From the data, regressions for EO-IR sensor characteristics were derived from the variables.

EO-IR Sensor Characteristics: The range (miles) and resolution (feet) variables define the characteristics of an EO-IR sensor subsystem. The characteristics are the EO-IR sensor's weight, energy consumption, diameter, and length. These characteristics help define the geometric entity of the EO-IR sensor which is a cylinder. Equations 51, 52, 53, and 54 displays the EO-IR sensor's weight, energy consumption, diameter, and height respectively. The weight is in pounds, power is in Watts, diameter in feet, and height in feet.

$$W_{EO/IR} = 5.39 + 23.0 \times \frac{Range}{Resolution} : R^2 = 0.986 \quad (51)$$

$$P_{EO/IR} = 13.2 \times W_{EO/IR}^{0.678} : R^2 = 0.879 \quad (52)$$

$$Diameter_{EO/IR} = 30.6 \times W_{EO/IR}^{0.341} : R^2 = 0.975 \quad (53)$$

$$Height = 0.415 \times W_{EO/IR}^{0.325} \times Range/Resolution^{0.0403} : R^2 = 0.918 \quad (54)$$

When it comes to unadjusted function points for an EO-IR sensor, it is assumed the EO-IR sensor is connected to the processor which requires a high-complexity external input and output. It will be transferring images back to the processor which will process them and send commands back to the EO-IR sensor. Since it will wait for some commands from the processor, it will require an external inquiry which outlines what functions the EO-IR sensor wants to perform. Also the interfaces between the processor and the EO-IR sensor require two external interfaces that determine how the two subsystems will communicate with each other. Finally, the

Table 45: Calculating the Total Number of Unadjusted Function Points for a EO-IR Sensor [84]

Type of Component	Complexity of Components			Total
	Low	Average	High	
External Inputs	0 x 3 = 0	0 x 4 = 0	1 x 6 = 6	6
External Outputs	0 x 4 = 0	0 x 5 = 0	1 x 7 = 7	7
External Inquiries	0 x 3 = 0	0 x 4 = 0	1 x 6 = 6	6
Internal Logical Files	0 x 7 = 0	0 x 10 = 0	1 x 15 = 15	15
External Interface Files	0 x 5 = 0	0 x 7 = 0	2 x 10 = 20	20
Total Number of Unadjusted Function Points =				54

EO-IR sensor will require at least one logical file to interpret commands and process its own processes. As a result from using Table 45, an EO-IR sensor is approximated to have 54 unadjusted function points.

EO-IR Sensor Optional Dependent Characteristics: The EO-IR sensor must be able to have a certain resolution over a range. Therefore, the FA²UST Module has the ability to connect the variables with a dependent interface to a mission segment. The operator of the FA²UST Module can define a ratio for which a cruise or loiter segment's altitude is proportional to the range versus resolution: $\frac{Range}{Resolution} = K \times Altitude_{Cruise-Loiter}$.

Navigation - GPS and INS: A UAV has the option of using both a GPS and INS, just a GPS, or just an INS for navigation and control during operations. A GPS and INS subsystems sizing consists of variables and characteristics. The operator of the module has the option to make some of the characteristics dependent on some of the mission segments.

GPS Variables: The variables for a GPS are the error in altitude (*ErrAlt*: ft) and error in horizontal position (*ErrPos*: ft). These variables are based on how a GPS functions. A GPS determines the distance from orbiting satellites and with the knowledge of the satellites' positions, the GPS can hone in on the position of

the UAV. However, there is sometimes error in the calculation. Through the data collection the two variables drive the characteristics of the GPS.

GPS Characteristics: The error in altitude and error in horizontal position variables define the characteristics of a GPS subsystem. The characteristics are the GPS's weight, energy consumption, diameter, and length. These characteristics help define the geometric entity of the GPS which is a cylinder. Equations 55, 56, 57, and 58 displays the GPS's weight, energy consumption, diameter, and length respectively. The weight is in pounds, power is in Watts, diameter in feet, and length in feet.

$$W_{GPS} = 0.400 \times ErrAlt^{2.01} \times ErrPos^{-0.569} : R^2 = 0.953 \quad (55)$$

$$P_{GPS} = 3.14 \times W_{GPS} : R^2 = 0.953 \quad (56)$$

$$Diameter_{GPS} = 0.292 \times W_{GPS}^{0.268} : R^2 = 0.950 \quad (57)$$

$$Length_{GPS} = 0.293 \times W_{GPS}^{0.404} : R^2 = 0.932 \quad (58)$$

When it comes to unadjusted function points for a GPS, it is assumed the GPS is connected to the processor which requires an average-complexity external input and output since it is only relaying position. However, it also must connect with at least three other satellites to hone in on its position. These inputs are only considered average-complexity external inputs since they are only relaying position. Since it will wait for some commands from the processor, it will require an external inquiry which outlines what functions the GPS wants to perform. Also, the GPS will query the three satellites to get positioning information. The interfaces between the processor and the GPS require two external interfaces that determine how the two subsystems will communicate with each other and another external interface to determine how

Table 46: Calculating the Total Number of Unadjusted Function Points for a GPS [84]

Type of Component	Complexity of Components			Total
	Low	Average	High	
External Inputs	0 x 3 = 0	4 x 4 = 16	0 x 6 = 0	16
External Outputs	0 x 4 = 0	1 x 5 = 5	0 x 7 = 0	5
External Inquiries	0 x 3 = 0	4 x 4 = 16	0 x 6 = 0	16
Internal Logical Files	0 x 7 = 0	0 x 10 = 0	1 x 15 = 15	15
External Interface Files	0 x 5 = 0	3 x 7 = 21	0 x 10 = 0	21
Total Number of Unadjusted Function Points =				73

the GPS communicates with the satellites. Finally, the GPS will require at least one logical file to interpret commands and process its own processes. As a result from using Table 46, a GPS is approximated to have 73 unadjusted function points.

INS Variables: The variables for a INS are the maximum acceleration the subsystem can measure (*MaxAcc*: ft/s²), maximum angular rotation the subsystem can measure (*MaxGyro*: rad/s), and error in the acceleration (*ErrAcc*: ft/s²). These variables are based on how an INS functions. An INS uses an accelerometer to measure accelerations in three directions and a gyroscope to measure rotational speeds on the three axes. It then integrates the readings to provide position and orientation measures. A better INS can measure higher accelerations, rotational speeds, and is more sensitive allowing for greater accuracies. From the data collected on existing INS's, regressions for the characteristics were derived.

INS Characteristics: The maximum acceleration the subsystem can measure, maximum angular rotation the subsystem can measure, and error in the acceleration variables define the characteristics of an INS subsystem. The characteristics are the INS's weight, energy consumption, diameter, and length. These characteristics help define the geometric entity of the INS which is a cylinder. Equations 59, 60, 61, and 62 displays the INS's weight, energy consumption, diameter, and length respectively. The weight is in pounds, power is in Watts, diameter in feet, and length in feet.

$$W_{INS} = 1.57 - 4886 \times \frac{ErrAcc}{MaxAcc} + 7.60 \times 10^{-5} \times MaxGyro : R^2 = 0.991 \quad (59)$$

$$P_{INS} = 5.12 \times W_{INS}^{0.625} : R^2 = 0.996 \quad (60)$$

$$Diameter_{INS} = 0.255 \times W_{INS}^{0.319} : R^2 = 0.962 \quad (61)$$

$$Length_{INS} = 0.248 \times W_{INS}^{0.317} : R^2 = 0.965 \quad (62)$$

When it comes to unadjusted function points for an INS, it is assumed the INS is connected to the processor which requires an average-complexity external output since it must receive commands from the processor. It requires two average-complexity external outputs since it must send orientation and acceleration information. Since it will wait for some commands from the processor, it will require an external inquiry which outlines what functions the INS wants to perform. The interfaces between the processor and the GPS require two external interfaces that determine how the two subsystems will communicate with each other. Finally, the INS will require at least two logical file to interpret commands and process its own processes from the gyroscope and accelerometer. As a result from using Table 47, a INS is approximated to have 62 unadjusted function points.

INS Optional Dependent Characteristics: The INS must be able to handle a certain amount of acceleration. Therefore, the FA²UST Module has the ability to connect the variables with a dependent interface to a mission segment. The operator of the FA²UST Module can define a turning or load case to the maximum acceleration the INS can measure: $MaxAcc = Load_{MaxSegment_i}$.

Table 47: Calculating the Total Number of Unadjusted Function Points for a INS [84]

Type of Component	Complexity of Components			Total
	Low	Average	High	
External Inputs	$0 \times 3 = 0$	$1 \times 4 = 4$	$0 \times 6 = 0$	4
External Outputs	$0 \times 4 = 0$	$2 \times 5 = 10$	$0 \times 7 = 0$	10
External Inquiries	$0 \times 3 = 0$	$1 \times 4 = 4$	$0 \times 6 = 0$	4
Internal Logical Files	$0 \times 7 = 0$	$0 \times 10 = 0$	$2 \times 15 = 30$	30
External Interface Files	$0 \times 5 = 0$	$2 \times 7 = 14$	$0 \times 10 = 0$	14
Total Number of Unadjusted Function Points =				62

Communications: All UAVs require a means to communicate back to human operators or between other systems in the operational space. Thus, they require a communications subsystem. However, when looking for existing UAV communication subsystems limited amount of information was provided by the manufacturers. Therefore, the FA²UST Module requires user input to define the communication's characteristics which are weight, energy consumption, diameter, and length. In most cases, the characteristics are set to nominal values 16.5-lbs, 50-Watts, 0.63-feet, and 0.63 feet respective. The nominal values were based on the average values of a couple communications subsystems during the data collection.

When it comes to unadjusted function points for an communications subsystem, it is assumed the communications subsystem is connected to the processor which requires an average-complexity external input and a high-complexity output since it must receive commands from the processor and send data streams back to the processor. The communications subsystem requires three high-complexity external inquiries as it will query the processor, an operations hub, and other systems in the operational space on what functions the UAV should perform. The communications between the processor, an operations hub, and other systems will require three external interface files to determine how information is transfered between entities. Finally, the communications subsystem will require at least one logical file to interpret commands and

Table 48: Calculating the Total Number of Unadjusted Function Points for a Communications Subsystem [84]

Type of Component	Complexity of Components			Total
	Low	Average	High	
External Inputs	$0 \times 3 = 0$	$1 \times 4 = 4$	$0 \times 6 = 0$	4
External Outputs	$0 \times 4 = 0$	$0 \times 5 = 0$	$1 \times 7 = 7$	7
External Inquiries	$0 \times 3 = 0$	$0 \times 4 = 0$	$3 \times 6 = 18$	18
Internal Logical Files	$0 \times 7 = 0$	$0 \times 10 = 0$	$1 \times 15 = 15$	15
External Interface Files	$0 \times 5 = 0$	$0 \times 7 = 0$	$3 \times 10 = 30$	30
Total Number of Unadjusted Function Points =				74

process its own processes. As a result from using Table 48, a INS is approximated to have 74 unadjusted function points.

Processor and Software: The processor and software subsystem is not considered for its weight or power consumption do to the limited public information regarding the computing subsystems when gathering data of the subsystems. Instead the processor and the software subsystem drive the cost of developing the system. The cost can be calculated using Equation 20 found in Section 2.3.4. The user has the option of setting all of the coefficients associated with Equation 20 found in Tables 25 and 26. Any coefficients not set by the user are set to nominal or mid-point values. The FA²UST Module combines the unadjusted function points for each component to predict the total software lines of code. Tables 33 and 34 show the calculations for the determination of unadjusted function points and conversion to software lines of code respectively.

Additional or Miscellaneous Payload: The user of the FA²UST Module can add additional or miscellaneous payload weight to the UAV. This option is supposed to take up any additional weight that is required by the system, such as a weapon system or mission specific payload. The additional payload does not add to any of the aerodynamic forces; it just adds additional weight to the payload, increasing the

overall gross weight of the system.

Propulsion: While reviewing past UAV design, the products used four types of propulsion systems: a piston engine, a turboprop engine, a turbofan engine, or a turbojet engine. The FA²UST Module allows the user to define what type of engine to use. Once the operator selects the type of engine to incorporate in the product, the operator must define specific variables to define the component. From the variables, regressions are used to define the characteristics of the component. Furthermore, during operations of the product, each type of propulsion system interacts with the atmosphere differently. Therefore, traditional relationships were used to define the propulsion performance characteristics of the product.

Piston or Turboprop Engine: The piston or turboprop engine sizing consists of variables, characteristics, and performance characteristics. The performance characteristics determine the product's propulsive performance characteristics allowing the FA²UST Module to determine the operational performance of the system.

Piston or Turboprop Engine Variables: The primary variable that determines the rest of the characteristics of a piston or turboprop engine is the break-horsepower (hp). Everything else can be conceptually explained from this variable. During the collection of data of existing UAVs and their piston engines, regressions for the characteristics of a piston engine were determined from the engine's break-horsepower. The regressions for the turboprop engine's characteristics were taken from Raymer's work [121].

Online Reconfigurable Variables of Piston or Turboprop Engine: The operator of the FA²UST Module has the option to turn on one online reconfigurable interface between the piston turboprop or engine and its propeller. The option is whether to have a variable speed or variable pitch propeller. The major impact of this online reconfigurability is the change in propulsive efficiency (η) of the combined

engine and propeller subsystem. From the overall averages over various advanced ratios ($J = \frac{V}{N \times D}$) and David F. Rogers's work on predicting a propeller's propulsive efficiency [124], a propulsive efficiency of 0.80 was assigned to a variable pitch propeller and a propulsive efficiency of 0.75 was assigned to a variable speed propeller.

Piston and Turboprop Engine Characteristics: The break-horsepower variable defines the characteristics of a piston engine. The characteristics are the piston engine's weight, diameter, and length. These characteristics help define the geometric entity of the piston engine which is a cylinder. Equations 63, 64, and 65 displays the engine's weight, diameter, and height for a piston engine respectively. The weight is in pounds, diameter in feet, and length in feet.

$$W_{eng} = 14.2 \times hp^{0.736} : R^2 = 0.945 \quad (63)$$

$$Diameter_{eng} = 1.72 \times hp^{0.0849} : R^2 = 0.863 \quad (64)$$

$$Length_{eng} = 0.377 \times hp^{0.393} : R^2 = 0.869 \quad (65)$$

The break-horsepower variable defines the characteristics of a turboprop engine. The characteristics are the turboprop engine's weight, diameter, and length. These characteristics help define the geometric entity of the turboprop engine which is a cylinder. Equations 66, 67, and 68 displays the engine's weight, diameter, and height for a turboprop engine respectively. The weight is in pounds, diameter in feet, and length in feet [121].

$$W_{eng} = 2.12 \times hp^{0.803} \quad (66)$$

$$Diameter_{eng} = 0.394 \times hp^{0.373} \quad (67)$$

Table 49: Calculating the Total Number of Unadjusted Function Points for a Piston or Turboprop Engine [84]

Type of Component	Complexity of Components			Total
	Low	Average	High	
External Inputs	0 x 3 = 0	1 x 4 = 4	0 x 6 = 0	4
External Outputs	0 x 4 = 0	1 x 5 = 5	0 x 7 = 0	5
External Inquiries	0 x 3 = 0	1 x 4 = 4	0 x 6 = 0	4
Internal Logical Files	0 x 7 = 0	0 x 10 = 0	1 x 15 = 15	15
External Interface Files	0 x 5 = 0	1 x 7 = 7	0 x 10 = 0	7
Total Number of Unadjusted Function Points =				35

$$Length_{eng} = 0.820 \times hp^{0.12} \quad (68)$$

When it comes to unadjusted function points for a piston or turboprop engine, it is assumed the piston or turboprop engine is connected to the processor which requires an average-complexity external input and output since it must receive commands from the processor and send data streams back to the processor. The piston or turboprop engine requires a average-complexity external inquiry as it will query the processor on what thrust setting the engine should be at. The communications with the processor will require an external interface file to determine how information is transferred between the engine and the processor. Finally, the piston or turboprop engine will require at least one logical file to interpret commands and process its own processes. As a result from using Table 49, a piston or turboprop engine is approximated to have 35 unadjusted function points.

Piston Engine Performance Characteristics: As the UAV uses a piston engine during its operations, there are two performance characteristics that determine its fuel consumption and temperature efficiency. The fuel consumption corresponds to the thrust specific fuel consumption ($TSFC$: 1/s). Equation 69 shows the derivation of $TSFC$ and Equation 70 shows the derivation of temperature efficiency (α) for a piston engine, where V is the airspeed in feet per second and σ is the density ratio

with respect to standard sea level altitude. Equation 69 came from exploring past piston and turboprop engines used on UAVs, and Equation 70 came from Raymer's work on aircraft design [121]. These parameters help determine the overall UAV's performance during operations.

$$TSFC = 2.91 \times 10^{-7} \times V \quad (69)$$

$$\alpha = \sigma - (1 - \sigma)/7.55 \quad (70)$$

Turboprop Engine Performance Characteristics: As the UAV uses a piston engine during its operations, there are two performance characteristics that determine its fuel consumption and temperature efficiency. The fuel consumption corresponds to the thrust specific fuel consumption (*TSFC*: 1/s). Equation 69 shows the derivation of *TSFC* and Equation 71 shows the derivation of temperature efficiency (α) for a turboprop engine, where σ is the density ratio with respect to standard sea level altitude.. Equation 69 came from exploring past piston and turboprop engines used on UAVs, and Equation 71 came from Raymer's work on aircraft design [121]. These parameters help determine the overall UAV's performance during operations.

$$\alpha = \sigma^{0.7} \quad (71)$$

Turbofan or Turbojet Engine The turbofan or turbojet engine sizing consists of variables, characteristics, and performance characteristics. The performance characteristics determine the product's propulsive performance characteristics allowing the FA²UST Module to determine the operational performance of the system.

Turbofan or Turbojet Engine Variables: The primary variable that determines the rest of the characteristics of a turbofan or turbojet engine are the installed

thrust (T : lbs) and bypass ratio (BPR). Everything else can be conceptually explained from these two variables. The regressions for the turbofan engine's characteristics were taken from Raymer's work [121].

Online Reconfigurable Variables of Turbofan or Turbojet Engine: Both the turbofan and turbojet have the option of using thrust vectoring. The FA²UST Module uses thrust vectoring in tandem with simulating a mission segment. The thrust vectoring affects the trim of the aircraft. Therefore, the FA²UST Module trims the aircraft while trying to optimize the flight path of the UAV depending on the mission segment. This process will be explained later when showing how all of the element of the FA²UST Module are tied together.

Turbofan or Turbojet Engine Characteristics: The break-horsepower variable defines the characteristics of a turbofan or turbojet engine. The characteristics are the turbofan or turbojet engine's weight, diameter, and length. These characteristics help define the geometric entity of the turbofan or turbojet engine which is a cylinder. Equations 72, 73, and 74 displays the engine's weight, diameter, and height for a turbofan or turbojet engine respectively. For a turbofan or turbojet engine bypass ratio must be included. The weight is in pounds, diameter in feet, and length in feet [121].

$$W_{eng} = 13.6 \times T^{1.1} \times \exp(-0.045 \times BPR) \quad (72)$$

$$Diameter_{eng} = 0.49 \times T^{0.4} \quad (73)$$

$$Length_{eng} = 0.15 \times T^{0.5} \times \exp(0.04 \times BPR) \quad (74)$$

When it comes to unadjusted function points for a turbofan or turbojet engine, it is assumed the turbofan or turbojet engine is connected to the processor which requires

Table 50: Calculating the Total Number of Unadjusted Function Points for a Turbofan or Turbojet Engine [84]

Type of Component	Complexity of Components			
	Low	Average	High	Total
External Inputs	0 x 3 = 0	1 x 4 = 4	0 or 1 x 6 = 0 or 6	4 or 10
External Outputs	0 x 4 = 0	1 x 5 = 5	0 or 1 x 7 = 0 or 7	5 or 12
External Inquiries	0 x 3 = 0	1 x 4 = 4	0 or 1 x 6 = 0 or 6	4 or 10
Internal Logical Files	0 x 7 = 0	0 x 10 = 0	1 or 2 x 15 = 15 or 30	15 or 30
External Interface Files	0 x 5 = 0	1 or 2 x 7 = 7 or 14	0 x 10 = 0	7 or 14
Total Number of Unadjusted Function Points =				35 or 76

an average-complexity external input for the thrust setting and a high-complexity external input if the engine uses thrust vectoring. It must feedback these settings to the processor as an average-complexity external output for the thrust setting and a high-complexity external output if the engine uses thrust vectoring. The turbofan or turbojet engine requires an average-complexity external inquiry as it will query the processor on what thrust setting the engine should be at and a high-complexity external inquiry if the engine uses thrust vectoring. The communications with the processor will require one or two external interface files (depending on whether the engine uses thrust vectoring) to determine how information is transferred between the engine and the processor. Finally, the turbofan or turbojet engine will require at least one or two (depending if the engine uses thrust vectoring) logical files to interpret commands and process its own processes. As a result from using Table 50, a turbofan or turbojet engine is approximated to have 35 or 76 unadjusted function points.

Turbofan Engine Performance Characteristics: As the UAV uses a turbofan engine during its operations, there are two performance characteristics that determine its fuel consumption and temperature efficiency. The fuel consumption corresponds to the thrust specific fuel consumption ($TSFC$: 1/s). Equation 75 shows the derivation of $TSFC$ and Equation 76 shows the derivation of temperature efficiency (α) for a turbofan engine. Equation 75 and Equation 76 came from Raymer's work on aircraft design [121]. These parameters help determine the overall UAV's performance during

operations.

$$TSFC = 1.11 \times 10^{-4} \times (1 + 0.4 \times M) \quad (75)$$

$$\alpha = 0.369 \times \sigma^{0.7} \times M^{-0.305} \quad (76)$$

Turbojet Engine Performance Characteristics: As the UAV uses a turbojet engine during its operations, there are two performance characteristics that determine its fuel consumption and temperature efficiency. The fuel consumption corresponds to the thrust specific fuel consumption ($TSFC$: 1/s). Equation 77 shows the derivation of $TSFC$ and Equation 78 shows the derivation of temperature efficiency (α) for a turbojet engine. Equation 77 and Equation 78 came from Raymer's work on aircraft design [121]. These parameters help determine the overall UAV's performance during operations.

$$TSFC = \begin{cases} 2.77 \times 10^{-4} + 1.11 \times 10^{-4} \times M & M < 1.1 \\ 4 \times 10^{-4} & \text{otherwise} \end{cases} \quad (77)$$

$$\alpha = \begin{cases} \sigma^{0.7} & M < 1.1 \\ \sigma \times [1 + 1.18 \times (M - 1)] & \text{otherwise} \end{cases} \quad (78)$$

Aerodynamic Surfaces: There are three aerodynamic surfaces used in the FA²UST Module. They are a main wing, horizontal tail, and vertical tail. Each surface has its own pair of control surface that help in the control of the vehicle during flight. A main wing has ailerons and the option for flaps, the horizontal tail has elevators, and the vertical tail has a rudder. All of the aerodynamic surfaces consist of a planform and airfoil geometric entities.

Aerodynamic Surface Variables: The FA²UST Module explains an aerodynamic surface with five variables: area (S), aspect ratio (AR), taper ratio (λ), quarter chord sweep ($\Lambda_{1/4}$), and the type of airfoil used. From these four variables the rest of the characteristics and performance characteristics of the aerodynamic surface can be explained.

Aerodynamic Surface Online Reconfigurable Variables: The main wing in an UAV has four possible online reconfigurable interfaces: ailerons, flaps, variable sweep, and variable pitch. Ailerons are a mandatory feature of the main wing as they help maintain roll and yaw control during flight. Flaps are optional to help the wing gain higher lift coefficients during takeoff and landing. The impact of the flaps are primarily shown as drag and maximum lift contribution during those mission segments. Variable sweep allows the vehicle to reach higher speeds but still have high aerodynamic efficiency at low speed. When variable sweep is activated it impacts the leading edge sweep, quarter chord sweep, and the effective aspect ratio of the wing, as Equations 79 through 81 show. Finally, variable pitch helps the vehicle reach more optimal trim orientations during flight. If variable sweep or pitch are activated for the main wing, then the FA²UST Module treats them as controls and optimizes them alongside other trim settings during the mission simulation.

$$\Lambda_{LE} = \Lambda_{LE_{planform}} + \Lambda_{OnlineRecon} \quad (79)$$

$$\Lambda_{1/4} = \Lambda_{1/4_{planform}} + \Lambda_{OnlineRecon} \quad (80)$$

$$AR = \frac{[b \times \cos(\Lambda_{OnlineRecon})]^2}{S} \quad (81)$$

The horizontal tail uses elevators. They are mandatory to help the UAV control its pitch during flight. The FA²UST Module assumes the impact of them is minimal

when compared to the rest of the aerodynamic forces. The vertical tail uses a rudder. It is mandatory to help the UAV control its pitch and roll during flight. The FA²UST Module assumes the impact of it is minimal when compared to the rest of the aerodynamic forces.

Aerodynamic Surface Characteristics: The FA²UST Module uses form factors to predict the drag of the aircraft when no lift is being generated. For each aerodynamic surface, its form factor is calculated using Equation 82, where t/c is its airfoil's maximum thickness to chord ratio, x/c is its airfoil's location of maximum thickness to chord ratio, $\Lambda_{1/4}$ is its quarter chord sweep, and S is its area. This can be used in the aerodynamic surface's contribution to zero-lift drag.

$$FFS_W = \left(1 + 0.6 \times \frac{t/c}{x/c} + 100 \times t/c^4\right) \times \left[1.34 \times \cos(\Lambda_{1/4})^{0.28}\right] \times 2.02 \times \frac{S}{S_{MainWing}} \quad (82)$$

When it comes to unadjusted function points for an aerodynamic surface, it is assumed the aerodynamic surface is connected to the processor which requires an average-complexity external input and output since it must receive control commands for its ailerons, elevator, or rudder from the processor and send feedback to the processor. The aerodynamic surface requires an average-complexity external inquiry as it will query the processor on what setting the control surface should be at. The communications with the processor will require an external interface file to determine how information is transferred between the aerodynamic surface and the processor. Finally, the aerodynamic surface will require at least one logical file to interpret commands and process its own processes.

However, when it comes to the main wing the variable pitch or sweep online reconfigurable interface require an extra inquiry, logical file, and interface file. As a result from using Table 51, an aerodynamic surface is approximated to have 35

Table 51: Calculating the Total Number of Unadjusted Function Points for a Main Wing [84]

Type of Component	Complexity of Components			Total
	Low	Average	High	
External Inputs	0 x 3 = 0	1 x 4 = 4	0 x 6 = 0	4
External Outputs	0 x 4 = 0	1 x 5 = 5	0 x 7 = 0	5
External Inquiries	0 x 3 = 0	1 x 4 = 4	0, 1, or 2 x 6 = 0, 6, or 12	4, 10, or 16
Internal Logical Files	0 x 7 = 0	0 x 10 = 0	1, 2, or 3 x 15 = 15, 30, or 45	15, 30, or 45
External Interface Files	0 x 5 = 0	1 x 7 = 7	0, 1, or 2 x 10 = 0, 10, or 20	7, 17, or 27
Total Number of Unadjusted Function Points =				35, 66, or 91

unadjusted function points.

Aerodynamic Surface Performance Characteristics: The primary performance characteristics of an aerodynamic surface are its contributions to drag and lift. The form factor analysis to predict the aerodynamic surface's contribution to zero lift drag requires the ability to determine if the flow over its surface is laminar or turbulent. Therefore, the Reynold's number laminar-turbulent limit is calculated in Equation 83. This is compared against the actual Reynold's number the surface is experiencing (Equation 84) to calculate its coefficient of friction (Equation 85).

$$Re_{C_W} = 38.21 \times \left(\frac{\bar{c} \times 10^5}{0.17} \right)^{1.053} \quad (83)$$

$$Re_W = \frac{\rho \times V \times \bar{c}}{\mu} \quad (84)$$

$$C_{f_W} = \begin{cases} \frac{0.455}{\log_{10}(Re_W)^{2.58 \times (1 + 0.144 \times M^2)^{0.65}}} & Re_W > Re_{C_W} \\ \frac{1.328}{\sqrt{Re_W}} & otherwise \end{cases} \quad (85)$$

The FA²UST Module considered drag divergence at transonic speeds. Therefore, it includes an approximation of the divergence though Equation 86. The coefficient K_{DD} considers the impact the sweep, aspect ratio, and taper of the surface have on

the divergence. The module then multiplies this factor to all contributions of drag to get an accurate approximation of drag in the transonic regime.

$$K_{DD} = \left| \frac{1}{\sqrt{1 - 1.4 \times \frac{M \times i}{\lambda \times AR} - [M \times \cos(\Lambda_{LE})]^2}} \right| \quad (86)$$

The final calculation to predict the aerodynamic surface's contribution to zero lift drag is found in Equation 87. It multiplies the coefficient of friction with the form factor, drag divergence coefficient, and a contribution of the Mach number. All aerodynamic surfaces contribute this to the drag.

$$C_{D_{0W}} = C_{fw} \times FFS_W \times K_{DD} \times M^{0.18} \quad (87)$$

The Main Wing's Induced Drag Contributions: The FA²UST Module only considers the main wing's induced drag. The module first calculates the Oswald Efficiency Factor using Equations 88 through 91.

$$e_1 = 1 + 0.12 \times M^2 \quad (88)$$

$$e_2 = \frac{0.142 + 0.005 \times [1 + 1.5 \times (\lambda - 0.6)^2] \times AR \times (10 \times t/c)^{0.33}}{\cos^2(\Lambda_{1/4})} \quad (89)$$

$$e_3 = \frac{0.4}{(4 + AR)^{0.8}} \quad (90)$$

$$e = \frac{1}{e_1 \times (1 + e_2 + e_3)} \quad (91)$$

Once the Oswald Efficiency Factor is calculated the linear and quadratic terms of the main wing's induced drag can be calculated. The linear and quadratic coefficients are found in Equations 92 and 93. The zero lift drag and induced drag are later combined in Equation 116.

$$k_1 = \frac{1}{\pi \times AR \times e} \quad (92)$$

$$k_2 = -2 \times k_1 \times C_{l_{min}} \quad (93)$$

The flaps impact the maximum lift and drag produced by the wing. Equation 94 shows the formulation of $C_{L_{max}}$ when flaps are and are not engaged. The additional zero-lift drag incurred by the wing is assumed to be 0.01 [121].

$$C_{L_{max}} = \begin{cases} 1.9 \times \cos(\Lambda_{1/4_{MainWing}}) & \text{Flaps} \\ 1.6 \times \cos(\Lambda_{1/4_{MainWing}}) & \text{otherwise} \end{cases} \quad (94)$$

Fuselage: The FA²UST Module usually considers the fuselage as the platform of the UAV since it has the most immediate connections with the other components in the system. The FA²UST Module represents the fuselage conceptually as a cylindrical geometric entity.

Fuselage Variables: Due to the fuselage's cylindrical shape the fuselage can be summarized by two variables: its length (*Length*: feet) and diameter (*Diameter*: feet). The two variables define its cylindrical geometric entity and can help define its performance characteristics.

Fuselage Characteristics: The FA²UST Module uses form factors to predict the drag of the aircraft when no lift is being generated. For a fuselage, its form factor is calculated using Equation 95. This can be used in the aerodynamic surface's contribution to zero-lift drag.

$$FFS_F = \left[1 + 60 \left(\frac{Diameter}{Length} \right)^3 + \frac{1}{400} \frac{Length}{Diameter} \right] \times \frac{\pi \times Diameter \times Length}{S_{MainWing}} \quad (95)$$

Fuselage Dependent Characteristics: The FA²UST Module automatically sets the fuselage's diameter to the maximum value of all the subsystems' and engine's diameter. This ensures all of the components that are placed inside the fuselage have enough space. Therefore, the diameter of the fuselage can be represented by Equation 96.

$$Diameter = \max [Diameter_{Component_i}] \quad (96)$$

The operator of the FA²UST Module has two options when it comes to setting the fuselage's length. Either the operator can manually set the length to a specific value, or the module can use the historical regression, found in Equation 97, to define the length of the fuselage.

$$Length = 1.39 \times b_{MainWing}^{0.757} \quad (97)$$

Fuselage Performance Characteristics: The primary performance characteristics of an fuselage are its contributions to drag. The form factor analysis to predict the fuselage's contribution to zero lift drag requires the ability to determine if the flow over its surface is laminar or turbulent. Therefore, the Reynold's number laminar-turbulent limit is calculated in Equation 98. This is compared against the actual Reynold's number the fuselage is experiencing (Equation 99) to calculate its coefficient of friction (Equation 85).

$$Re_{C_F} = 38.21 \times \left(\frac{Length \times 10^5}{0.17} \right)^{1.053} \quad (98)$$

$$Re_F = \frac{\rho \times V \times Length}{\mu} \quad (99)$$

$$C_{f_F} = \begin{cases} \frac{0.455}{[\log_{10}(Re_F)]^{2.58} \times [1 + 0.144 \times M^2]^{0.65}} & Re_F > Re_{C_F} \\ \frac{1.328}{\sqrt{Re_F}} & otherwise \end{cases} \quad (100)$$

The final calculation to predict the fuselage's contribution to zero lift drag is found in Equation 101. It multiplies the coefficient of friction with the form factor, drag divergence coefficient, and a contribution of the Mach number.

$$C_{D_{0_F}} = C_{f_F} \times FFS_F \times K_{DD} \times M^{0.18} \quad (101)$$

4.2.1.3 Medium: Atmospheric Properties

During operations, a UAV must interact with the atmosphere. The atmosphere acts as a control volume that flows around the aircraft. Therefore in the performance analysis, there are certain properties or characteristics that are essential.

Atmospheric Medium Variables: The variables that explain the properties of the atmospheric medium are altitude (h : feet) and speed (V : ft/s). Using the variables, properties such as air temperature, pressure, density, viscosity, dynamic pressure, speed of sound, Mach number, and acceleration from gravity [65].

Atmospheric Medium Characteristics or Properties: The air temperature (*Temperature*: R) uses temperature altitude (θ) and standard sea-level air temperature (518.4° R) in its calculation. The temperature altitude is a function of altitude. The function depends on what layer of the atmosphere the UAV is operating. Equation 102 and Equation 103 show the formulation of temperature altitude and air temperature respectively.

$$\theta = \begin{cases} 1 - \frac{h}{145442} & h < 36089 \\ 0.752 & h \geq 36089 \& h < 65617 \\ 0.682 + \frac{h}{945374} & h \geq 65617 \& h < 104987 \\ 0.483 + \frac{h}{337634} & h \geq 104987 \& h < 154199 \\ 0.939 & h \geq 154199 \& h < 167323 \\ 1.43 - \frac{h}{337634} & h \geq 167323 \end{cases} \quad (102)$$

$$Temperature = 518.4 \times \theta \quad (103)$$

The air pressure (*Pressure*: lbs/ft²) uses pressure altitude (δ) and standard sea-level air pressure (2116.8 lbs/ft²) in its calculation. The pressure altitude is a function of altitude. The function depends on what layer of the atmosphere the UAV is operating. Equation 104 and Equation 105 show the formulation of pressure altitude and air pressure respectively.

$$\delta = \begin{cases} \left(1 - \frac{h}{145442}\right)^{5.26} & h < 36089 \\ 0.223 \times \exp\left(\frac{36089-h}{20806}\right) & h \geq 36089 \& h < 65617 \\ \left(0.989 + \frac{h}{652600}\right)^{-34.2} & h \geq 65617 \& h < 104987 \\ \left(0.898 + \frac{h}{181373}\right)^{-12.2} & h \geq 104987 \& h < 154199 \\ 0.00109 \times \exp\left(\frac{h-154200}{-25992}\right) & h \geq 154199 \& h < 167323 \\ \left(0.838 - \frac{h}{577922}\right)^{12.2} & h \geq 167323 \end{cases} \quad (104)$$

$$Pressure = 2116.8 \times \delta \quad (105)$$

The air density (ρ : slug/ft³) uses density altitude (σ) and standard sea-level air density (0.00238 slug/ft³) in its calculation. The density altitude is a function of altitude. The function depends on what layer of the atmosphere the UAV is operating. Equation 104 and Equation 105 show the formulation of density altitude and air density respectively.

$$\sigma = \begin{cases} \left(1 - \frac{h}{145442}\right)^{4.255876} & h < 36089 \\ 0.297076 \times \exp\left(\frac{36089 - alt Ft}{20806}\right) & h \geq 36089 \& h < 65617 \\ \left(0.978261 + \frac{h}{659515}\right)^{-35.16319} & h \geq 65617 \& h < 104987 \\ \left(0.857003 + \frac{h}{190115}\right)^{-13.20114} & h \geq 104987 \& h < 154199 \\ 0.00116533 \times \exp\left(\frac{alt Ft - 154200}{-25992}\right) & h \geq 154199 \& h < 167323 \\ \left(0.79899 - \frac{h}{606330}\right)^{11.20114} & h \geq 167323 \end{cases} \quad (106)$$

$$\rho = 0.00238 \times \sigma \quad (107)$$

The air viscosity (μ : lb-sec/ft²) uses air temperature (*Temperature*) and standard sea-level air viscosity (3.62×10^{-7} lb-sec/ft²) in its calculation. Equation 108 shows the formulation of air viscosity.

$$\mu = 3.62 \times 10^{-7} \times \left(\frac{Temperature}{518.7}\right)^{1.5} \times \frac{717.42}{Temperature + 198.72} \quad (108)$$

The air dynamic pressure (*DynamicPressure*: lbs/ft²) uses air density (ρ) and the speed (V : ft/s) in its calculation. The dynamic pressure is often used to calculate the forces on the UAV during its operations. Equation 109 show the formulation of air dynamic pressure.

$$DynamicPressure = \frac{\rho \times V^2}{2} \quad (109)$$

The speed of sound at a given altitude (*SoS*: ft/s) uses air temperature (*Temperature*: R) in its calculation. After calculating the speed of sound, the Mach number (M) of the aircraft can be calculated using the aircraft's speed (V : ft/s). Equation 110 and Equation 111 show the formulation of speed of sound at a given altitude and Mach number respectively.

$$SoS = \sqrt{2404 \times Temperature} \quad (110)$$

$$M = \frac{V}{SoS} \quad (111)$$

The final property of the medium is the acceleration due to gravity (g : ft/s²). It is a function of the altitude (h : feet), the radius of the earth (20900000 feet), and the sea-level standard acceleration due to gravity (32.2 ft/s²). Equation 112 shows the derivation of the acceleration due to gravity.

$$g = 32.2 \times \left(\frac{20900000}{20900000 + h} \right)^2 \quad (112)$$

4.2.1.4 UAV Configurations

The FA²UST Module uses configurations to determine the physical connections and arrangements of the components in a UAV system. As a result, the module incorporates four types of configurations into the analysis. These configurations are:

- A traditional tube-body, wing, horizontal tail, and vertical tail (Figure 65)
- A tube-body, wing, and horizontal tail (Figure 66)
- A tube-body, wing, and vertical tail (Figure 67)
- A flying wing (Figure 68)

All of the configurations have optional combinations of subsystems that can be incorporated into a design, depending on the desired capabilities of the system.

Figures 65 through 68 show the arrangement and physical connections between the components in each configuration. The arrangement of the components help with the product architecture index calculations and trim optimization during the mission simulations.

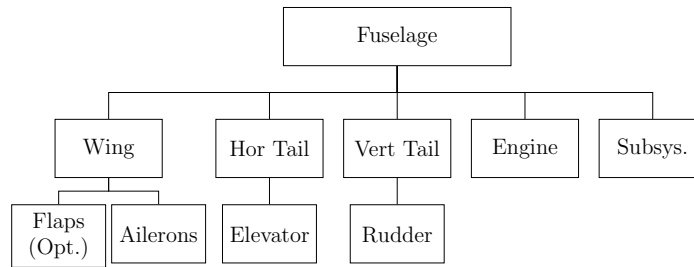


Figure 65: UAV Tree Diagram Outlining Physical Connections of Components and Subsystems for a Traditional Tube-Body, Wing, Horizontal Tail, and Vertical Tail Configuration

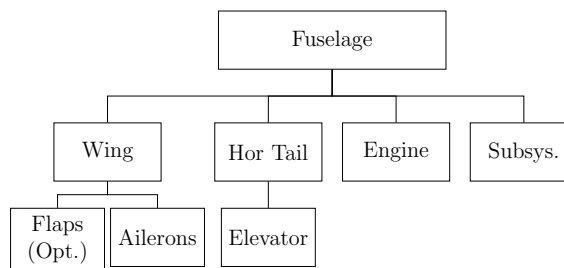


Figure 66: UAV Tree Diagram Outlining Physical Connections of Components and Subsystems for a Traditional Tube-Body, Wing, and Horizontal Tail Configuration

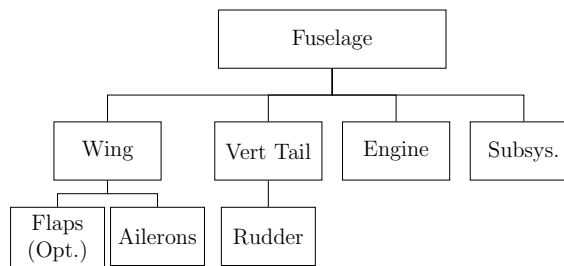


Figure 67: UAV Tree Diagram Outlining Physical Connections of Components and Subsystems for a Tube-Body, Wing, and Vertical Tail Configuration

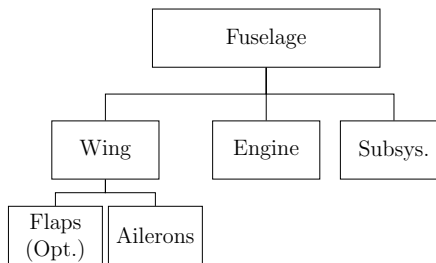


Figure 68: UAV Tree Diagram Outlining Physical Connections of Components and Subsystems for a Flying Wing Configuration

4.2.1.5 Product or Design: An Integrated UAV System Architecture

An integrated product or design is the composition of physical entities and the configuration rules that outline their arrangement. Once the FA²UST Module establishes the product, there are a couple key characteristics that must be defined in the UAV product or design.

UAV Product Characteristics: The characteristics that define the integrated product and design are the empty weight, payload weight, fuel weight, and the takeoff gross weight. The fuel weight is determined during the sizing process within the FA²UST Module. The takeoff gross weight is the sum of the empty weight, payload weight, and fuel weight, as Equation 113 shows.

$$W_{TO} = W_{Empty} + W_{Payload} + W_{Fuel} \quad (113)$$

The empty weight can be defined by a couple of the variables and characteristics of the components within the product. Equation 114 shows the formulation of the empty weight (W_{Empty} : lbs) of a UAV, where $VarSweep$ is the variable that determines if the UAV has an online reconfigurable wing, $Flaps$ is the variable that determines if the UAV has flaps, $S_{MainWing}$ is the main wing's area (ft²), $AR_{MainWing}$ is the main wing's aspect ratio, hp is the engine's horsepower, T is the engine's thrust (lbs), and $Length_{UAV}$ is the UAV's length (ft). The UAV's length is the fuselage's length for all configurations that have a fuselage, otherwise it is the distance from the main wing's front of the root chord to the back of the tip chord.

$$W_{Empty} = 2.20 \times \begin{cases} \exp \{0.873 + 0.712 \times \ln [(1 + 0.17 \times VarSweep) \times (1 + 0.11 \times Flaps) \times \frac{\sqrt{S_{MainWing} \times AR_{MainWing}}}{3.28}] + 0.473 \times \ln (hp) + 0.617 \times \ln \left(\frac{Length_{UAV}}{3.28} \right) \} & \begin{matrix} Engine = \\ Piston|| \\ Turboprop \end{matrix} \\ \exp \{4.58 + 0.434 \times \ln [(1 + 0.17 \times VarSweep) \times (1 + 0.11 \times Flaps) \times \frac{\sqrt{S_{MainWing} \times AR_{MainWing}}}{3.28}] + 0.0446 \times \ln \left(\frac{Length_{UAV}}{3.28} \right) + 0.710 \times \ln \left(\frac{T}{224.8} \right) \} & R^2 = 0.991 \quad otherwise \end{cases} \quad (114)$$

The payload weight consists of the weight of all subsystems included in the UAV product plus the additional payload defined by the module's operator. Equation 115 shows the formulation of the payload weight ($W_{Payload}$: lbs) for a UAV product.

$$W_{Payload} = W_{AdditionalPayload} + \sum_{i=1}^N W_{Subsystem_i} \quad (115)$$

As stated earlier, the sizing process in the module determines the fuel weight by simulating a mission. An initial guess for the takeoff gross weight is provided to the process which iterates until the takeoff gross weight and the fuel weight converges. An initial guess of $1.5 \times (W_{Empty} + W_{Payload})$ is given to the process.

A key element of the process is determining the drag the UAV experiences during each mission segment. First the process optimizes the trim orientation that provides the best conditions for flight relative to each mission segment. During this process, the necessary lift coefficient (C_L) is calculated which is fed to the drag coefficient calculation shown in Equation 116. The trim setting influences the zero-lift drag coefficients, the linear, and quadratic terms of the induced drag.

$$C_D = \sum_{i=1}^N (C_{D0_i}) + k_2 \times C_L + k_1 \times C_L^2 \quad (116)$$

4.2.1.6 FA²UST Module UAV Requirements

The FA²UST Module uses requirements to add constraints, determine performance, or determine the cost of a UAV product. These requirements include mission segments, subsystem cooling analysis, volume sizing, and cost modeling. The mission segments are considered ordinal requirements and require some pre-determination by the module's operator on the order of which the mission simulation should go through each. The subsystem cooling analysis adds a constraint to ensure the UAV has enough fuel to keep the subsystems within the UAV from overheating. The volume sizing adds a constraint to determine whether the vehicle has enough room to hold all subsystems.

Finally, the cost modeling predicts the development and production costs required for an entire UAV product line.

Ordinal Requirements: Mission Analysis and Mission Segments: The mission segments within the FA²UST Module are used to size the vehicle. Each has its own set of variables that define it. The sizing process determines the weight of the fuel by determining the weight fraction (β) from each mission segment. The weight fraction is the ratio of vehicle's weight at the conclusion of the mission segment to the vehicle's weight at the start of the mission segment ($\frac{W_f}{W_i}$). Equation 117 shows how each segment calculates its contribution to the overall mission's weight fraction, where $TSFC$ is the engine's thrust specific fuel consumption (1/s), T_R is the segment's required thrust, dt is the time span of the segment, and W_{TO} is the takeoff gross weight of the UAV. Each segment's span can be broken up into smaller segments, generating greater accuracy. Hence the use of dt rather than overall time.

$$\frac{W_f}{W_i} = \exp \left(\frac{-TSFC \times T_R \times dt}{\beta W_{TO}} \right) \quad (117)$$

Some of the engines produce power rather than thrust. Equation 118 shows how the power for a piston or turboprop engine is converted into thrust, where T is thrust (lbs), η_{Prop} is the propulsive efficiency of the propeller, P is power (lbs-ft/s), and V is the UAV's velocity (ft/s).

$$T = \frac{\eta_{Prop} \times P}{V} \quad (118)$$

Multiplying all of the mission segments' weight fractions together provides the overall weight fraction for the mission. The weight fraction then can be used to determine the takeoff gross weight of the vehicle and its fuel weight, as shown in Equation 119. It is assumed three percent of the fuel is lost in the vehicle so the takeoff gross weight is 103% of the empty weight and payload weight divided by the

weight fraction.

$$W_{TO} = 1.03 \times \frac{W_{Empty} + W_{Payload}}{\beta} \quad (119)$$

The mission segments can also add constraints to the design. Most often, the UAV must be able to generate enough lift and have enough thrust to overcome the drag of the segment. The coefficient of lift required during the mission segment is subtracted by the maximum lift coefficient available of the wing. Equation 120 shows the formulation of this constraint. It is important to notice all constraints are normalized to prevent bias toward any one.

$$g_{C_L} = \frac{C_L}{C_{L_{max}}} - 1 \leq 0 \quad (120)$$

Another common constraint for a mission segment is the assurance the UAV has enough thrust to overcome the drag and excess power required of the segment. The module assumes the aircraft is flying at steady level flight and can use Mattingly's equation to calculate the thrust required of the segment. The traditional Mattingly equation was modified to include the use of thrust vectoring. However, the thrust required (T_R : lbs) must be solved for iteratively if the engine has thrust vectoring. Equation 121 shows the Mattingly Equation used in the FA²UST Module, where T_R is the thrust required (lbs), $\delta_{ThrustVector}$ is the trim setting for thrust vectoring (rad), β is the weight fraction during the segment, α is the engine's thermal efficiency, k_1 is the main wing's coefficient of its quadratic contribution to the induced drag, n is the load factor the aircraft is experiencing, W_{TO} is the UAV's takeoff gross weight (lbs), k_2 is the main wing's coefficient of its linear contribution to the induced drag, C_{D_0} is the UAV's zero-lift drag coefficient, Q is the dynamic pressure the UAV is experiencing (lbs/ft²), $S_{MainWing}$ is the main wing's area (ft²), V is the UAV's velocity (ft/s), h is the UAV's altitude (ft), and g is the acceleration due to gravity the aircraft is experiencing.

$$T_R = \begin{cases} \frac{1}{\cos(\delta_{ThrustVector})} \frac{\beta}{\alpha} \{k_1 [n\beta W_{TO} - T_R \sin(\delta_{ThrustVector})]^2 + \\ k_2 [n\beta W_{TO} - T_R \sin(\delta_{ThrustVector})] + \\ C_{D_0} Q S_{MainWing} + \frac{W_{TO}}{V} \frac{d}{dt} \left(h + \frac{V^2}{2g} \right) \} \end{cases} \quad (121)$$

The segment will often optimize the trim settings with respect to optimal flight conditions for that segment when the vehicle has online reconfigurable components. Once the trim setting is set, Equation 122 shows the formulation of the thrust required constraint, where T_R is the thrust required (lbs) and T_A is the thrust available from the engine (lbs). It is important to notice all constraints are normalized to prevent bias toward any one.

$$g_{T_R} = \frac{T_R}{T_A} - 1 \leq 0 \quad (122)$$

Each segment can have its own way of contributing to the weight sizing. Therefore, some of the constraints it adds to the design are not the traditional lift and thrust required constraints.

Warm-Up Segment: The warm-up segment uses one variable to define itself: time. The segment adds no constraints to the design, but does contribute to the weight sizing of the vehicle. During this segment, the engine is operating at maximum power or thrust for the time specified to calculate the weight fraction using Equation 117.

Takeoff Segment: The takeoff segment uses one variable to define itself: the desired runway length. It simulates the acceleration during takeoff and determines if the aircraft can achieve takeoff speed within the given runway length (s_{desTO}). Equation 123 shows the formulation of required runway length (s_{reqTO}), where k_{TO} is the takeoff speed to stall speed ratio (1.2).

$$s_{reqTO} = \frac{k_{TO} \times (\beta W_{TO})^2}{\rho \times g \times C_{L_{max}} \times T_A \times S_{MainWing}} \quad (123)$$

Equation 124 shows the formulation of the takeoff constraint. It is important to notice all constraints are normalized to prevent bias toward any one.

$$g_{TO} = \frac{s_{reqTO}}{s_{desTO}} - 1 \leq 0 \quad (124)$$

The module uses a simplified approach to calculating the weight fraction of the takeoff segment. It makes the assumption of using $\frac{7}{10}$'s the takeoff speed in Equation 117 to calculate the segment's weight fraction. Since the takeoff segment occurs on the ground, no lift or thrust constraints are added.

Climb Segment: The climb segment is the more basic of the two options for simulating a UAV's climb. It uses three variables to define itself: the speed, the starting altitude, and ending altitude of the UAV. It makes the aircraft maintain a constant speed during the climb, though the rate of climb may change. The segment enforces this rule by operating at the engine's maximum power or thrust. Then, it trims the aircraft by varying the angle of climb and the online reconfigurable elements of the UAV so the aircraft can achieve its maximum rate of climb. Therefore, the optimization problem for the climb segment can be found in Equation 125. Equation 121 is used to trim the aircraft and calculate the thrust required (T_R).

$$\max \left(\frac{dh}{dt} \right) w.r.t. \rightarrow T_R = T_A \quad (125)$$

If the maximum rate of climb of the UAV becomes less than zero before the UAV reaches its final altitude, the module adds a constraint to the design. The constraint uses the altitude of the UAV when the maximum rate of climb becomes zero ($h_{R/C=0}$) and compares it to the desired ending altitude (h_{end}). Equation 126

shows the formulation of this constraint. It is important to notice all constraints are normalized to prevent bias toward any one.

$$g_{R/C} = \frac{h_{R/C=0}}{h_{end}} - 1 \leq 0 \quad (126)$$

Finally, the segment uses Equation 117 to calculate its contribution to the mission's overall weight fraction and Equation 120 determines the lift constraint for the segment.

Maximum Rate of Climb Segment: The maximum rate of climb segment is the more complicated of the two options for simulating a UAV's climb. It uses two variables to define itself: the starting altitude and ending altitude of the UAV. The segment enforces this rule by operating at the engine's maximum power or thrust. Then, it trims the aircraft by varying the aircraft's speed, angle of climb, and the online reconfigurable elements of the UAV so the aircraft can achieve its maximum rate of climb. Though similar to the climb segment it adds the aircraft's speed into the optimization problem. Therefore, the optimization problem for the maximum rate of climb segment can be found in Equation 125. Equation 121 is used to trim the aircraft and calculate the thrust required (T_R). If the maximum rate of climb of the UAV becomes less than zero before the UAV reaches its final altitude, the module adds a constraint to the design. The constraint uses the altitude of the UAV when the maximum rate of climb becomes zero ($h_{R/C=0}$) and compares it to the desired ending altitude (h_{end}). Equation 126 shows the formulation of this constraint. Finally, the segment uses Equation 117 to calculate its contribution to the mission's overall weight fraction and Equation 120 determines the lift constraint for the segment.

Cruise Segment: The cruise segment is one of the more basic segment's in the module. It uses four variables to define itself: the range, the speed, the starting altitude, and ending altitude of the UAV. It makes the aircraft maintain constant

rate of climb and speed. It trims the aircraft by varying the online reconfigurable elements of the UAV so the aircraft can achieve its maximum range per fuel burned. Therefore, the optimization problem for the cruise segment can be found in Equation 127. Equation 121 is used to trim the aircraft and calculate the thrust required (T_R).

$$\max \left(\frac{V}{\dot{m}} \right) \quad (127)$$

Finally, the segment uses Equation 117 to calculate its contribution to the mission's overall weight fraction, Equation 120 determines the lift constraint for the segment, and Equation 122 determines the thrust constraint for the segment.

Best Mach Number Cruise Segment The best Mach number cruise segment uses two variables to define itself: the range, the speed and the altitude of the UAV. It assumes the aircraft will maintain a certain altitude. It trims the aircraft by varying the speed and the online reconfigurable elements of the UAV so the aircraft can achieve its maximum range per fuel burned. Therefore, the optimization problem for the maximum rate of best Mach number cruise segment can be found in Equation 127. Equation 121 is used to trim the aircraft and calculate the thrust required (T_R). Finally, the segment uses Equation 117 to calculate its contribution to the mission's overall weight fraction, Equation 120 determines the lift constraint for the segment, and Equation 122 determines the thrust constraint for the segment.

Best Mach Number and Altitude Cruise Segment The best Mach number and altitude cruise segment uses one variable to define itself: the range. It trims the aircraft by varying the speed, the altitude, the online reconfigurable elements of the UAV so the aircraft can achieve its maximum range per fuel burned. By varying the altitude throughout the segment, this impacts the rate of climb which is considered during the segment. Therefore, the optimization problem for the maximum rate of best Mach number cruise and altitude segment can be found in Equation 127.

Equation 121 is used to trim the aircraft and calculate the thrust required (T_R). Finally, the segment uses Equation 117 to calculate its contribution to the mission's overall weight fraction, Equation 120 determines the lift constraint for the segment, and Equation 122 determines the thrust constraint for the segment.

Acceleration Segment The acceleration segment uses four variables to define itself: the altitude, the start speed, the end speed, and the time to complete the acceleration. It trims the aircraft by varying the online reconfigurable elements of the UAV so the aircraft can achieve its maximum endurance during this segment, because the acceleration, speeds, and altitude are set. Therefore, the segment tries to make the segment as efficient as possible by varying the online reconfigurable elements of the UAV. Therefore, the optimization problem for the acceleration segment can be found in Equation 128. Equation 121 is used to trim the aircraft and calculate the thrust required (T_R).

$$\max \left(\frac{1}{\dot{m}} \right) \quad (128)$$

Finally, the segment uses Equation 117 to calculate its contribution to the mission's overall weight fraction, Equation 120 determines the lift constraint for the segment, and Equation 122 determines the thrust constraint for the segment.

Maximum Rate of Acceleration Segment The maximum rate of acceleration segment uses three variables to define itself: the altitude, the start speed, and the end speed. The segment sets the engine's power or thrust to its maximum setting, and it trims the aircraft by varying the rate of acceleration and the online reconfigurable elements of the UAV so the aircraft can achieve its maximum rate of acceleration during this segment. Therefore, the optimization problem for the maximum rate of acceleration segment can be found in Equation 129. Equation 121 is used to trim the aircraft and calculate the thrust required (T_R).

$$\max \left(\frac{dV}{dt} \right) w.r.t. \rightarrow T_R = T_A \quad (129)$$

If the maximum rate of acceleration of the UAV becomes less than zero before the UAV reaches its final speed, the module adds a constraint to the design. The constraint uses the speed of the UAV when the maximum rate of acceleration becomes zero ($V_{R/A=0}$) and compares it to the desired ending speed (V_{end}). Equation 130 shows the formulation of this constraint. It is important to notice all constraints are normalized to prevent bias toward any one.

$$g_{R/A} = \frac{V_{R/A=0}}{V_{end}} - 1 \leq 0 \quad (130)$$

Finally, the segment uses Equation 117 to calculate its contribution to the mission's overall weight fraction, and Equation 120 determines the lift constraint for the segment.

Loiter or Best Endurance Segment The loiter or best endurance segment uses two variables to define itself: the time elapsed during the segment and the altitude of the UAV. It makes the aircraft maintain constant altitude while trimming the aircraft by varying its speed and its online reconfigurable elements so the aircraft can achieve its maximum endurance. Therefore, the optimization problem for the loiter or best endurance segment can be found in Equation 128. Equation 121 is used to trim the aircraft and calculate the thrust required (T_R). Finally, the segment uses Equation 117 to calculate its contribution to the mission's overall weight fraction, Equation 120 determines the lift constraint for the segment, and Equation 122 determines the thrust constraint for the segment.

Maximum Speed Capability Segment The maximum speed capability segment does not contribute to the vehicle's weight sizing. Instead it provides additional

constraints that determines whether the vehicle can actually meet a speed at a specific point in the mission. The segment is defined by two variables: the desired speed and the altitude of the UAV. It trims the aircraft by varying the online reconfigurable elements of the UAV so the aircraft can achieve its maximum endurance during this segment, because the speed and altitude are already set and the segment should be made as efficient as possible. Equation 121 is used to trim the aircraft and calculate the thrust required (T_R). From the trim setting, Equation 120 determines the lift constraint for the segment, and Equation 122 determines the thrust constraint for the segment.

Landing Segment The landing segment does not contribute to the vehicle's weight sizing. Instead it provides an additional constraints that determines whether the vehicle can land within a given runway length. The segment is defined by one variable: the desired runway length ($s_{desland}$). Equation 131 shows the formulation of required runway length ($s_{reqland}$), where k_L is the landing speed to stall speed ratio (1.15), t_{roll} is the time of the initial roll (3 sec), and μ_{brake} is the additional friction from braking.

$$s_{reqland} = k_L \times t_{roll} \times \sqrt{\frac{2 \times \beta \times W_{TO}}{\rho \times S_{MainWing} \times C_{L_{max}}}} + \frac{k_L^2 \times \beta \times W_{TO}}{g \times \rho \times C_{L_{max}} \times \mu_{brake} \times S_{MainWing}} \quad (131)$$

Finally, Equation 132 shows the formulation of the landing constraint. It is important to notice all constraints are normalized to prevent bias toward any one.

$$g_{land} = \frac{s_{reqland}}{s_{desland}} - 1 \leq 0 \quad (132)$$

Reserve Segment The reserve segment uses one variable to define itself: the time elapsed during the segment. It makes the aircraft maintain constant altitude at standard sea level while trimming the aircraft by varying its speed and its online

reconfigurable elements so the aircraft can achieve its maximum endurance. Therefore, the optimization problem for the reserve segment can be found in Equation 128. Finally, the segment uses Equation 117 to calculate its contribution to the mission's overall weight fraction, Equation 120 determines the lift constraint for the segment, and Equation 122 determines the thrust constraint for the segment.

Subsystem Cooling The operator of the FA²UST Module has the option to add a cooling constraint to the design of UAVs. Since the subsystems often run the risk of overheating, a fuel cooling model was added. It makes the assumption that fuel is pumped around the subsystems to extract the heat. It makes the assumption that 15% of the power consumed by a subsystem is transformed into heat [49]. This heat then increases the overall temperature of the fuel. The fuel must stay below its evaporation point (797.67°R) during operations. Using this information, the temperature of the fuel is simulated during the mission analysis using Equation 133 [49].

$$T(t) = T_{t=0} - \frac{\dot{Q}_h}{\dot{m} \times c_{p_{fuel}}} \ln \left(1 - \frac{\dot{m} \times t}{m_{t=0}} \right) \quad (133)$$

Equation 134 shows the final formulation of the cooling constraint after the mission simulation has been completed.

$$g_{cool} = \frac{T_f}{T_{evap}} - 1 \leq 0 \quad (134)$$

Volume Sizing Volume sizing is important in designing an aircraft that heavily relies on subsystems [146], and ensures the vehicle can fit all of its subcomponents in the vehicle. The FA²UST Module conducts volume sizing with respect to subsystem location constraints and assumptions were implemented to the overall design of the UAV.

The radar must be in the nose of the aircraft. This constraint fixed the radar compartment's location. Radar tends to be a rectangular box where all of the processors

are kept and a circular antenna facing forward. Therefore for basic sizing principles, the radar can be considered to be a cylinder with diameter of the antenna and length equal to the length of the box. The radar must be in the nose of the aircraft in order to expose its antenna forward and the synthetic aperture radar (SAR) down facing targets on the ground.

An EO is a cylinder with a sphere camera housing that can rotate to get images in 3 directions. To fit the EO in the aircraft it must have room to fit the cylinder into the belly of the aircraft which is where the subsystem can be connected to the aircraft. The EO must be located near the front of the aircraft and protruding from the bottom of the aircraft so the sensor to view important targets on the ground. It must be far from the engine since vibrations could impair the pictures taken by the EO. Therefore, the EO was place just behind the radar and sticking out the bottom of the aircraft.

There are two types of GPS or INS. The GPS tends to be a bit larger but both tend to be constructed as cylinders housing their sensors and hardware. The GPS needs to be at the top of the fuselage so that it can communicate with the satellites orbiting in space. The INS does not have this restriction, but both must be far from the engine in order to reduce thermal and vibration noise which can corrupt the signals of both sensors. Therefore, the navigation compartment was placed above the EO and behind the radar. It was assumed to be shaped as cylinder.

Communications are essential to UAV operations and need to have sufficient range in order to communicate with the controller a far distance away. As seen in the database section, weight and size is directly related to the range of the subsystem. This means the communications tend to be a large subsystem with respect to the rest of the subsystems. Therefore the communications compartment is assumed to be entire section of the fuselage. This is a cylindrical compartment where the mechanics can reach the subsystem and swap systems. The communications, like the GPS and

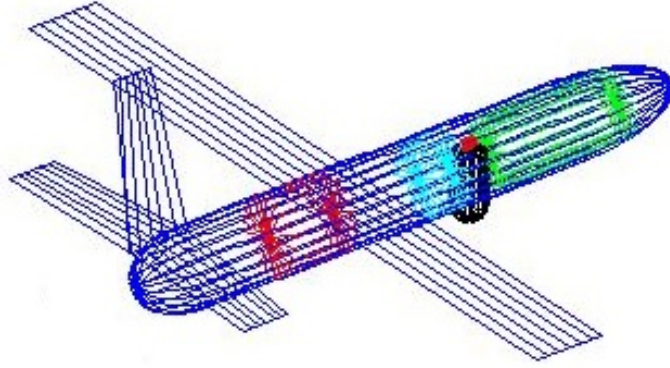


Figure 69: Final UAV Layout (m)

INS, must be far from the engine in order to reduce thermal and vibration noise.

The fuel of an aircraft tends to be stored in the fuselage and the wing; however to reduce complexity, the fuel is assumed to be at the center of gravity of the aircraft ignoring the fact that different missions require different amounts of fuel. This will eliminate center of gravity movement throughout the flight of the aircraft.

Finally, engine is cylindrical in shape. The location of the engine is assumed to be at the rear of the aircraft. This is due to electronic subsystems constraining factors. This assumption can be verified since most UAVs have electronic sensors at the front and engine at the rear.

With these constraints and assumptions the layout of internal components of the UAV have been created. The fuselage can then be fit around the subsystems, engine, and fuel. An example layout of the UAV is seen in Figure 69. The blue is the aircraft. The green cylinder is the radar, the black is the EO, the cyan cylinder is the communications, the red cylinder at the front is the navigation, and the red cylinder at the back is the engine. Though it is not iterative it is a systematic approach that provides a logical way to organize the subsystems.

The user of the FA²UST Module has the option to turn this analysis on. It provides the sizing and synthesis analysis a constraint determining whether or not there is enough capacity for the fuel required of all design missions assigned to a UAV

product. Equation 135 shows the formulation of the constraint itself. It is important to notice all constraints are normalized to prevent bias toward any one.

$$gVolume = \frac{Length_{Radar} + Diameter_{EO} + Length_{Comms} + Length_{Engine} + \frac{W_F}{46.5\pi Diameter_{Fuse}^2}}{Length_{Fuse}} - 1 \leq 0 \quad (135)$$

UAV Cost Modeling The FA²UST Module uses a modified Gudmundsson aircraft cost model to predict the development and production costs of a UAV [63]. In his model, he predicts the engineering cost (C_{ENG}), tooling cost (C_{TOOL}), manufacturing cost (C_{MFG}), development support cost (C_{DS}), flight test operations cost (C_{FT}), quality control cost (C_{QC}), cost of materials (C_{MAT}), power plant cost (C_{PP}), and propeller cost ($C_{CSTPROP}$) to predict the development (C_{DEV}) and production costs (C_{PROD}). In the FA²UST Module, these costs are predicted for each product in the product line and the cost of software development cost is added to Gudmundsson's representation of the development cost. Then, the costs are broken down based on components in each product, allowing for component specific learning curves to be applied and the elimination of redundant costs from the use of common components. Then, taking the maximum development and production costs associated with each component the costs are reformed to predict the total cost of development and production (C_{TOTAL}).

A couple of existing UAVs were used to calibrate the model, modifying the coefficients to meet the costs of developing and producing the UAVs. Throughout the explanation of the cost model the modified coefficients are highlighted in red.

The model starts with the number demanded of each product within the product line (\bar{N}_D) which the user of the module defines as an input vector. The FA²UST Module assumes three vehicles of each product will be produced during the research, development, test, and evaluation (RDT&E) phase of the design process, so $N_{RDTE} = 3$. Therefore the total number vehicles associated with each product is: $\bar{N}_{Prod} =$

$$N_{RDTE} + \bar{N}_D.$$

As explained in Section 4.1, each type of component has a vector associated with it determining which engine is associated with each product (\bar{Comp}). Using a logical expression to determine which products have component i , the resultant vector is dotted with \bar{N}_{Prod} to determine the number of each component to be produced.

$$N_{Comp_{m,n}} = \bar{N}_{Prod} \cdot (\bar{Comp}_m = n) \quad (136)$$

Using the number of each component is to be produced, a quantity discount factor (QDF) - otherwise known as a learning curve - can be enforced in the production of each component. Equation 137 shows Gudmundsson's formulation of a QDF . In the FA²UST Module, the experience factor or learning rate (F_{EXP}) is assumed to be 95%.

$$QDF_{Comp_{m,n}} = F_{EXP}^{1.44 \times \ln(N_{Comp_{m,n}})} \quad (137)$$

Gudmundsson uses a couple primary factors in his model. Some of the factors' value changes depending on the type of cost being calculated. However, one factor that does not change is the composite (f_{COMP}) factors. In the FA²UST Module's modified version of Gudmundsson model the assumption is made that a UAV will primarily be made out of composite materials, meaning $f_{COMP} = 1$.

Some of Gudmundsson's cost regressions require the maximum speed of the product (V_{H_j} : kts). The FA²UST Module usually uses the speed from the maximum speed capability segment. However, if there is no maximum speed capability segment in the product's design mission then the module takes the maximum speed from the design mission's simulation as V_{H_j} .

Gudmundsson's regressions were based on the value of a 2012 US dollars, so inflation must be included in the analysis given by the consumer price index relative to 2012 (CPI_{2012}).

Development Support Cost: Gudmundsson's regression for the predicted development support cost is a function of the product's empty weight, maximum speed, number of vehicles developed during RDT&E, a composite factor, and a complex flap factor. Equation 138 shows the formulation of the composite factor specific to the development support cost.

$$F_{COMP_{DS}} = 1 + 0.5f_{comp} \quad (138)$$

Equation 139 shows the formulation of the complex flap factor specific to the development support cost. In the FA²UST Module a one percent increase is added to the factor when the UAV has flaps, variable sweep, variable pitch, or thrust vectoring.

$$F_{CF,j_{DS}} = 1 + 0.01 (Flaps_j + VarSweep_j + VarPitch_j + VecThrust_j) \quad (139)$$

Finally, Equation 140 shows the formulation of the development support cost for a single product in the UAV product line.

$$C_{DS_j} = 0.0824 (0.65W_{Empty_j})^{0.873} V_{H_j}^{1.89} N_{RDTE}^{0.346} F_{CF,j_{DS}} F_{COMP_{DS}} CPI_{2012} \quad (140)$$

Flight Test Operations Cost: Gudmundsson's regression for the predicted flight test operations cost is a function of the product's empty weight, maximum speed, and the number of vehicles developed during RDT&E. Equation 141 shows the formulation of the flight test operations cost for a single product in the UAV product line.

$$C_{FT_j} = 0.0144 (0.65W_{Empty_j})^{1.16} V_{H_j}^{1.3718} N_{RDTE}^{1.281} CPI_{2012} \quad (141)$$

Software Development Cost: The calculation of the software development cost uses Equation 20 in Section 2.3.4. The coefficient A is user defined, and values

for it can be found in Table 25. The scaling exponents (SF_j) and cost drivers (EM_i) are user defined, and values for them can be found in Table 26. If the user does not define either A , any of the scaling exponents, or any of the cost drivers the module sets them to nominal values. The calculation of lines-of-code sums up all the unadjusted function points associated with the control of each component and multiplies it by 128 - representing the conversion of UFPs to lines of code (Table 34). Finally, the result of Equation 20 is the total effort required by the development (people-months). Therefore, the effort is multiplies by \$6,667 (2017-US)/month [7] to get the total software development cost.

Development Cost: The total development cost is the sum of the development support, flight test operations, and software development costs. Equation 142 shows the formulation of the development cost for a single product in a UAV product line.

$$C_{DEV_j} = C_{DS_j} + C_{FT_j} + C_{SOFT_j} \quad (142)$$

Manufacturing Cost: Gudmundsson's regression for the predicted manufacturing cost is a function of the product's empty weight, maximum speed, total number of vehicles produced, a composite factor, and a complex flap factor. Equation 143 shows the formulation of the composite factor specific to the manufacturing cost.

$$F_{COMP_{MFG}} = 1 + 0.25f_{comp} \quad (143)$$

Equation 144 shows the formulation of the complex flap factor specific to the manufacturing cost. In the FA²UST Module a one percent increase is added to the factor when the UAV has flaps, variable sweep, variable pitch, or thrust vectoring.

$$F_{CF,j_{MFG}} = 1 + 0.01 (Flaps_j + VarSweep_j + VarPitch_j + VecThrust_j) \quad (144)$$

Gudmundsson's regression for the predicted manufacturing cost predicts the number of man hours associated with manufacturing. Equation 145 shows the formulation of the manufacturing hours for a single product in the UAV product line.

$$H_{MFG_j} = 11.2 (0.65W_{Empty_j})^{0.74} V_{H_j}^{0.543} N_{Prod_j}^{0.524} F_{CF,j_{MFG}} F_{COMP_{MFG}} \quad (145)$$

Finally, Equation 146 shows the formulation of the manufacturing cost for a single product in the UAV product line.

$$C_{MFG_j} = 2.21 H_{MFG_j} \times 53CPI_{2012} \quad (146)$$

Tooling Cost: Gudmundsson's regression for the predicted tooling cost is a function of the product's empty weight, maximum speed, total number of vehicles produced, the production rate (Q_m : N/month), and a complex flap factor. Equation 147 shows the formulation of the production rate. The FA²UST Module assumes the total vehicles to be produced will occur over a five year span.

$$Q_m = N_{Prod}/60; \quad (147)$$

Equation 148 shows the formulation of the complex flap factor specific to the tooling cost. In the FA²UST Module a two percent increase is added to the factor when the UAV has flaps, variable sweep, variable pitch, or thrust vectoring.

$$F_{CF,j_{TOOL}} = 1 + 0.02 (Flaps_j + VarSweep_j + VarPitch_j + VecThrust_j) \quad (148)$$

Gudmundsson's regression for the predicted tooling cost predicts the number of man hours associated with tooling. Equation 149 shows the formulation of the tooling hours for a single product in the UAV product line.

$$H_{TOOL_j} = 0.975 (0.65W_{Empty_j})^{0.764} V_{H_j}^{0.899} N_{Prod_j}^{0.178} Q_m^{0.066} F_{CF,jTOOL} F_{COMP} \quad (149)$$

Finally, Equation 150 shows the formulation of the tooling cost for a single product in the UAV product line.

$$C_{TOOL_j} = 2.21 H_{TOOL_j} \times 65 CPI_{2012} \quad (150)$$

Quality Control Cost: Gudmundsson's regression for the predicted quality control cost is a function of the manufacturing cost and a complex flap factor. Equation 151 shows the formulation of the composite factor specific to the quality control cost.

$$F_{COMP_{QC}} = 1 + 0.5 f_{COMP} \quad (151)$$

Finally, Equation 152 shows the formulation of the quality control cost for a single product in the UAV product line.

$$C_{QC_j} = 0.191 C_{MFG_j} F_{COMP_{QC}} \quad (152)$$

Engineering Cost: Gudmundsson's regression for the predicted engineering cost is a function of the product's empty weight, maximum speed, total number of vehicles produced, a composite factor, and a complex flap factor. Equation 153 shows the formulation of the composite factor specific to the engineering cost.

$$F_{COMP_{ENG}} = 1 + f_{COMP} \quad (153)$$

Equation 154 shows the formulation of the complex flap factor specific to the manufacturing cost. In the FA²UST Module a three percent increase is added to the factor when the UAV has flaps, variable sweep, variable pitch, or thrust vectoring.

$$F_{CF,j_{ENG}} = 1 + 0.03 (Flaps_j + VarSweep_j + VarPitch_j + VecThrust_j) \quad (154)$$

Gudmundsson's regression for the predicted engineering cost predicts the number of man hours associated with engineering. Equation 155 shows the formulation of the engineering hours for a single product in the UAV product line.

$$H_{ENG_j} = 0.0495 (0.65W_{Empty_j})^{0.791} V_{H_j}^{1.526} N_{Prod_j}^{0.183} F_{COMP_{ENG}} F_{CF,j_{ENG}} \quad (155)$$

Finally, Equation 146 shows the formulation of the engineering cost for a single product in the UAV product line.

$$C_{ENG_j} = 2.21 H_{ENG_j} \times 92CPI_{2012} \quad (156)$$

Cost of Materials: Gudmundsson's regression for the predicted cost of materials is a function of the product's empty weight, maximum speed, total number of vehicles produced, and a complex flap factor. Equation 157 shows the formulation of the complex flap factor specific to the cost of materials. In the FA²UST Module a two percent increase is added to the factor when the UAV has flaps, variable sweep, variable pitch, or thrust vectoring.

$$F_{CF,j_{MAT}} = 1 + 0.02 (Flaps_j + VarSweep_j + VarPitch_j + VecThrust_j) \quad (157)$$

Finally, Equation 158 shows the formulation of the cost of materials for a single product in the UAV product line.

$$C_{MAT_j} = 28.2 (0.65W_{Empty_j})^{0.689} V_{H_j}^{0.624} N_{Prod_j}^{0.792} F_{CF_{MAT}} \quad (158)$$

Power Plant Cost: Gudmundsson's regression for the predicted power plant cost is a function of the engine's horsepower or thrust (lbs). It also depends on the type of engine installed on the aircraft. Equation 158 shows the formulation of the power plant cost for a single product in the UAV product line.

$$C_P P = \begin{cases} 174hp \times CPI_{2012} & Piston \\ 377.4hp \times CPI_{2012} & Turboprop \\ 1035.9T^{0.8356}CPI_{2012} & Turbofan \\ 868.1T^{0.8356}CPI_{2012} & Turbojet \end{cases} \quad (159)$$

Propeller Cost: Gudmundsson's regression for the predicted propeller cost is a function of the engine's horsepower. It also depends on the type of engine installed on the aircraft. Equation 160 shows the formulation of the propeller cost for a single product in the UAV product line.

$$C_{CSTPROP_j} = \begin{cases} 10^{[0.7746+1.1432 \log_{10}(hp)]} CPI_{1989} & Piston || Turboprop \\ 0 & otherwise \end{cases} \quad (160)$$

Production Cost: The total production cost of a UAV is the sum of the manufacturing, engineering, tooling, quality control, materials, power plant, and propeller costs. The FA²UST Module increases the cost by 25% to account for liability insurance and integration complexity. Therefore, Equation 161 shows the formulation of the production cost for a single product in the UAV product line.

$$C_{PROD_j} = 1.25 (C_{MFG_j} + C_{ENG_j} + C_{TOOL_j} + C_{QC_j} + C_{MAT_j} + C_{PP_j} + C_{CSTPROP_j}) \quad (161)$$

Component Breakdown of Costs: The component breakdown of costs requires percentages of the development and production costs associated with each component. Using various sources of information from general aviation cost breakdowns,

time spent on the development and production of each UAV component, and overall UAV cost studies the percentages for each were created [104, 87, 149]. Equations 162 and 163 show the component contribution to the development and production costs respectively.

$$\bar{K}_{CompDEV} = \left\{ \begin{array}{ll} 15\% & MainWing \\ 24\% & Engine \\ 12\% & Fuselage \\ 10\% & HorizontalTail \\ 8\% & VerticalTail \\ 12\% & EO - IR \\ 12\% & Radar \\ 7\% & other \end{array} \right\} \quad (162)$$

$$\bar{K}_{CompPROD} = \left\{ \begin{array}{ll} 18\% & MainWing \\ 19\% & Engine \\ 17\% & Fuselage \\ 12\% & HorizontalTail \\ 10\% & VerticalTail \\ 7\% & EO - IR \\ 7\% & Radar \\ 10\% & other \end{array} \right\} \quad (163)$$

The two vectors can be multiplied by the transpose of their respective development or production cost vector to create a matrix of components and their associated costs for each product across the product line. For each component (m) the maximum cost of each type of that component (n) represents the respective development or production cost associated with that specific component. Equations 164 and 165 show this process.

$$C_{Comp_{m,n-DEV}} = \max [K_{Comp_{m,PROD}} \times \bar{C}_{DEV}] \rightarrow Comp_m = n \quad (164)$$

$$C_{Comp_{m,n-PROD}} = \max [K_{Comp_{m,PROD}} \times \bar{C}_{PROD}] \rightarrow Comp_m = n \quad (165)$$

Creating the Final Development and Production Costs of the UAV Product Line: The individual component's development costs are summed together to create the total development cost of the product line. If there are common components their development cost should not be counted twice, as to avoid redundancy since it will be assumed each product is only developed once. Equation 166 shows the formulation of the overall development cost for the product line.

$$C_{PLDEV} = \sum_{m=1}^M \sum_{n=1}^N C_{Comp_{m,n-DEV}} \quad (166)$$

The individual component's production costs are first multiplied by their respective QDFs. Then, the production cost of each individual component with a product is summed to create the production cost of the product. Finally, all of the product's production costs are summed to create the overall production cost of the product line. Equation 167 shows this formulation.

$$C_{PLPROD} = \sum_{j=1}^J \sum_{m=1}^M [C_{Comp_{m,n-PROD}} \times QDF_{Comp_{m,n}} \rightarrow n \in Product_j] \quad (167)$$

4.2.2 Automobile Sizing and Synthesis Models

The FA²UST Module uses the Future Automotive Systems Technology Simulator (FASTSim) module for its automobile sizing and synthesis analysis. It was developed by the US Department of Energy's (DOE) National Renewable Energy Laboratory (NREL) to "evaluate the impact of technology improvements on efficiency,

performance, cost, and battery life in conventional vehicles, hybrid electric vehicles (HEVs), plug-in hybrid electric vehicles (PHEVs), and all-electric vehicles (EVs) [26].” Though, its purpose is to evaluate hybrid and electrical vehicles it still can determine the performance of traditional automobiles.

With its extensive database of historical vehicles, baselines can provide some of the more lesser design variables. It has sizing and performance models built in it as well. The capabilities provide the platform necessary to conduct all the analysis required.

Since FASTSim is an internalized tool for automobile sizing and performance analysis, FASTSim integrates into the FA²UST Module by setting the inputs and analyzing the outputs.

4.2.2.1 Automobile Components Considered in FASTSim

All civilian cars tend to have the same configuration, which includes a frame, engine, fuel tank, four wheels, steering system, transmission, and cargo. FASTSim’s capabilities allow the user to analyze the frame, engine, fuel tank, wheels, and cargo. It does not provide the ability to analyze the steering apparatus, and has limited ability to analyze the transmission of the vehicle.

Automobile Frame: The FA²UST Module represents the automobile frame through four variables: its frontal area (S : ft²), weight (W_{Frame} : lbs), drag coefficient (C_D), and the wheel base (b_{wheel} : ft). The module represents the frame this way because all of the variables represent the dimensions of the frame (the height can be approximated to be the frontal area divided by the wheel base - $\frac{S}{b_{wheel}}$), the weight of the frame, and the aerodynamic technologies incorporated into the design.

Automobile Engine: The FA²UST Module includes analysis of the automobile engine through the two variables: engine power (hp) and specific power (c_P : hp/lb).

Specific power is the inverse of specific fuel consumption. These two variables outline the strength and efficiency of the engine, which are key aspects of automobile design.

Automobile Fuel Tank: The FA²UST Module includes analysis of the automobile fuel tank through the two variables: fuel storage energy (FS_E : lb-ft) and fuel storage energy per weight of tank (FS_{c_E} : lb-ft/lb). These two variables explain the energy storage capacity of the fuel tank and its weight efficiency to hold that amount of fuel.

Automobile Wheels: The FA²UST Module includes analysis of the automobile wheels through the two variables: radius (R : ft) and rolling friction coefficient (μ_{roll}). These two variables explain the wheel's size and efficiency.

Automobile Cargo: The FA²UST Module includes analysis of the automobile cargo through the one variable: additional cargo weight (W_C : lbs). The cargo is just the weight the vehicle must be able to carry during all of the drive cycles. It will impact the efficiency, weight, and overall cost of the vehicle.

4.2.2.2 The Drive Cycle Requirements used by FASTSim to Size an Automobile Product

The main requirements that drive the design of an automobile are drive cycles. They outline a profile of speeds and grades of a vehicle over some timespan. FASTSim uses drive cycles to determine the overall efficiency, performance of the vehicle, size, and costs of the vehicle.

Figure 70 shows four traditional drive cycles used to size and analyze an automobile design. The first test looks at varying operating speeds at no grade. The speeds incrementally increase from 45, 55, 60, and 65 mph. These are standard civilian operating speeds in the US and close to those in Europe. The second test accelerates the vehicle as quickly as possible and holds a speed of 90 mph. This test pushes the engine and drive-train to its limits. The last two tests vary grades and speeds to

represent driving on a highway and in an urban center. Developed by the EPA, they determine the fuel efficiency and whether the vehicle can handle any extreme loads put on it by the user.

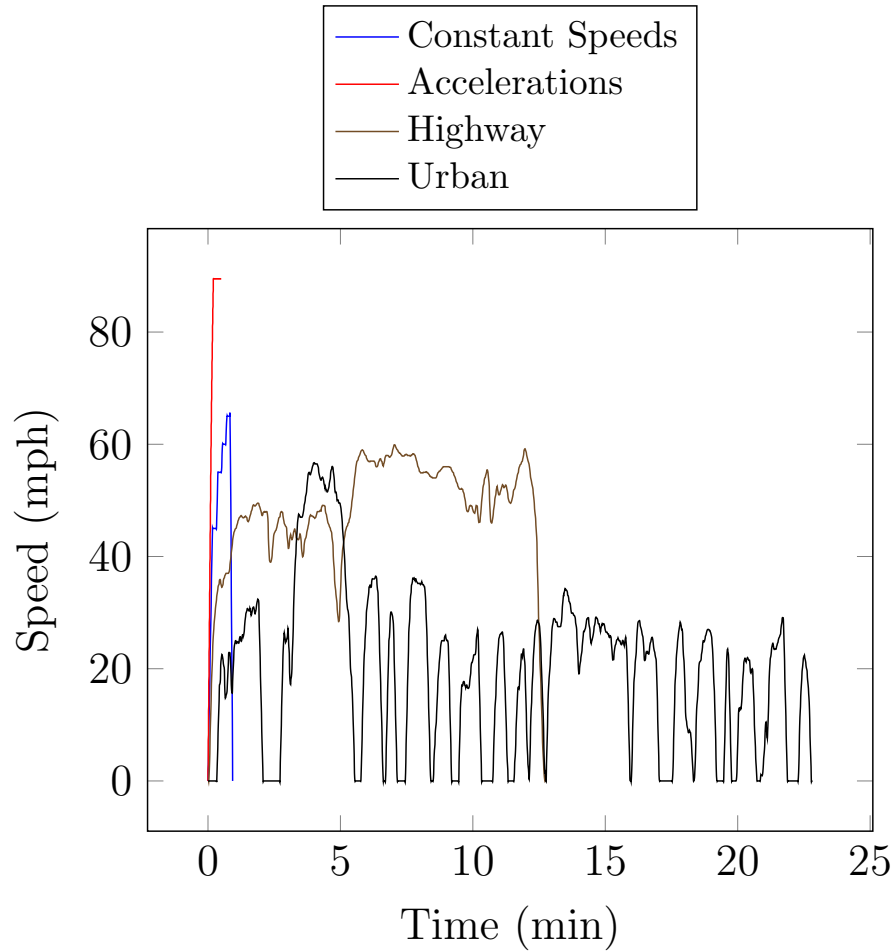


Figure 70: Drive Cycles Used to Size and Test the Automobiles

4.2.2.3 FASTSim Outputs Analyzed by the FA²UST Module

At the conclusion of automobile sizing, FASTSim outputs a wealth of information about the vehicle. The outputs can be grouped into the vehicle's fuel economy, performance, cost, and size. Specific terms were integrated into the FA²UST Module that pertains to the analysis of product architectures.

Fuel Economy Outputs: The fuel economy outputs include the laboratory and adjusted fuel economy for city (MPG_{City_L} or MPG_{City_A}), highway (MPG_{HW_L} or MPG_{HW_A}), and combined driving conditions (MPG_{Comb_L} or MPG_{Comb_A}) (mpg). These metrics give an overall picture of the cars efficiency in a laboratory setting or in real life conditions.

Performance Outputs: The two performance outputs considered are the vehicle's time to accelerate from zero to sixty miles per hour (t_{0-60} : sec) and range (R : miles). These two metrics provide insights on the power and endurance of the vehicle.

Cost Outputs: The cost output considered is the manufacturer's suggested retail price ($MSRP$: \$). This price is the acquisition cost for the consumer and has a considerable impact on the vehicle's sales.

Size Outputs: The primary size output is the simulated total weight of the vehicle (W_{Total} : lbs). It is the sum all of the components and fuel weights, and is a good metric to summarize the overall vehicle.

4.3 *FA²UST Module Outer Layer Outputs*

The second part of the FA²UST Module's outer layer calculates the evaluation metrics of the product architecture which allow the user to determine the favorability of the product architecture. Specifically, the FA²UST Module looks at desirability, requirement flexibility (the product architecture's exposure to the risk of changing requirements), and design complexity (the product architecture's internal coupling and interdependence among components or subsystems).

4.3.1 *Calculating a Product Architecture's Desirability*

The design problem for a system is brought up in Section 3.5.1 where it is presented in Equation 32, where \bar{x} is the vector of n design variables and \bar{R} is a vector of

m requirements. Equation 32 is then converted to a function in Equation 33. The components of Equation 33 include $f(\bar{x}, \bar{R})$ which is the often the inverse or negative overall evaluation criteria (OEC). The OEC is a combination output metrics from the FA²UST Module's internal analysis. It is defined in the third stage of the FA²UST framework (Section 3.3).

Another component of Equation 33 is $\phi_{g_i}(\bar{x}, \bar{R})$ which are penalty functions of inequality constraints from the FA²UST Module's internal analysis. Though there are many ways to create penalty functions from inequality constraints [60], the FA²UST Module uses a linear extended interior penalty function. Equation 168 shows how the FA²UST Module transforms the inequality constraints into penalty functions.

$$\phi_{g_i}(\bar{x}, \bar{R}) = \begin{cases} \frac{-1}{g_i(\bar{x}, \bar{R})} & g_i(\bar{x}, \bar{R}) \leq \epsilon \\ -\frac{2\epsilon - g_i(\bar{x}, \bar{R})}{\epsilon^2} & g_i(\bar{x}, \bar{R}) > \epsilon \end{cases} \quad (168)$$

Another component of Equation 33 is $\phi_{h_j}(\bar{x}, \bar{R})$ which are penalty functions of equality constraints from the FA²UST Module's internal analysis. The FA²UST Module uses a quadratic penalty function. Equation 169 shows how the FA²UST Module transforms the equality constraints into penalty functions.

$$\phi_{h_j}(\bar{x}, \bar{R}) = h_j(\bar{x}, \bar{R})^2 \quad (169)$$

Finally, the desirability (D) of a product can be expressed as the combination of these components in Equation 170. The sum of the product's desirability provide the product line's desirability.

$$D(\bar{x}, \bar{R}) = OEC(\bar{x}, \bar{R}) - \left[\sum_{i=1}^N \phi_{g_i}(\bar{x}, \bar{R}) + \sum_{j=1}^M h_j(\bar{x}, \bar{R})^2 \right] \quad (170)$$

4.3.2 Calculating a Product Architecture's Requirement Flexibility

Section 3.5.3 shows a high level or abstract formulation of the requirement flexibility. The FA²UST Module calculates the gradient and Hessian matrix of the desirability function using a finite difference method. Equation 171 shows how each element of the gradient is calculated, where e is the finite difference and given as one percent of either x_i or R_j . However, if $\|e\|$ is less than 0.001 then it is set to 0.001. To simplify the equation the design variable and requirement vectors are combined into one variable vector \bar{v} where v_k is one of the elements in the vector. The elements not defined are kept at their original values.

$$\frac{\delta D}{\delta v_k} = \frac{D(v_k = v_k + e) - D(v_k = v_k - e)}{e} \quad (171)$$

Equation 172 shows how each element of the Hessian matrix is calculated, where e is the finite difference and given as one percent of either x_i or R_j . However, if $\|e\|$ is less than 0.001 then it is set to 0.001. To simplify the equation the design variable and requirement vectors are combined into one variable vector \bar{v} where v_k is one of the elements in the vector and v_l is another. The elements not defined are kept at their original values.

$$\begin{aligned} \frac{\delta^2 D}{\delta v_k^2} &= \frac{D(v_k = v_k + e) - 2D(v_k = v_k) + D(v_k = v_k - e)}{e^2} \\ \frac{\delta^2 D}{\delta v_k \delta v_l} &= \frac{\frac{D(v_k = v_k + e, v_l = v_l + e) - D(v_k = v_k - e, v_l = v_l + e)}{e} - \frac{D(v_k = v_k + e, v_l = v_l - e) - D(v_k = v_k - e, v_l = v_l - e)}{e}}{e} \\ &= \frac{D(v_k = v_k + e, v_l = v_l + e) - D(v_k = v_k - e, v_l = v_l + e) - D(v_k = v_k + e, v_l = v_l - e) + D(v_k = v_k - e, v_l = v_l - e)}{e^2} \end{aligned} \quad (172)$$

After the gradient and Hessian calculations are complete, the requirement flexibility of the product line is calculated using Equation 38.

4.3.3 Calculating a Product Architecture's Design Complexity

The calculation of the product line's design complexity uses the information gathered from the calculation of the Hessian and gradient of the desirability of the product line (shown through Equations 171 and 172). After the Hessian and gradient calculations are complete, the design complexity of the product line is calculated using Equation 40.

4.4 *Development of FA²UST Module Summary*

All of the models explained in this chapter create the FA²UST Module. To show how they fit together, Figure 71 and Figure 72 shows the integration for the UAV and automobile analysis respectively. In the UAV analysis, the input section of the FA²UST Module outer layer reads the input file creating the list of requirements, components and their design variables, generates the product architecture indices, and provides physical, system, product architecture rules for the internal analysis.

This information is sent to the inner layer where the products within the product line are initialized. The components characteristics are defined. Then, the module delegates the physical and system architecture rules to each product. From this information the empty and payload weight of each product can be calculated.

After initializing the products, the UAV's are sent to the sizing and synthesis analysis where their weight is determined and cooling constraint can be calculated. During the sizing and synthesis, the module simulates a mission stepping through each mission segment. For each segment, the trim of the aircraft is optimized to maximize the objective function specific to each segment. Once the takeoff gross weight converges, the performance and size metrics of the UAV are sent to any additional constraints that need to be added to each product. Finally, the cost analysis of the product line is conducted.

The module then extracts specific output metrics and constraints that are required

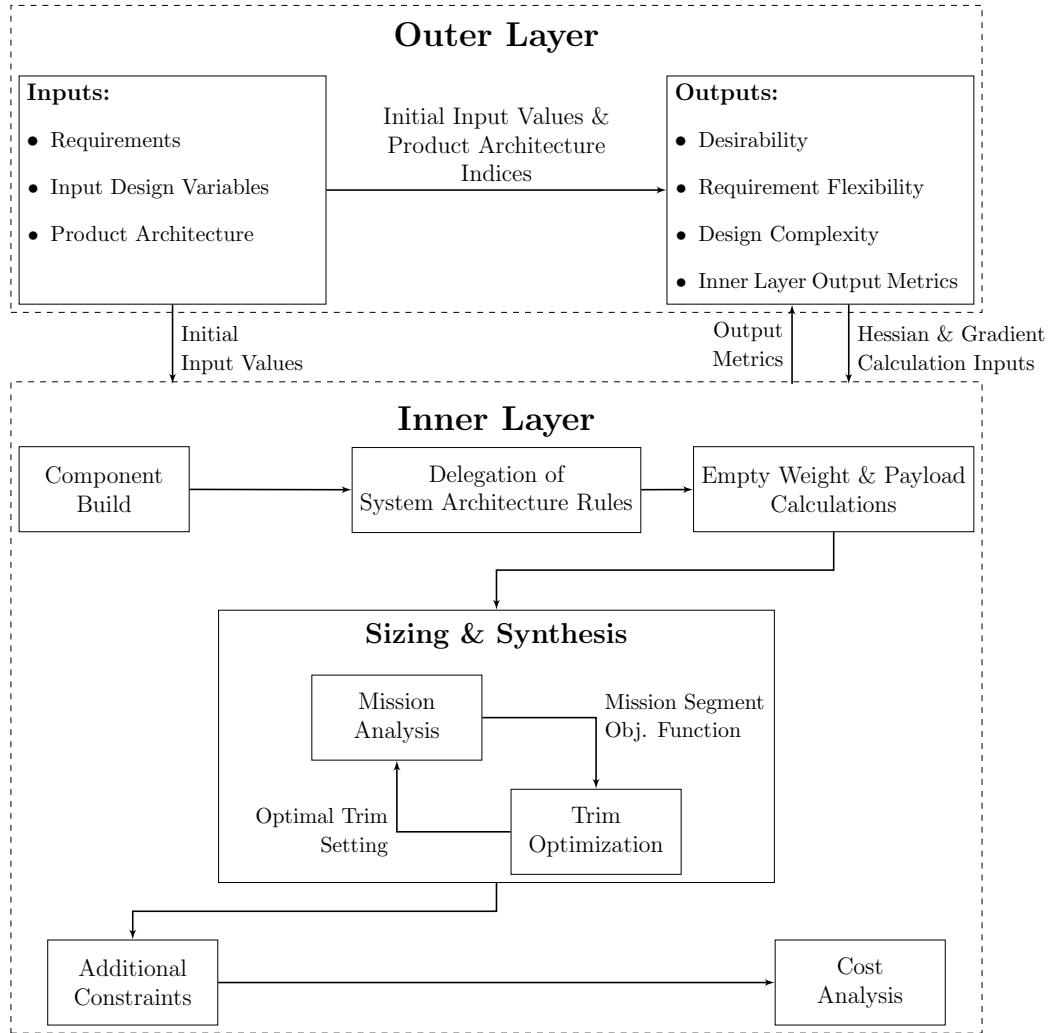


Figure 71: Integration of UAV Inner Layer Models with the Outer Layer of the FA²UST Module

to calculate the desirability of the product line. Then, the outer layer starts feeding inputs back to the inner layer so the Hessian and gradient of the product line's desirability can be calculated. Finally, the Hessian and gradient are used to calculate the requirement flexibility and design complexity of the product line.

In the UAV analysis, the input section of the FA²UST Module outer layer reads the input file creating the list of requirements, components and their design variables, generates the product architecture indices, and provides physical, system, product architecture rules for the internal analysis.

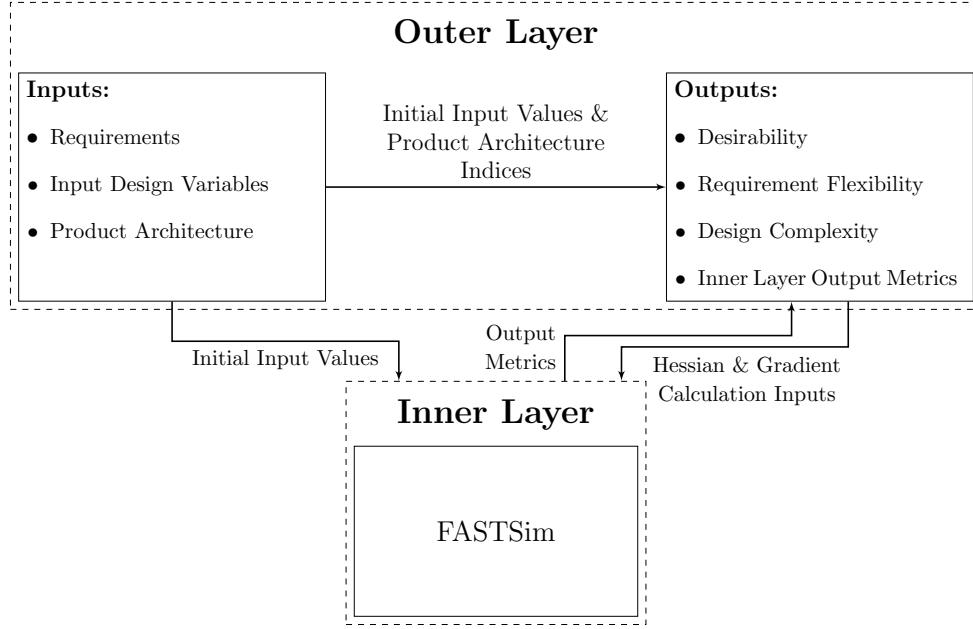


Figure 72: Integration of FASTSim Inner Layer with the Outer Layer of the FA²UST Module

This information is sent to the inner layer where FASTSim takes the components variables for each product. Then, FASTSim sizes, determines the performance, and determines the costs for each product individually.

The module then extracts specific output metrics and constraints that are required to calculate the desirability of the product line. Then, the outer layer starts feeding inputs back to the inner layer so the Hessian and gradient of the product line's desirability can be calculated. Finally, the Hessian and gradient are used to calculate the requirement flexibility and design complexity of the product line.

The integration of the FA²UST Module creates the desired capabilities to test the hypothesis formulated in Chapter 3. The next chapter tests these hypothesis using a case study of the development of a new UAV product line.

CHAPTER V

UNMANNED AERIAL VEHICLE CASE STUDY

The first case study analyzes the unmanned aerial vehicle (UAV) industry because it is developing and manufacturers within the industry implement numerous product architectures to satisfy diverse requirements and customer needs. However, most industries choice in product architecture converges over time. The materializing product architecture ends up dominating the industry and becomes the standard until a paradigm shift occurs from new technologies or market behavior. Since this industry is in its infancy, a dominant product architecture will emerge in the next few years. Therefore, this case study looks at a medium-sized, conventional UAV manufacturer.

The manufacturer is to resemble many of the players in the UAV industry whose goal is to capture as much of the potential market available when the FAA relaxes regulations, as is expected, as well as win some military contracts. As a result, their choice in product architecture will determine the success of this venture.

5.1 Establishing the Need for a New UAV Product

The first step in developing a new product is determining the needs of the new product. Following the process detailed in Figure 34, the manufacturer can determine the customer needs, the resources or capabilities required to meet them, and product specific needs. The first step in the process is to analyze the industry and the internal dynamics of the firm. Together, they can help determine the producers place in the industry. From this analysis, the firm can choose a business strategy can be, down-selecting needs specific to the new product, and determine the product-based, customer-oriented strategy.

5.1.1 UAV Industry External Analysis

The external analysis focuses on the dynamics that occur outside of the manufacturer in question. The two recommended frameworks to facilitate this analysis are the PESTEL and Five Forces. However, depending on the industry and global breadth of operations other frameworks can be introduced. However, for this case with a small manufacturing firm operating in the United States the PESTEL and Five Forces frameworks are sufficient.

5.1.1.1 PESTEL Analysis of UAV Industry

The PESTEL Framework allows engineers and management to break down the external factors that influence the industry. The factors specific to the unmanned aerial vehicle industry are as follows:

- *Political:* The unmanned aerial vehicle industry is highly sensitive to government policies and regulations. Though the United States and European regulators express their interest in opening up the private industry, both governments are slow to pursue deregulation [74, 154, 52, 2, 37]. The markets that are open tend to be maritime or remote area surveillance. In the future, these regulations will be relaxed allowing for more uses and profitability in the industry. Furthermore, militaries are finding UAVs especially useful for dull or dangerous missions [51].
- *Economic:* Producing UAVs require a significant amount of capital to produce the products. Risky ventures such as these require the economy to be healthy and growing. Since the world economy is considered healthy, multiple firms invested recently in the industry. Companies such as Boeing, Lockheed Martin, and General Dynamics have made considerate investments in the industry following an explosion in venture capital to numerous UAV start-ups [28]. These factors mean it is an excellent time to be in the industry.

- *Sociocultural:* The public's opinions of UAVs depend on country and region. Therefore, the public opinion is split regarding the UAV industry. While the vehicles can reduce costs of some mundane operations, they threaten privacy and cause the public to worry about government oversight [75]. However, moods towards the industry seem to be lightening as the regulators have promised to prevent violations of privacy. Implementation of these policies may help pave the way for the public's eventual acceptance of new technology.
- *Technological:* The industry emerged from the growth of digital electronics and silicon revolution. In the past, analog systems were too heavy to make unmanned aerial vehicles practical. The digitization of electronics drastically reduced the weight of the subsystems required in a UAV and expanded the capabilities of the systems. The technologies that have facilitated the industry's development are still improving. The subsystems' weight is decreasing, and computational power is increasing simultaneously, thus improving the performance and capability of UAVs.
- *Ecological:* The primary ecological factor influencing the aerospace industry as a whole is fuel consumption. However, UAVs require much less fuel to conduct the same missions than their manned counterparts. Therefore, fuel consumption is not as concerning in the UAV industry compared to the civil transport industry.
- *Legal:* UAVs have removed the human element from the vehicle reducing the liability of the vehicle. However, if the vehicle fails in operations it immediately becomes a projectile, threatening people and infrastructure on the ground or in the air. Vehicle failure and the resultant damage could cause hefty litigation. Therefore, the operating the vehicle should be conducted by licensed pilots and quality must be extremely high to offset the risk of failure.

The analysis from the PESTEL Framework describes an industry with great potential. The industry's growth is heavily dependent on political regulations relaxing, but the industry and outside investment think the trends will continue in the UAV industry's favor.

5.1.1.2 Five Forces of UAV Industry

Following the analysis of the UAV industry's external considerations, it is essential to analyze the profitability of the industry. The Five Forces model analyzes the manufacturer's relative power within the industry. The manufacturer's power relates to its ability to negotiate favorable deals and increase profit margins. Furthermore, it can help determine possible business strategies the firm can implement in the industry. The Five Forces for the UAV industry are as follows:

- *Bargaining Power of Buyers: (High)* Many of the firms that already exist in the UAV industry have not been able to differentiate themselves, and since there are so many, consumers can quickly switch amongst firms. The low switching costs allow consumers to demand more out of the products reducing margins for the producers.
- *Bargaining Power of Suppliers: (Medium)* Firms that supply the UAV industry primarily consist of raw material, electronics, or specialized subsystems including engines and sensors. The raw materials and commercially available electronics are commodities and do not have much bargaining power in general. However, the engines, military-grade electronics, and specialty sensors have much more power depending on the market segment the manufacturer is competing in.
- *The Threat of New Entrants: (Medium)* New entrants entered the industry consistently since 2004, spurred by innovative concepts and Wall Street's continued investment in technology start-ups. However, the established companies

present in the industry stemmed the flow by consolidating the highly profitable military contracts. As a result, those that still enter the industry focus on the civil market space.

- *The Threat of Substitute Products or Services: (Low)* The UAV industry is a substitute for many of the manned alternatives present in the unmanned industry. Therefore, no real substituting threat emerged to the industry.
- *Rivalry among Existing Competitors: (High)* Competition in the industry is extremely high. Multiple small firms are trying to grab their stake in the industry, and the aerospace giants have made their investments in the industry as well. Since the industry is emerging, the uncertainty in the markets is complex or chaotic. Thus, the established firms are trying to shape the industry by lobbying politicians or acquiring the competition. The smaller firms must react to the established firms positions and must fight over less profitable market segments.

The result of the analysis shows the two possible markets are hard to compete in for a smaller manufacturer. The large aerospace engineering firms dominate the higher margin military-contract market segment. The lower margin or undeveloped public market segment has a large number of competitors with relatively low power. The dynamics of the market leave the firm with two options: disrupt the military-grade market with lower-priced alternatives, or differentiate itself in the public marketplace. Both options are not extremely attractive. Disruptive strategies offer low-cost or revolutionary new products that rapidly take market share. These strategies require the volumes necessary to achieve economies of scale. So in this case military contracts do not demand the required volume to implement this strategy successfully. Furthermore, the only way to differentiate in the civil space is in quality of the product. Regulations determine the quality requirements which depend on vehicle size

and operations. Therefore, an internal analysis of the case study manufacturer must be conducted to see which option is more favorable.

5.1.2 UAV Manufacturer Internal Analysis

The internal analysis focuses on the resources, capabilities, and structure inherent of the firm in question. The VRIO framework and value chain analysis allow the firm to develop strategies around the strengths of the company. Furthermore, it identifies weaknesses that the business must bolster. There are other frameworks available, but the two methods are sufficient in determining high-level characteristics relevant to product development of the firm.

5.1.2.1 VRIO Analysis of UAV Manufacturer

This case's hypothetical manufacturer is mid-sized. Formed in the mid to late 2000s by a group of engineers who splintered off from a larger company or came together to form a new venture. This company does not have the large "bureaucratic-like" matrix structure like the larger aerospace firms, but due to its size does not have the same access to large sums of capital or developed specialized departments. The company instead has a competent group of systems, controls, and design engineers as well as competent technicians for UAV production. Its primary resources and capabilities are its systems engineering knowledge-base, UAV expertise, a tight-knit team, production knowledge-base, and access to venture capital.

The VRIO analysis looks at a company's resources and capabilities to determine which provides the firm with a distinct advantage compared to their competitors. The capabilities and resources that are valuable, rare, hard to imitate, and if the company is organized to capture their value should be leveraged in the new strategy. Table 52 displays the results from the VRIO analysis.

From the analysis, it is apparent that none of the resources and capabilities will

Table 52: UAV Industry VRIO Analysis

Resources	Valuable	Rare	Un-Imitable	Organized
Systems Knowledge-Base	✓	✓		✓
UAV Expertise	✓			✓
Tight-knit Team	✓	✓		✓
Production Knowledge-Base	✓			✓
Venture Capital	✓			✓

guarantee a long-term advantage; instead, they can create short-term success depending on the strategy the company implements. Specifically, the firm should leverage its systems engineering knowledge-base and tight-knit chemistry within the team, suggesting they could take on more complicated designs if necessary.

5.1.2.2 Value Chain Analysis of UAV Manufacturer

After, determining critical resources and capabilities available to the firm, value chain analysis looks at the internal structure of the organization. It is assumed the firm can handle the supporting activities, which include the organization's infrastructure, human resources, and resource procurement. For primary activities, the firm can manage the inbound and outbound logistics, marketing, sales, and service. Since the firm is small, there are some considerations concerning technology development and operations or production. The disciplines required for UAV production and the difficulty for the firm to develop subsystems and technologies in each domain are:

- *Aerodynamics: (Moderate-Easy)* The firm in question has expertise in UAV design. Therefore, the engineers in the company are quite competent in the field of aerodynamics and the subsystems required for flight.
- *Propulsion: (Hard)* The firm's engineers do not have expertise in the field, the field requires an enormous amount of investment to develop, and there are already some external firms that develop aerospace power plants.

- *Structures: (Moderate)* Designers can only conduct structural analysis once they choose the design's configuration. Thus, it is a part of the integration of the subsystems and components. The engineers have a history of systems and UAV design. Therefore, the firm can handle the structural analysis required.
- *Electronic Payloads: (Hard)* A UAV requires various electronic subsystems to complete the desired capabilities. The development of each demands extensive amounts of capital and expertise. Thus, there are plenty of firms that already specialization in these subsystems.
- *Controls: (Moderate)* Controls of UAVs is not a simple task, but the domain is an intricate part of systems design and integration. Control system development requires the at least the completion of a conceptual design. Therefore, control system design is a part the overall systems design.
- *Production: (Easy)* The firm's employees have expertise in this field and the company already invested in the facilities required for UAV production.

Engine and electronics development require an enormous amount of R&D costs and technical expertise. Furthermore, there are many entities in the industry that already focus on these subsystems' development and production. Thus, the firm should consider other approaches rather than vertically integrating these entities in the value chain.

The first option is to taper activities in the value chain. Tapering involves orchestrating external firms to manufacture subsystems or subcomponents. The second is to outsource activities by purchasing goods required by the product. Tapering implies cooperation between the firm and the external entities, while outsourcing primarily acquires products previously developed by the external entities.

The factor that drives this decision is suppliers power. In this case, the electronic and power plant companies have moderate bargaining power, depending on the

performance demanded of the subsystem. Therefore, it depends on what types of products the firm in question decides to produce.

The electronic subsystems and engine-design companies either produce generic subsystems that are supposed to meet specific markets requirements, or specialty, high-performing subsystems developed through cooperation between the vehicle designer and subsystem manufacturer.

If the firm decides to produce high-performance products, then taper integration is required. However, if the firm decides to produce lower performing products, then the firm should outsource. Either way, the firm must rely on external firms to produce and invest in technology specific to the subsystems required for the new product. The firm should focus on the design and integration of the systems extracting more value from the combined capabilities of the products.

5.1.3 Selecting UAV Industry Business Strategy

After analyzing the industry and the internal capabilities of the firm in question, the firm must formulate a business strategy. The firm in question is entering a highly competitive market with multiple market segments. Figure 38 shows the three primary segments: hobby, civil surveillance, and military-grade UAVs.

The hobby segment is a low margin market with many producers. The barriers to entering this segment are so low that people often build UAVs, hence “do-it-yourself (DIY).” The civilian market is slowly growing and will explode once regulations loosen. In this market, there are numerous competitors, and margins are slim. Finally, the military grade segment is the most established market. A few of the more significant UAV manufacturers dominate this market, making it hard for smaller firms to gain government contracts.

Therefore, Figure 73 shows the proposed target segments for this case study.

The firm should focus on the civilian and military-grade segments. The strategy

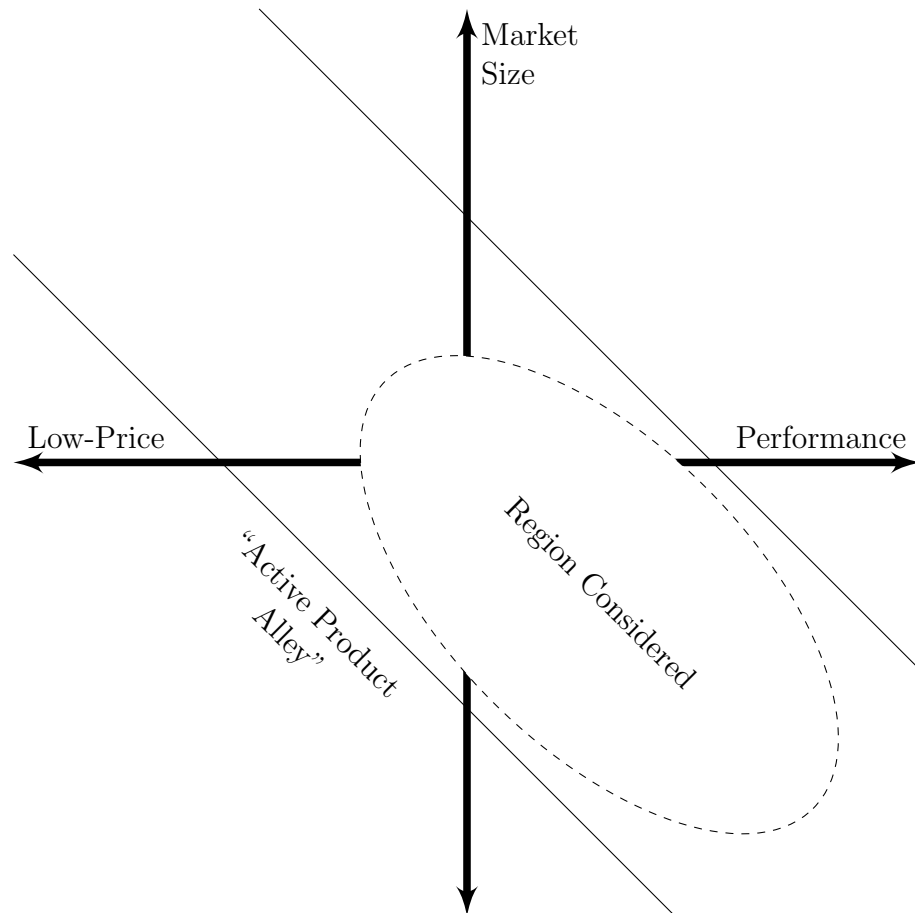


Figure 73: The Capability and Market Size Relational Space for the UAV Industry

should be a hybrid of differentiation and price leadership in the civil and military-grade market segments. The firm can produce a higher quality product in the private space, increasing margins and a low-cost alternative in the military-grade market, gaining market share. The firm can achieve this strategy by producing products for both segments on the same production line. Concurrent productions will take advantage of combining volumes from both segments and enforcing the higher quality standards to both. Furthermore, since it will be difficult to break into the military grade market, the firm can establish itself in the civilian market which will lead to government contracts and confidence. Thus, the military-grade market will open for the firm.

The hybrid strategy creates two strategic position and competitive scope pairings (Figure 37). The position and scope pairing for the civilian segment is differentiation and for the military-grade segment is focused-price leadership.

5.1.4 Extracting Customer Needs for New UAV

Following the formation of the firm's business strategy, the firm must establish the needs for the product. The external analysis already discovered three needs, based on the dynamics of the industry. The value chain analysis identified the need to incorporate modular techniques with regards to the engines and electronic subsystems since the firm should plan to outsource their production. The business strategy requires the firm to produce the products on the same production line. Concurrent production implies common components or processes which will achieve the volume required to meet the desired price points.

Now, the firm must derive the customer-specific needs. Figure 39 displays the options available for the firm. Concerning the formulated business strategy, the civilian and military-grade segments require different approaches. The customers in

the private space have moderate-to-low negotiating power. Thus, the firm can conduct customer surveys or market analysis. The customers the firm would likely be targeting are large corporations that need specific assets monitored. For example, it could be an oil company monitoring pipelines or off-shore oil rigs. Another example would be large farming corporations who need to monitor their crops or livestock over large swaths of land. The customers in the military-grade segment have a lot of power. Therefore, the company would have to meet the desired capability specified by the government organization, for example, a request for proposal often outlines the desired capability. Furthermore, the product must meet stringent military-grade standards.

The combination of concurrent production and diverse mission requirements suggests the implementation of online reconfigurable components to maximize performance concerning the concurrent production constraints. The final needs of the product line are as follows:

1. **Price:** The purposes of UAVs are to be a lower-priced alternative to manned aerial vehicles that can achieve the same or more significant capabilities. Therefore, operating and acquisition costs must be less than their manned counterparts.
2. **Sufficient Performance:** The vehicles must be as capable as their manned counterparts.
3. **Reliability:** The vehicles must be highly reliable during operations to prevent failure and loss of vehicle, primarily when operating in the private space.
4. **Modular and Outsourced Components:** The product line should incorporate modular techniques with regards to the engines and electronic subsystems since the firm should plan to outsource their production.

5. **Production Line:** The need for concurrent production implies common components or processes which will achieve the volume required for meeting the desired price points.
6. **Civilian Market Segment:**
 - (a) *Performance:* The UAV must be able to fly over moderate distances and have moderate endurance. Due to the lack of speed requirements, the vehicle must have good endurance characteristics.
 - (b) *Price:* The price point can be a higher than average in this segment since a differentiation should draw a higher price than the rest of the field.
7. **Military-Grade Market Segment:** The military-grade segment contains two types of UAV capabilities: surveillance and high-speed. Derivation of the mission profiles can originate from military standards [43].
 - (a) *Surveillance Performance:* The UAV must be able to fly over long distances, resulting in high endurance. Due to the lack of speed requirements, the vehicle must have good endurance characteristics.
 - (b) *Surveillance Price:* The strategy indicates the production of low-cost solutions. Therefore, the price needs to be less than the price offered by its competitors.
 - (c) *High-Speed Performance:* The UAV must be able to reach transonic speeds. The vehicle must also be able to fly over mid to long distances.
 - (d) *High-Speed Price:* The strategy indicates the production of low-cost solutions. Therefore, the price needs to be less than the price offered by its competitors.
8. **Online Reconfigurability:** Online reconfigurable characteristics should be

considered to offset the degradation of performance as a result of concurrent production and commonality.

The needs identified using this approach form general descriptions of the tasks, missions, and capabilities required by the vehicle.

5.1.5 Final UAV-Based, Customer-Oriented Business Strategy

The firm should focus on satisfying the requirement of delivering a payload over a given distance. The manufacturer can achieve this capability over various distances and speeds requiring a UAV product line to satisfy the desired tasks. The firm should pursue the market targeting two market segments: civilian and military-grade. The pursuit of two different market segments suggests the production of a few vehicle variants on the same production line. Concurrent production and modular standards should save on cost without diminishing performance extensively. Furthermore, the designer can offset the possible degradation of performance by implementing online reconfigurable characteristics. The uncertainty of the product architecture drives the need to explore the space and analyze the trade-offs between various vehicles. The next step for the firm would be to create concrete definitions of the tasks, missions, and capabilities required of the systems.

5.2 Defining the UAV Design Problem

After establishing the needs for the new product line and the formulation of a customer-oriented, product-based business strategy, designers must go through the process of transforming the abstract needs to detailed functional requirements. Many of the facilitators found in Section ?? can help, alongside the requirements analysis process found in Section 2.1.2 to create these requirements. For UAVs, the analysis forms design missions which outline the series of events the vehicle must be able to produce. Furthermore, their technical requirements can provide benchmarks of expected costs,

speeds, reliability, technology level, or other metrics. In this case study, the primary concerns are performance and cost.

The first step in this process is defining the technical parameters that engineers can directly trace from the product's needs. By looking at historical data benchmarks can be set for the UAVs' cost, speed, endurance, and range. Figure 74 shows the distribution of historical UAVs with missions similar to that of a civil surveillance mission. Since the product should be a little better than the market, a benchmark price of \$50K (2017-US) was selected, reflecting the 75% of the distribution. An endurance of 4 to 10 hours was selected, reflecting the 50% to 75% of the distribution. A range of 100 to 200 miles was selected reflecting the 75% of the distribution to the mean. A payload weight of 100 lbs was set as a benchmark. Finally, a fuel weight of 100 lbs was set as a benchmark representing a value slightly higher than the median of the distribution.

Figure 75 shows the distribution of historical UAVs with missions similar to that of a military-grade surveillance mission. Since the product should be a lower-cost solution when compared to the market, a benchmark price of \$5Million (2017-US) was selected, reflecting the 25% of the distribution. An endurance of 4 to 10 hours was selected, reflecting the 50% to 75% of the distribution. A range of 150 to 450 miles was selected reflecting the 75% of the distribution to the mean. A payload weight of 200 lbs was set as a benchmark representing a value slightly higher than the median of the distribution. Finally, a fuel weight of 200 lbs was set as a benchmark representing a value slightly higher than the median of the distribution.

Figure 76 shows the distribution of historical UAVs with missions similar to that of a military-grade high speed mission. Since the product should be a lower-cost solution when compared to the market and with a limited amount of publicly available data on this type of mission, a benchmark price of \$5Million (2017-US) was selected, reflecting the 50% of the distribution. A maximum Mach number of 1.2 was selected since it

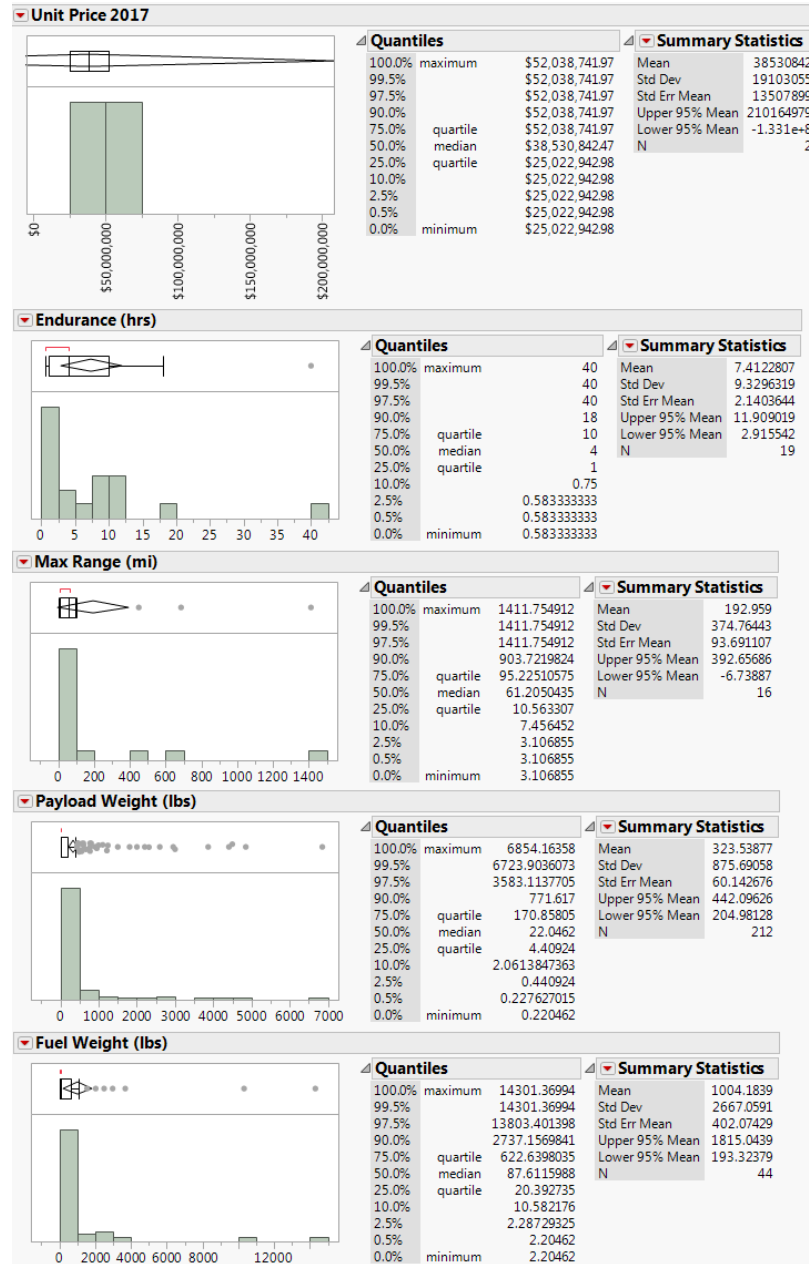


Figure 74: Civil Surveillance Mission Historical Vehicles

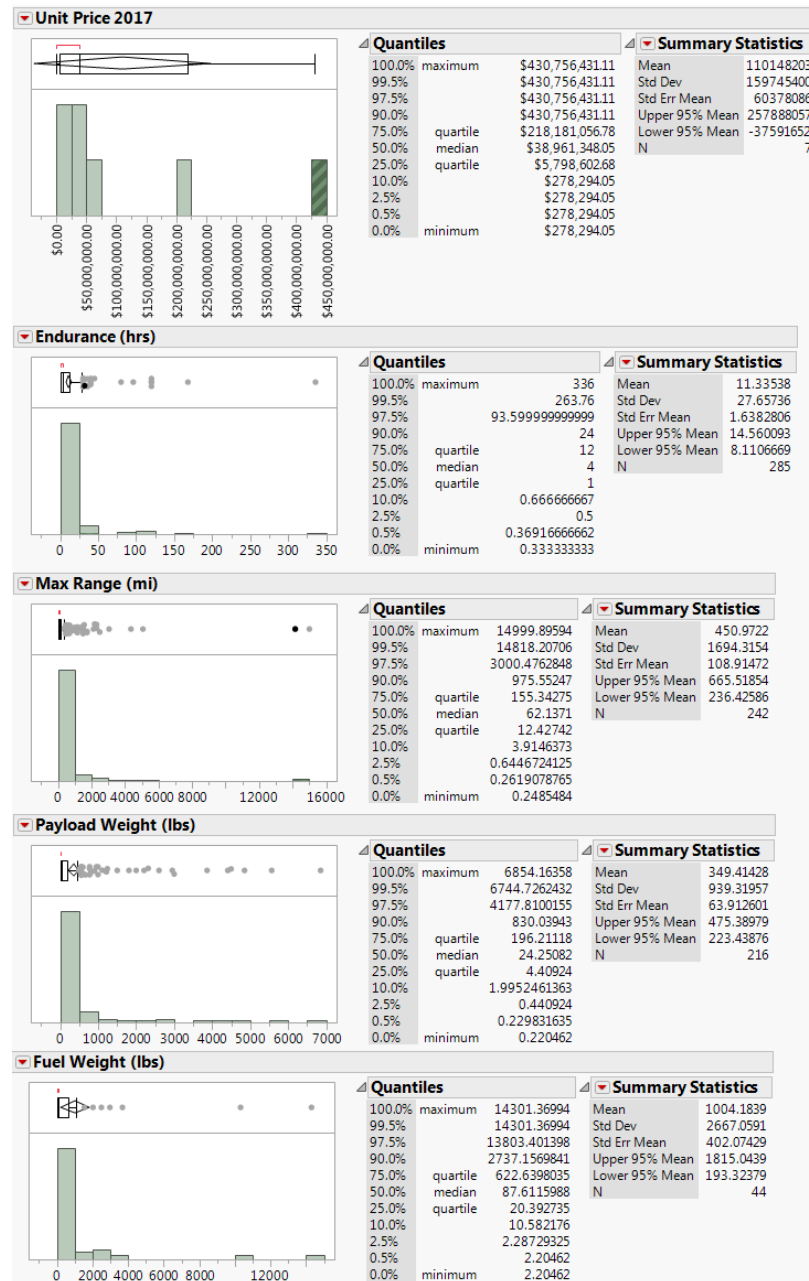


Figure 75: Military-Grade Surveillance Mission Historical Vehicles

was the highest speed of any UAV in the database. An endurance of 4 to 8 hours was selected, reflecting the 25% to 50% of the distribution. A range of 150 to 350 miles was selected reflecting the lower 25% of the distribution. A payload weight of 100 lbs was set as a benchmark representing the lower 25% of the distribution. Finally, a fuel weight of 300 lbs was set as a benchmark reflecting the lower 25% of the distribution.

In this case, the cost, speed, endurance, and range associated with each mission are displayed in Table 53.

Table 53: Technical Requirements of UAV Case Study

Mission	Acquisition Price	Max Speed (Mach)	Endurance (hrs)	Range (mi)
Civil Surveillance	\$50K	N/A	4 - 10	100 - 200
Military-Grade Surveillance	\$5-mil	N/A	4 - 10	150 - 450
Military-Grade High-Speed	\$5-mil	1.2	4 - 8	150 - 350

The next step is form design missions that define the actual serial steps a UAV must be able to complete. The design missions provide inputs to the simplified models (Section 2.1.4.3) which size the vehicle and estimate performance.

5.2.1 Decomposition of Tasks Required of UAV

Standardized mission profiles provide the benchmarks for the design missions. Since the UAV industry's origin comes from military practices, the military profiles provide the benchmarks. MIL-STD-3013 provides example missions varying from support, fighter, attack, bomber, and reconnaissance aircraft missions [43]. For this case, the strategy requires three missions with varying degrees of endurance, range, and speed requirements.

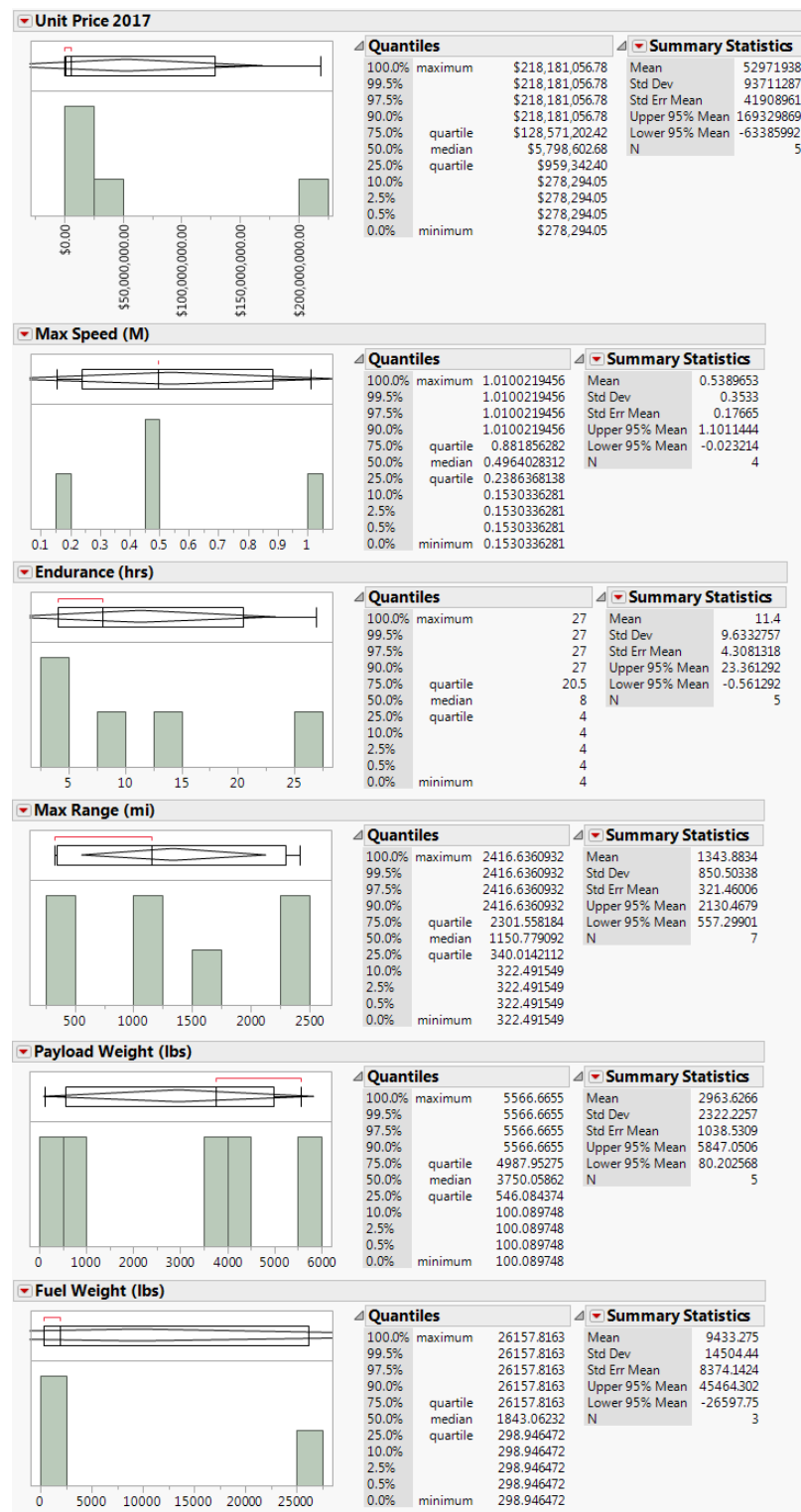


Figure 76: Military-Grade High-Speed Mission Historical Vehicles

5.2.1.1 UAV Civilian Surveillance Mission

The civilian reconnaissance mission would be most similar to a combat air patrol. The vehicle cruises at flight conditions specific to maximum range out to a location, then loiters as it surveys an area. After hitting a fuel limit or finishing the required tasks, it would cruise back to the mission's origins. Thus, Table 54 outlines the mission details.

Table 54: UAV Industry Civil Surveillance Design Mission [43]

Segment	Fuel	Time	Distance	Speed	Altitude
Warm-Up, Takeoff, Accelerate to Climb Speed OR Catapult	20min at Ground Idle + 30 SEC at Takeoff / Maximum / IRT (A/B if required) + Fuel to accelerate from obs. clearance to climb speed at IRT. No distance credit OR Catapult				
Climb				Minimum Time Climb Schedule	Takeoff to Max Range
Cruise			100 - 200 mi.	Max Range	Max Range
Descent	None	None	No Credit		End Cruise to Loiter Alt.
Loiter		4 - 10 hrs.	No Credit	Max Endurance	Max Endurance
Climb				Minimum Time Climb Schedule	Loiter Alt. to Max Range
Cruise			100 - 200 mi.	Max Range	Max Range
Descent	None	None	No Credit		End Cruise to Landing
Reserves	20min + 5% of initial fuel		No Credit	Maximum Endurance	Sea Level

The civil surveillance mission consists of nine phases. The takeoff phase can be either from a runway or use of a catapult. The other primary mission segments are a

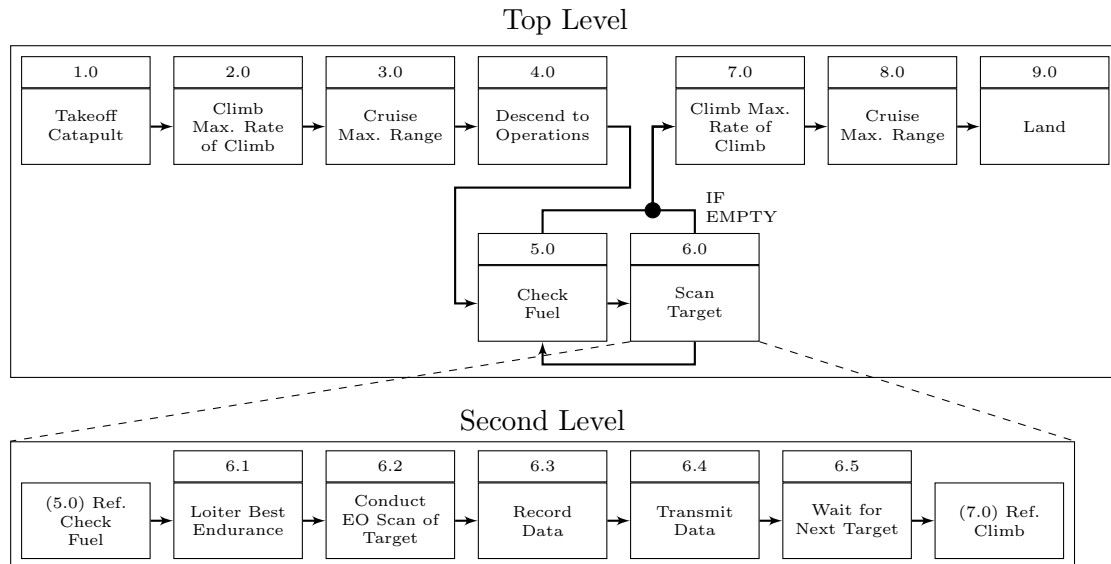


Figure 77: Function Flow Block Diagram of UAV Civil Surveillance Mission

climb, cruise, and loiter segments whose flight conditions should be set to maximize the rate of climb, range, and endurance. The mission outline then can be broken down into a functional flow block diagram. This step details the functions the system must conduct. Figure 77 displays the functional flow block diagram for the civil surveillance mission.

Figure 77 breaks down the civil surveillance mission. It focuses explicitly on the “Scan Target” phase which includes loitering and using an EO sensor to image the target. The “Scan Target” phase repeats until the fuel hits a limit or the vehicle has completed its mission.

Following the functional flow block diagram, a functional-physical matrix can assist in determining configuration options. There are certain phases in the mission that require specific components. Therefore, the design must include those subsystems. Furthermore, a phase might have multiple component combinations. Therefore, these should remain as options during the design process. Figure 78 shows the functional-physical breakdown of the civil surveillance mission.

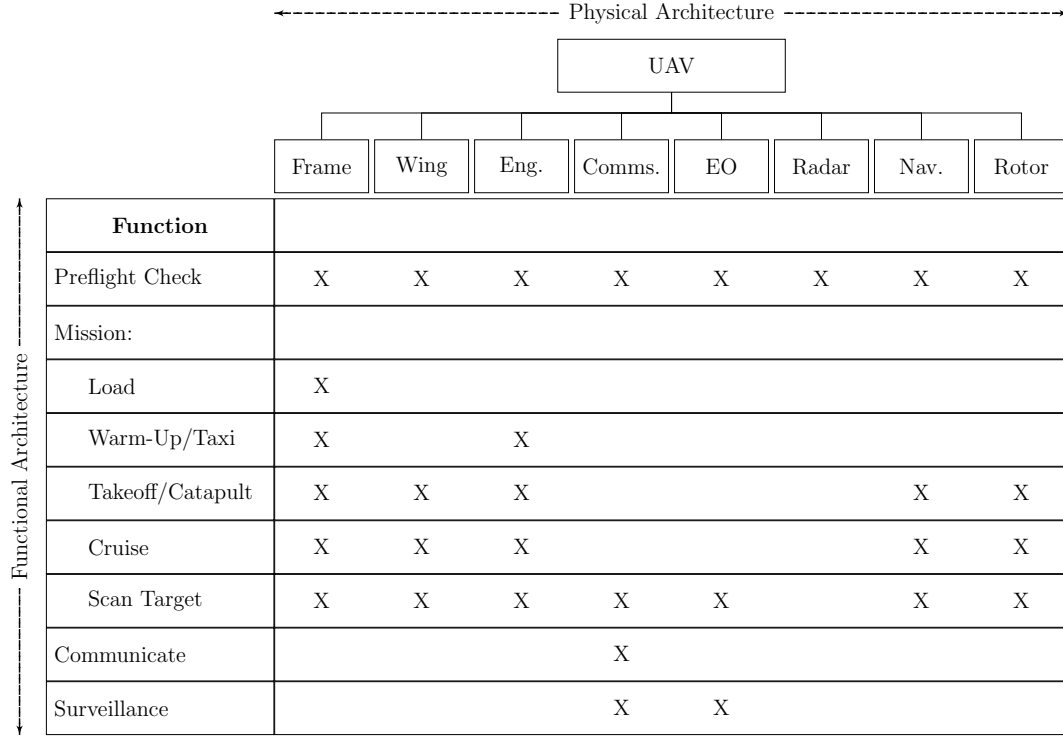


Figure 78: UAV Civil Surveillance Mission Functional/Physical Matrix

From the functional-physical matrix in Figure 78, the mission requires an EO sensor, but not necessarily radar. Furthermore, it could use a wing or a rotor to provide lift, due to its low required speeds, range, and altitude. However, there is no requirement for vertical takeoff and landing making a rotor irrelevant. Furthermore, the low speeds, range, and altitudes required of the system suggest turbofan or turboprop engines are impractical. Designers use jet engines to create the thrust characteristics to achieve high speeds or altitudes. Therefore, the system favors the use of a piston or turboprop engine.

5.2.1.2 UAV Military-Grade Surveillance Mission

The military-grade surveillance mission's structure is the same as the civil version. The vehicle is required to cruise at maximum range flight conditions to a location where it would monitor an area. After completing the assigned task or reaching a fuel limit, the aircraft would cruise back to the mission's origin. However, it requires

longer ranges and loiters, expecting higher performance from the product. Table 55 displays the outline of the military-grade surveillance mission.

Table 55: UAV Industry Military-Grade Surveillance Design Mission [43]

Segment	Fuel	Time	Distance	Speed	Altitude
Warm-Up, Takeoff, Accelerate to Climb Speed OR Catapult	20min at Ground Idle + 30 SEC at Takeoff / Maximum / IRT (A/B if required) + Fuel to accelerate from obs. clearance to climb speed at IRT. No distance credit OR Catapult				
Climb				Minimum Time Climb Schedule	Takeoff to Max Range
Cruise			150 - 450 mi.	Max Range	Max Range
Descent	None	None	No Credit		End Cruise to Loiter Alt.
Loiter		4 - 10 hrs.	No Credit	Max Endurance	Max Endurance
Climb				Minimum Time Climb Schedule	Loiter Alt. to Max Range
Cruise			150 - 450 mi.	Max Range	Max Range
Descent	None	None	No Credit		End Cruise to Landing
Reserves	20min + 5% of initial fuel		No Credit	Maximum Endurance	Sea Level

Consistent with the civil surveillance mission, the military-grade surveillance mission consists of nine phases. However, the takeoff phase uses a runway. The rest of the mission is self-explanatory. The other primary mission segments are a climb, cruise, and loiter segments whose flight conditions should be set to maximize the rate of climb, range, and endurance. The mission outline then can be broken down into a functional flow block diagram. This step details the functions the system must

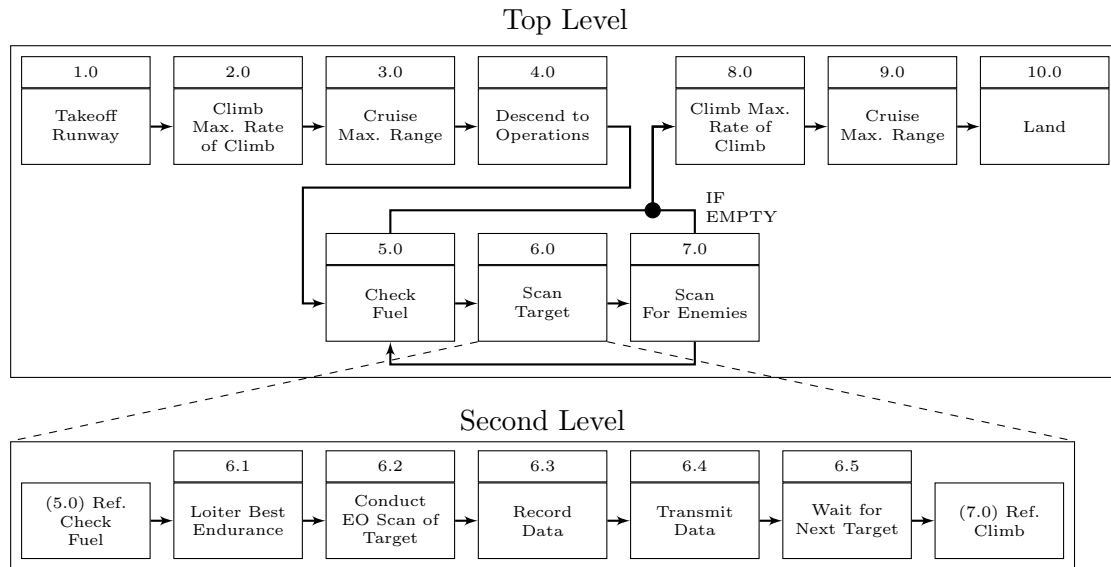


Figure 79: Function Flow Block Diagram of UAV Military-Grade Surveillance Mission

conduct. Figure 79 displays the functional flow block diagram for the military-grade surveillance mission.

The functional flow block diagram, in Figure 79, breaks down the mission segments into functions. Specifically, this FFBD focuses on the “Scan Target” phase. The vehicle must use an EO sensor to capture images of the target and radar to detect any threats in the area. Military missions require more capabilities to ensure the mission’s success.

Following the functional flow block diagram, a functional-physical matrix can assist in determining configuration options. There are certain phases in the mission that require specific components. Therefore, the design must include those subsystems. Furthermore, a phase might have multiple component combinations. These combinations remain as options during the design process. Figure 80 shows the functional-physical breakdown of the military-grade surveillance mission.

From the functional-physical matrix in Figure 80, the mission requires an EO sensor and radar. Furthermore, it could use a wing or a rotor to provide lift, due to its nominal required speeds, range, and altitude. However, there is no requirement for

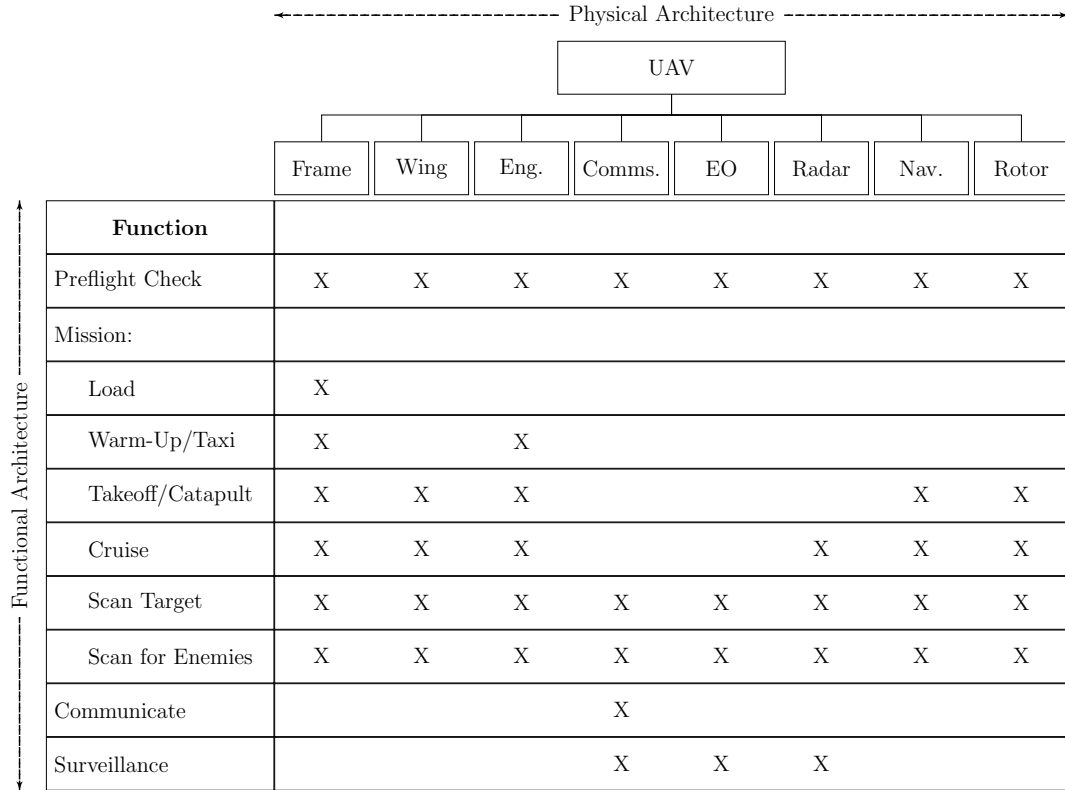


Figure 80: UAV Military-Grade Surveillance Mission Functional/Physical Matrix

vertical takeoff and landing making a rotor irrelevant. Rotors tend to be inefficient, and for a more extended range and endurance mission, configurations with rotors become impractical. Furthermore, the nominal altitudes and ranges allow designers to consider using piston, turboprop, turbofan, and turboprop engines depending on cost and performance preferences.

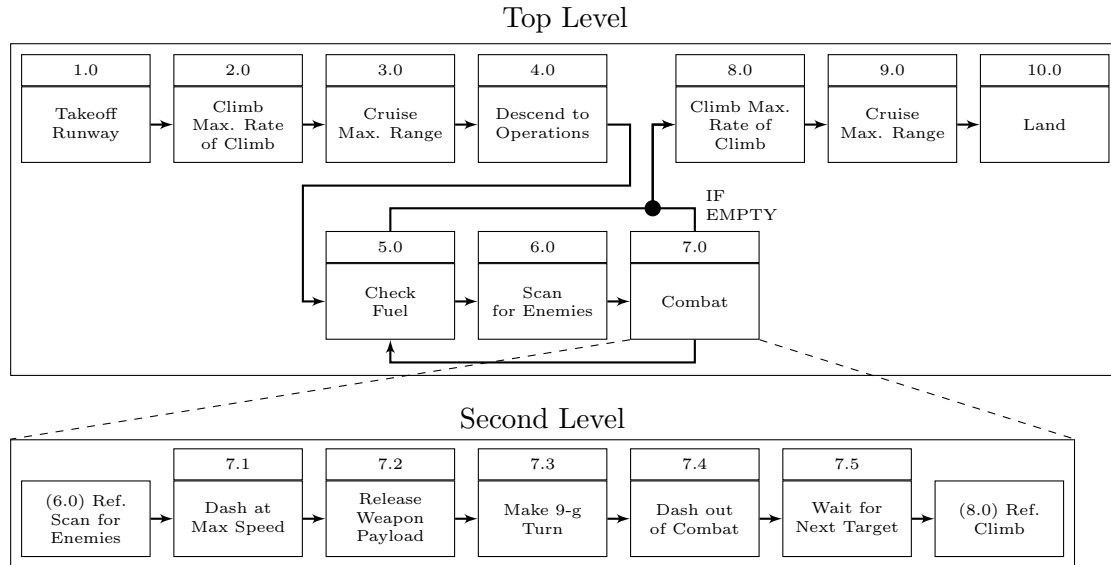


Figure 81: Function Flow Block Diagram of UAV Military-Grade High-Speed Mission

5.2.1.3 UAV Military-Grade High-Speed Mission

The military-grade, high-speed mission's structure is similar to the surveillance mission. The mission exchanges some of the loiter time for a maximum speed dash into a combat zone followed a high-g turn and a maximum speed dash out of the combat zone. Before and after the combat phase, the aircraft is expected to conduct maximum range cruises, and maximum endurance loiters. The loiter segment is there to allow the vehicle to scan for targets or receive orders from command before dashing into the combat zone. Table 56 outlines the military-grade, high-speed mission.

This mission consists of twelve segments. Similar to the surveillance missions, the structure consists of a cruise to a combat zone, and a quick loiter time. There are three segments associated with combat that push the performance of the system, demanding high speeds and loads on the vehicle. These requirements often have a considerable impact on the design of the vehicle. The mission outline then can be broken down into a functional flow block diagram. This step details the functions the system must conduct. Figure 81 displays the functional flow block diagram for the military-grade, high-speed mission.

Table 56: UAV Industry Military-Grade High-Speed Design Mission [43]

Segment	Fuel	Time	Distance	Speed	Altitude
Warm-Up, Takeoff, Accelerate to Climb Speed OR Catapult	20min at Ground Idle + 30 SEC at Takeoff / Maximum / IRT (A/B if required) + Fuel to accelerate from obs. clearance to climb speed at IRT. No distance credit OR Catapult				
Climb				Minimum Time Climb Schedule	Takeoff to Max Range
Cruise			150 - 350 mi.	Max Range	Max Range
Descent	None	None	No Credit		End Cruise to Loiter Alt.
Loiter		1 - 8 hr.	No Credit	Max Endurance	Max Endurance
Dash		No Credit	5 - 30 mi.	Max Speed	Loiter Alt.
Turn	Nine ‘g’ 180 deg. turn at Max Speed and Loiter Alt.				
Dash		No Credit	5 - 30 mi.	Max Speed	Loiter Alt.
Climb				Minimum Time Climb Schedule	Loiter Alt. to Max Range
Cruise			150 - 350 mi.	Max Range	Max Range
Descent	None	None	No Credit		End Cruise to Landing
Reserves	20min + 5% of initial fuel		No Credit	Maximum Endurance	Sea Level

The functional flow block diagram, in Figure 81, breaks down the mission segments into functions. Specifically, this FFBD focuses on the “Combat” phase. The vehicle must use an EO and radar concurrently to identify potential threats and targets. Furthermore, the mission requires a sufficient amount of excess power to achieve the speeds to make it less vulnerable to enemy activity.

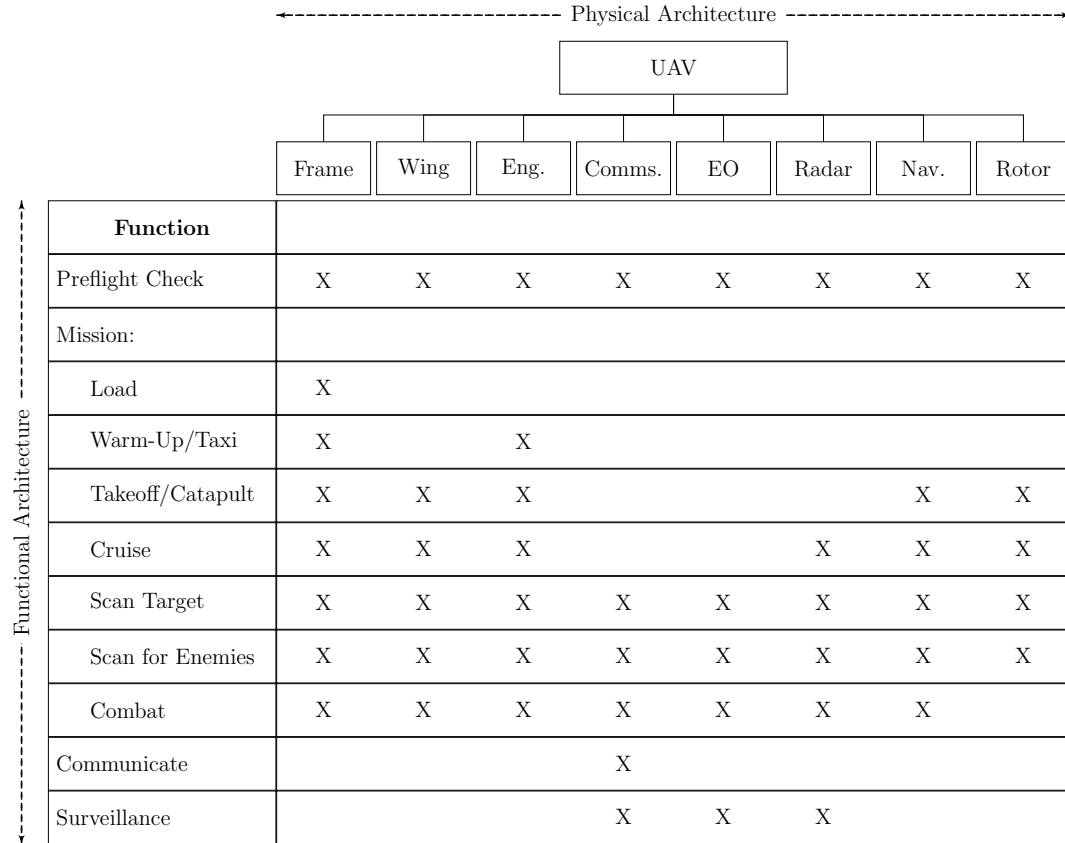


Figure 82: UAV Military-Grade Surveillance Mission Functional/Physical Matrix

Following the functional flow block diagram, a functional-physical matrix can assist in determining configuration options. There are certain phases in the mission that require specific components. The design must include those subsystems. Furthermore, a phase might have multiple component combinations. These should remain as options during the design process. Figure 82 shows the functional-physical breakdown of the military-grade, high-speed mission.

From the functional-physical matrix in Figure 82, the mission requires an EO sensor and radar. A rotor will not be able to achieve the desired speeds for the mission. A rotor's performance deteriorates at high speeds, making the use of a rotor impractical for this type of mission.

The high-speed characteristics of the mission drive the engine to be either a turbofan or turbojet. These types of engines create the necessary thrust to meet the

required performance.

5.2.2 Relating Design Missions to UAV Configuration

Combining the results in Figure 78, 80, and 82, designers can select the configurations of all three designs. The first two missions are very similar. The main differences between the two are the required range, endurance, altitudes, and payloads. However, the configurations are primarily the same. The third mission requires substantially more power compared to the first two. The excess power allows the design to achieve higher speeds, reducing the vehicle's vulnerability to enemy activity. Though the first two designs could use a rotor to provide lift, including a rotor is not practical on the production line. The rotor would require specific process and integration activities. Instead, the manufacturer could take advantage of a standard configuration across all systems to increase the likelihood of common components and interfaces, further reducing costs. Since the manufacturer is already expecting to outsource the production of the engines and electronics, the manufacturer is more flexible to implement different combinations of these subsystems. Thus, the configuration for all three designs is a tube-body-wing with various engines and subsystems depending on the mission.

5.2.3 Determination of Drivers Relevant to UAV Design

Of the product architecture selection drivers identified in Section 3.2.1.5, the primary drivers for the UAV industry are design requirements, market considerations, and technologies. Specifically, the range/endurance, speed, payload, acquisition cost, and electronics incorporated all have a significant impact on the design and choice of architecture. Many of the historical designs have lasted within the industry for over ten years. Manufacturers implemented any changes through upgrades of the original designs. Therefore, life cycle considerations are not as necessary. Thus, the following analysis must consider the identified requirements.

5.2.4 Conclusions from Defining the UAV Design Problem

This section developed three design missions that reflected the customers needs identified in the formulation of the manufacturer's customer-oriented, product-based strategy. Furthermore, the analysis set the configuration to be a tube-body-wing with various engines and electronics incorporated in the design depending on the mission. The process ensured the requirements were met. It also created constraints on the designs to meet the desires of the customers in the competitive marketplace. The next step in this process is to determine what makes a specific product architecture valuable, providing the ability to compare multiple product architecture alternatives. The comparison will lead to the final decision on which product architecture to implement.

5.3 *Establishing a “Valuable” UAV Product Architecture*

The next step requires the determination of weighting between metrics. For this case, desirability, flexibility, and complexity are the three primary metrics. First, desirability depends on the customers' desires and needs. Also, flexibility and complexity require the gradient and Hessian of a pseudo-objective function which is a combination of the desirability and penalty functions associated with various constraints of the design. Therefore, the derivation of desirability must come first.

A Quality Function Deployment (QFD) produces an objective function by combining the importance of customer needs, functional requirements and the relations between them [122]. Correlations were calculated between a series of performance and cost metrics of existing UAV designs. These correlations will help in the formulation of an OEC for each mission. Table 57 shows the value of these correlations.

The civil and military-grade surveillance missions are structured the same way. Though, the magnitudes of the needs and functional requirements differ, the relations stay relatively constant. Therefore the objective functions for both missions can be

Table 57: Correlations between UAV Performance and Cost Metrics

	Fuel (lbs)	Number Built	Unit Price 2017	Range (mi)	Endurance (hrs)	Max Speed (M)
Payload (lbs)	0.76	-0.25	0.74	0.49	0.40	0.50
Fuel (lbs)		-0.53	0.90	0.65	0.63	0.64
Number Built			-0.42	-0.16	-0.19	-0.30
Unit Price 2017				0.92	0.78	0.46
Range (mi)					0.65	0.29
Endurance (hrs)						0.17

derived from the same QFD. Table 58 depicts the QFD analysis of the civil and military-grade surveillance missions.

Table 58: UAV Civil and Military-Grade Surveillance Missions QFD

	Importance	Payload Weight	Fuel Weight	Endurance	Acq. Cost
Altitude to					
Detect Target	1	0.6	0.2	0.1	0.4
Range	2	0.49	0.65	0.65	0.92
Endurance	4	0.40	0.63	1	0.78
Num. Produce	3	-0.25	-0.53	-0.19	-0.42
Target (Civil)		100 lbs.	100 lbs.	10 hrs.	\$50K
Target (Mil)		200 lbs.	200 lbs.	10 hrs.	\$5mil
Absolute					
Importance		3.93	5.61	5.97	6.62
Relative					
Importance		0.18	0.25	0.27	0.30

In the QFD for the surveillance missions, the primary customer demands are the altitude to run scans of the target area, the range, endurance, and number demanded, and the functional requirements are the payload weight, fuel weight, endurance, and the acquisition cost. After providing weightings to each of the customer's needs and providing sensitivities among the customer needs and function requirements, absolute and relative importance of each functional requirement can be calculated. Combined with targets the objective functions for the two missions can be created.

Civil Surveillance Mission:

$$\phi = 0.18 \frac{W_P}{100} + 0.25 \frac{100}{W_F} + 0.27 \frac{E}{10} + 0.30 \frac{50K}{Cost_{Acq}} \quad (173)$$

Military-Grade Surveillance Mission:

$$\phi = 0.18 \frac{W_P}{200} + 0.25 \frac{200}{W_F} + 0.27 \frac{E}{10} + 0.30 \frac{5mil}{Cost_{Acq}} \quad (174)$$

Next, QFD analysis must provide the objective function for the military-grade, high-speed mission. Table 59 depicts the QFD analysis of the civil and military-grade surveillance missions.

Table 59: UAV Military-Grade High-Speed Mission QFD

	Import.	Payload Weight	Fuel Weight	Range	Max Speed	Acq. Cost
Distance to Detect Target	4	0.6	0.2	0.1	0.5	0.2
Range	2	0.49	0.65	1	0.29	0.92
Endurance	1	0.40	0.63	0.65	0.17	-0.19
Num. Produce Target	3	-0.25	-0.53	-0.16	-0.30	-0.42
		100 lbs.	300 lbs.	350 mi.	1.2 Mach	\$100mil
Absolute Importance		4.53	4.30	3.53	2.9	2.6
Relative Importance		0.23	0.21	0.18	0.18	0.20

In the QFD for the high-speed mission, the primary customer demands are the distance to detect targets/enemies, the range, endurance, and number demanded. The functional requirements are the payload weight, fuel weight, range, maximum speed and the acquisition cost. After providing weightings to each of the customers needs and providing sensitivities among the customer needs and function requirements absolute and relative importance of each functional requirement can be calculated. Combined with targets the objective functions for the two missions can be created.

Military-Grade High-Speed Mission:

$$\phi = 0.23 \frac{W_P}{100} + 0.21 \frac{300}{W_F} + 0.18 \frac{R}{750} + 0.18 \frac{V_{Max}}{1.2 Mach} + 0.20 \frac{100 mil}{Cost_{Acq}} \quad (175)$$

The next step in the process is determining the weightings for the three primary product architecture evaluation metrics. As shown in Section 3.3, the designer should ask the following six questions to provide general weightings. These weightings act more as guides to depict the general area where to search for product architectures.

- Desirability

1. How much power do the customers have? (**High Power - 3**) As stated in Section 5.1.3, the competitive marketplace that exists for the UAV industry creates a significant amount of power for the customers since they can switch producers relatively quickly, due to the abundance of choice.
2. How many requirement thresholds must the product achieve? (**Moderate Number - 2**) There were no significant considerations added in this case example, therefore, the number of thresholds, in this case, is moderate.

- Flexibility

1. How long is a product's traditional life span in the industry? (**Long Span - 3**) Many of the current UAVs in the market have existed for up to or even more than ten years, a relatively long time concerning the acquisition cost in the aerospace industry.
2. What is the cost to develop and produce a new product? (**High Cost - 3**) There is a high cost associated with the production of a new design. Processes and integration techniques cannot transfer over to an entirely new design. Therefore, ramping up production for a different product line would be extremely costly for the manufacturer.

- Complexity

1. What is the manufacturer's novelty producing a product? (**Low Novelty - 1**) The expectation is that the engineering team has plenty of experience in this field reduces the novelty of the employees concerning this project.
2. How many domains are associating with developing a new product? (**High Number - 3**) The number of domains and disciplines associated with producing this production line is high since development of a UAV requires many different disciplines.

The results of this analysis provide the following weightings for the three metrics: Desirability - 0.33, Flexibility - 0.4, and Complexity - 0.27. These are general directions and should act as guides. The product architecture space acts entirely different when compared to typical design problems. In this case, the highest priority should be given towards desirability followed by flexibility and then complexity.

5.4 Generating Alternative UAV Product Architectures

Producing alternative product architectures is not as easy as selecting index values of commonality, online reconfigurability, and offline reconfigurability. Instead, this process requires the possible production combinations. Each design requires a configuration, and each configuration requires an engine, wing, fuselage, horizontal and vertical tail, and various electronic subsystems. The process enforces commonality by giving each component a number varying from one to three. If two design's fuselages possess the same fuselage number, an equality constraint makes all the dimensions and characteristics of the two fuselages the same. Online reconfigurability options include variable wing sweep and pitch; inclusion of flaps, ailerons, rudders, and elevators; and variable propeller speed and pitch. During each design mission segment, the controls for all variable component are optimized to create the best performance

for the vehicle throughout the mission. Offline reconfigurability is assumed to be two different subcomponents (engines); sharing the same interface with a common platform (fuselage).

The design variables are varied as well to provide distributions of the various metrics relating to the different product architectures. The ranges of the design variable originate from past UAV designs. Figure 83 shows the distributions of existing UAV's relevant variables.

The ranges attempt to capture the overall design space. Table 60 displays the design variables considered and their minimum and maximum values.

Table 60: UAV Design Variable Ranges

Design Variable	Minimum	Maximum
Wing Area (ft ²)	20	175
Wing AR	5	20
Wing taper	0.5	1
Wing Sweep (deg)	1	30
Fuselage Length (ft)	5	30
Tail Arm (ft)	Fuse. Length / 2	
Horsepower ¹	25	525
Thrust (lbs) ¹	300	11,000
BPR ¹	2	10
Additional Payload (lbs)	10	100
Number Produced	10	300

¹ Depends on Engine Type

Combinations of various inputs create product architectures throughout the space. Since the experiments focus on the impact of commonality and reconfigurability on the product architecture's desirability, flexibility, and complexity, a Monte Carlo of the space was run to obtain distributions of each metric. The distributions will be used to determine the relationships amongst a product architecture's characteristics, the drivers and the metrics of interest. The next section will detail the tools used to size the vehicles and calculate the metrics of interest. Furthermore, it will introduce the results of the experiments showing the relations relevant in product architecture

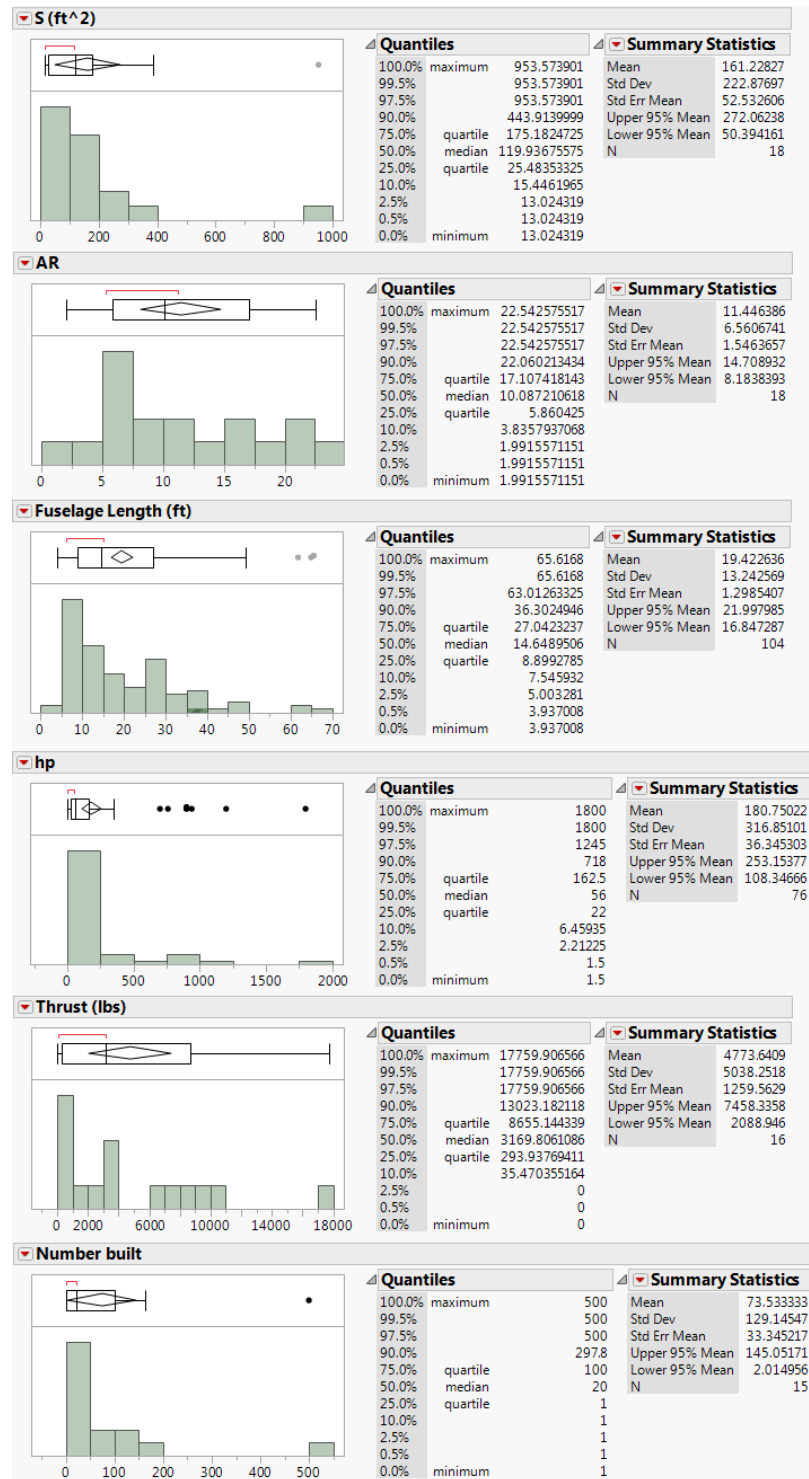


Figure 83: Design Variable Ranges Historical UAVs

selection.

5.5 *Evaluating Alternative UAV Product Architectures*

The evaluation of product architectures requires examination of many designs and configurations with the same architecture. The product architecture does not define the system's performance or cost but does influence it. The product architecture also has a relationship with the requirements. Switching the product architecture influences the constraints, relating to the flexibility and complexity of the system. For example, Figure 84 shows how an online reconfigurable wing changes the Mattingly (thrust required) constraint at different speeds and wing sweeps.

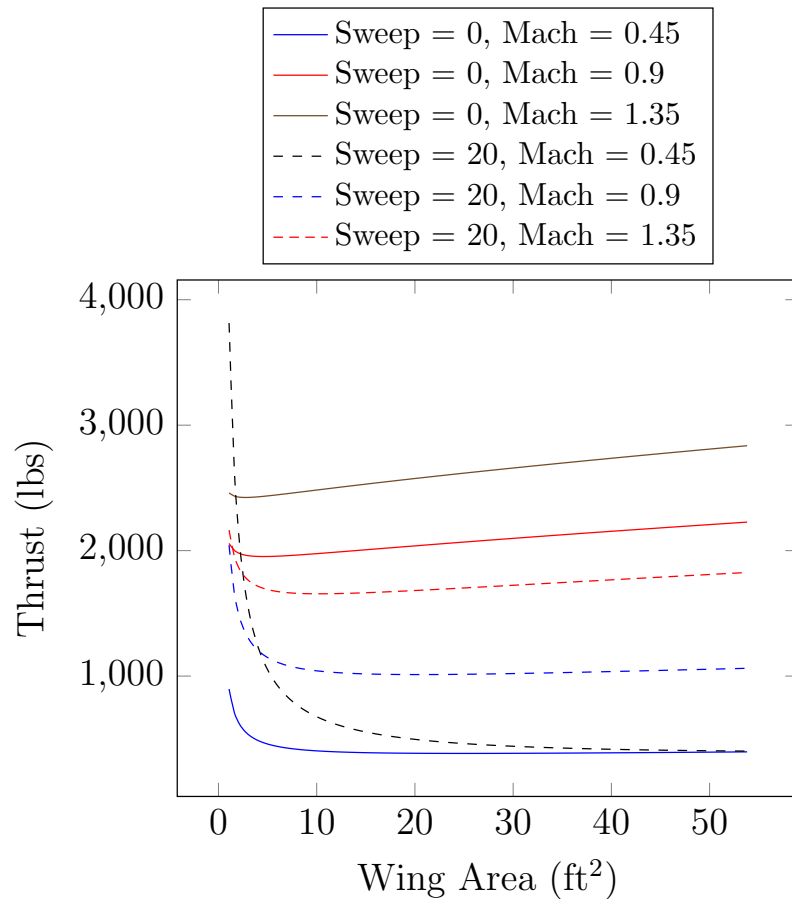


Figure 84: An Example of an Online Reconfigurable Wing's Impact on the Mattingly Constraint

Figure 84 demonstrates how the product architecture interacts with the design

and the functional requirements. Therefore, flexibility and complexity are used to determine the product architectures value by determining the impact of the architecture on the design space. Therefore, many designs within the space must be evaluated. The UAV sizing and synthesis tool FA²UST will provide this purpose.

5.5.1 Experiment 1: Testing the Evaluation Metrics Results and Conclusions

The second experiment analyzes many of the assumptions made about implementing certain types of product architectures. Of the claims, the predominant assertions are:

1. Designers use **fixed components** to design elements specifically for one role, thus increasing the desirability of the product.
2. Designers use **offline reconfigurable components** to change the role of the system, thereby increasing the requirement flexibility of the product.
3. Designers use **online reconfigurable components** to enhance the performance of the system in various conditions, thus increasing the requirement flexibility of the product.
4. Designers use **common components** to reduce costs by sharing manufacturing processes, thus reducing the design complexity of the product.

From the generation of alternative product architectures, the desirability, requirement flexibility, and design complexity of the alternatives were calculated. From the results, trends could be fitted to approximate each of the characteristic's impact on each of the metrics. Figures 85, 86, and 87 displays these trends.

Figure 85 shows each of the characteristics impacts on desirability. It uses a box plot to show the quartiles of the distributions of desirability, excluding outliers. The points represent the medians and the lines represent the first and fourth quartiles. First, commonality starts to increase the desirability of the product architecture by

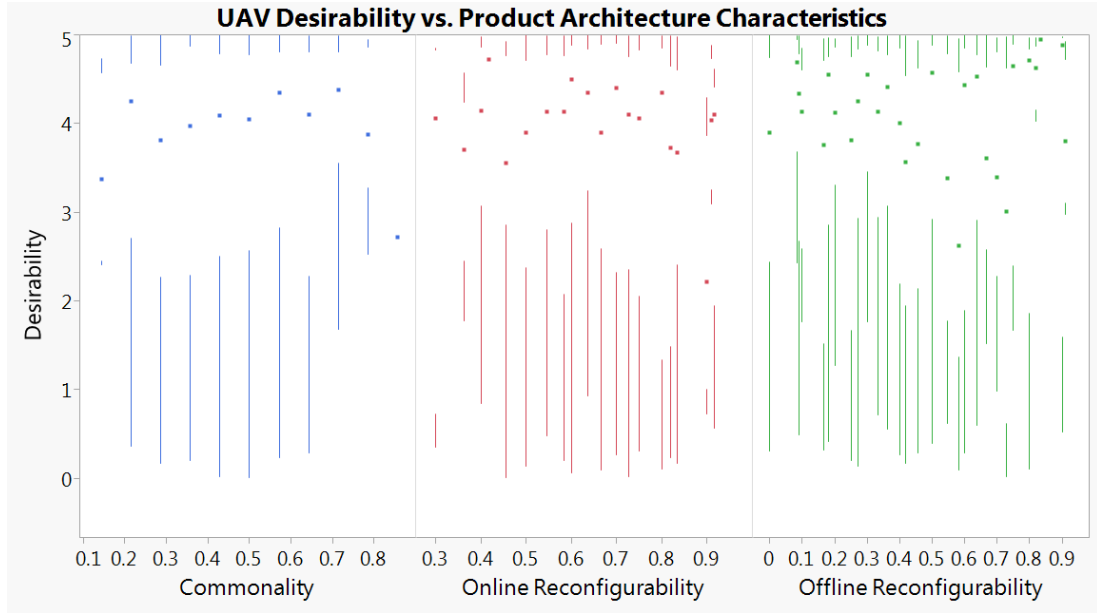


Figure 85: The Product Architecture's Impact on Desirability

reducing the cost of the system, but at a certain point, the desirability begins to decrease as commonality starts to hurt the performance of the product line. Second, online reconfigurability seems to have no improvement on the desirability. This phenomenon will be addressed later. Third, offline reconfigurability at first decreases desirability, but as offline reconfigurability increases, the desirability of the product begins to increase as well. At low offline reconfigurability, the additional interface constraints require over designing the platform to handle all offline reconfigurable components. As a designer adds more offline reconfigurable components, the ability for a design to change its components depending on the vehicle's purpose will both reduce the cost and increase the performance. The unique subcomponents are designed for a specific mission while the common platform reduces cost.

Figure 86 determines each of the characteristics impacts on requirement flexibility. It uses a box plot to show the quartiles of the distributions of requirement flexibility, excluding outliers. The points represent the medians and the lines represent the first and fourth quartiles. First, commonality decreases the requirement flexibility of the product architecture since the number of constraints on each common component

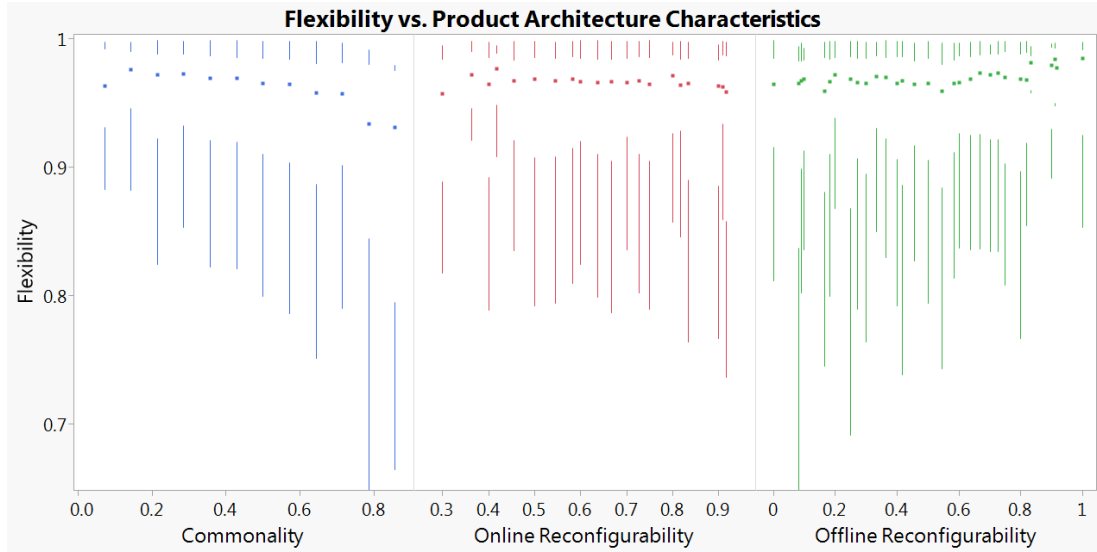


Figure 86: The Product Architecture's Impact on Flexibility

increase. The increase of constraints gives less design freedom to the designers concerning requirements. Second, online reconfigurability has a minimal impact on the requirement flexibility of the system. The reasoning will be addressed later. Third, offline reconfigurability increases the requirement flexibility of the product architecture since components can be swapped depending on its purpose, making it elusive to changing requirements.

Figure 87 displays each of the characteristics impacts on design complexity. It uses a box plot to show the quartiles of the distributions of design complexity, excluding outliers. The points represent the medians and the lines represent the first and fourth quartiles. First, commonality decreases the design complexity of the product since it reduces the number of processes required for production. However, there is an increased risk associated with commonality. The higher number of constraints added to the common components design can create more coupling between it and other components. Thus, commonality can increase design complexity. Figure 87 demonstrates this from the tails associated with the error of the fit between commonality and design complexity. Second, online reconfigurability has little impact

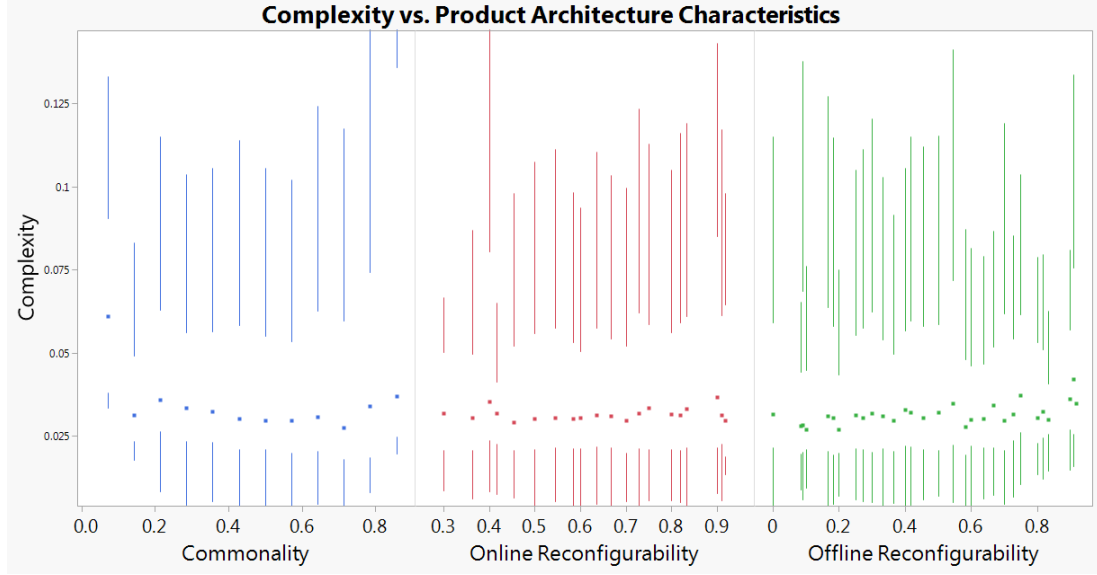


Figure 87: The Product Architecture's Impact on Complexity

on design complexity. The reasoning will be explained next. At first, offline reconfigurability decreases design complexity. However, as the designer implements offline reconfigurable components, the design complexity starts to increase. Offline reconfigurability requires interface constraints that create couplings between the platform and subcomponents, increasing the sensitivities of design variables.

Online reconfigurability, independently, had little impact on the desirability, requirement flexibility, and design complexity of the product line. However, the interactions amongst product architecture characteristics provide additional insight. Online reconfigurability is heavily dependent on the other product architecture characteristics. Table 61 shows the impact of the first and second order terms of the evaluation metrics. The interaction terms relating to online reconfigurability have more significant impact on the evaluation metrics than the independent terms. Therefore, online reconfigurability's impact on the evaluation metrics is dependent on the other architectural characteristics.

Table 62 shows the online reconfigurability interaction terms alongside their impact.

Table 61: First and Second Order Term's Impact on Evaluation Metrics

	Desirability	Requirement Flexibility Percent Impact	Design Complexity
Commonality	26.2	45.9	2.9
Online Recon.	3.3	2.9	0.1
Offline Recon.	6.1	12.0	4.8
Commonality x Commonality	37.3	5.4	0.1
Online x Online Recon.	4.5	3.3	0.5
Offline x Offline Recon.	9.5	2.7	8.9
Commonality x Online Recon.	3.8	13.1	10.0
Commonality x Offline Recon.	0.9	0.8	2.3
Online x Offline Recon.	8.5	13.9	7.9

Table 62: Online Reconfigurability's Interaction Term's and their Impact on Evaluation Metrics

		Requirement Desirability Percent Impact	Requirement Flexibility Percent Impact	Design Complexity Percent Impact
Online Recon. x	Coeff.	Coeff.	Coeff.	Coeff.
Commonality	3.00	3.8	0.044	13.1
Offline Recon.	-3.12	8.5	0.11	13.9

The coefficients suggest a couple of insights. First, online reconfigurability paired with commonality tends to increase desirability, increase requirement flexibility, and decrease design complexity. The increase in desirability is likely due to the increase in performance from online reconfigurability to compensate for the losses in performance from commonality. The increase in requirement flexibility and the decrease in design complexity are likely due to online reconfigurability's ability to adapt to conditions decreasing the impact of requirements on the product line and decreasing the coupling between components. Second, online reconfigurability paired with offline reconfigurability tends to decrease desirability, increase requirement flexibility, and decrease design complexity. The decrease in desirability is likely due to cost factors since both types of characteristics tend to increase cost. The increase in requirement flexibility

and decrease in design complexity are likely due to online reconfigurability's ability to adapt to conditions decreasing the impact of requirements on the product line and decreasing the coupling between components, either compounding or mitigating impacts from offline reconfigurability.

The results from this experiment isolate the impact of each characteristic on the evaluation metrics and allow us to address the legacy assumptions:

1. *Fixed components increase the desirability of a product.* - To a degree. Designers use fixed components for a specific mission, which makes them optimal for performance, but producing individual components in a product line drive up costs which could system's desirability depending on the customer.
2. *Offline reconfigurable components increase the requirement flexibility of a product.* - Yes, this is true. The requirement flexibility of a product architecture appears to increase as designers implement more offline reconfigurability in the design. This phenomenon is due to the ability to design specific subcomponents for each mission while still maintaining favorable cost conditions. However, the implementation of offline reconfigurability has an expected increase in design complexity.
3. *Online reconfigurable components increase the requirement flexibility of a product.* - No. Online reconfigurability's impact on the product is too dependent on requirements and other product architecture characteristics. The lack of a clear independent trend refutes this claim.
4. *Common components reduce the design complexity of a product.* - Yes. There is clear evidence to support this. However, the claim fails to mention the drawbacks of using commonality, such as deteriorating performance and requirement flexibility of the product. These drawbacks could offset the gains.

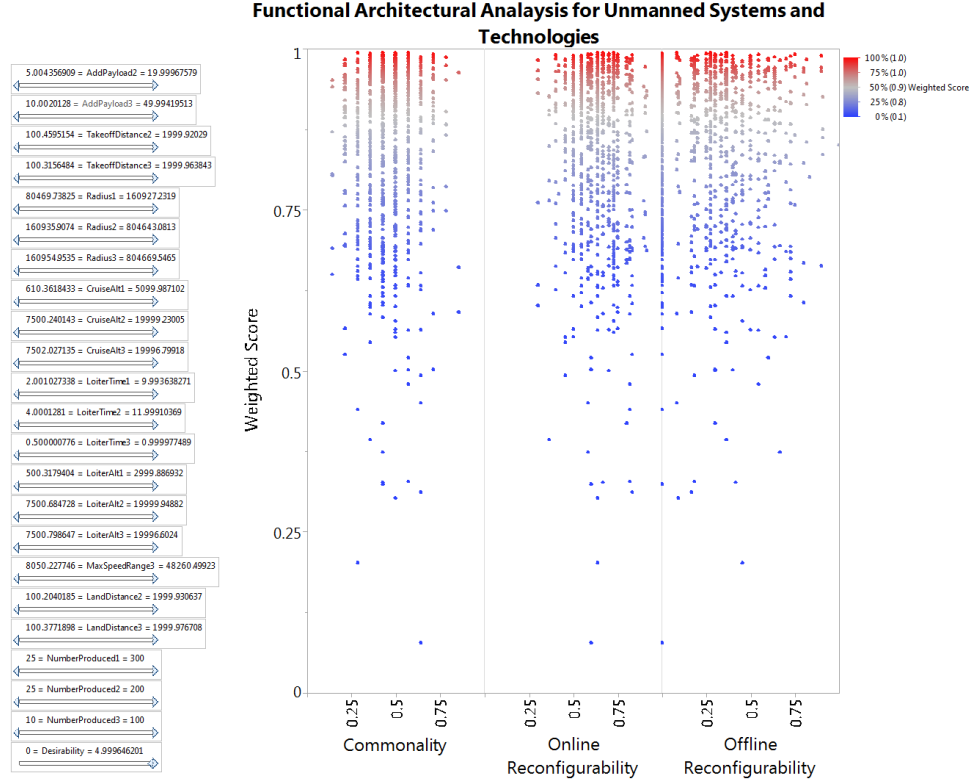


Figure 88: The FA²UST Framework Decision Facilitator

The lack of clear consensus with these claims provides evidence why product architecture analysis is crucial in the systems engineering process. The provided framework sets up the problem and provides the steps necessary to challenge many of the usual assumptions made during this process.

5.6 Final Decision of UAV Product Architecture to Implement

The final step in the framework requires a decision to be made on the composition of the product architecture's characteristics. The designer can take the weighting of each metric of interest to create an overall weighted score, for each alternative. Figure 88 shows all the possible combinations for this case study. Using statistical software ranges of requirements can be narrowed down to appropriately filter through the data and select the most appropriate architecture.

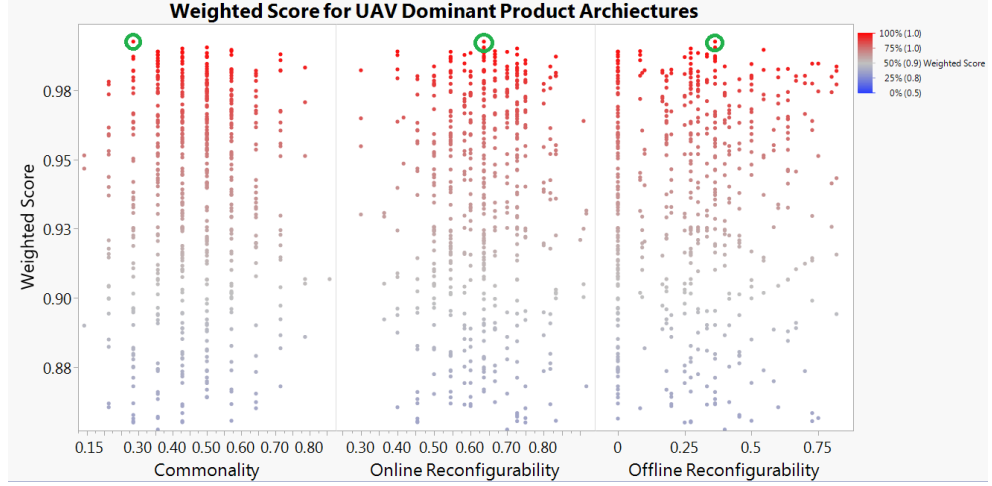


Figure 89: Final Decision of UAV Product Architecture

For this case study, the requirements were narrowed down based on the functional requirement and overall evaluation criteria. Figure 89 shows the remaining options left to the decision maker.

The decision maker should identify regions where there are many choices with high weighted scores. The ability to identify regions rather than specific designs increases design freedom and reduces the risk of unforeseen considerations hurting the development of the product and implementation of the product architecture. Thus, the recommended product architecture for this case consists of 29% commonality, 64% online reconfigurability, and 36% offline reconfigurability.

The selected product architecture contains three wings, two horizontal tails, three vertical tails, two fuselages, two engines, two EO-IR sensors, and two radars. Also, the product architecture uses some variable sweep wings, flaps, variable pitch propellers. Table 63 displays the selected product architecture's composition of components for each product in the product line, and Table 64 displays the selected product architecture's composition of online reconfigurability for each product in the product line. Table 63 shows certain aspects about the product architecture. The wing, vertical tail, and EO-IR sensor can be constructed modularly and the horizontal tail, fuselage, engine, and EO-IR sensor have some commonality. Also, the wing has variable sweep

in products 1 and 2 and uses flaps in product 2. Finally, the turboprop in products one and two uses a variable pitch propeller.

Table 63: Composition of Components for Final Selected Product Architecture

	Product 1	Product 2	Product 3
Wing	1	2	3
Horizontal Tail	1	1	2
Vertical Tail	1	2	3
Fuselage	1	1	2
Engine	1	1	2
Engine Type	Turboprop	Turboprop	Turbojet
EO-IR	1	2	2
Radar	None	1	2

Table 64: Composition of Online Reconfigurable Interfaces for Final Selected Product Architecture

	Product 1	Product 2	Product 3
Variable Sweep	1	1	0
Flaps	0	1	0
Variable Pitch Wing	0	0	0
Variable Pitch Propeller	1	1	0
Thrust Vectoring	0	0	0

5.6.1 Experiment 2: Sensitivity Studies on the Evaluation Metric Weightings Results and Conclusions

The framework proposes using a weighting system that analyzes the more qualitative aspects of the problem. Due to their qualitative nature, sensitivities of each on the selection process, sensitivity studies were conducted. The product architecture composition selected in the case study consisted of 29% commonality, 64% online reconfigurability, and 36% offline reconfigurability. Three cases were run in this experiment to allow for the impact to be observed. The cases were purely looking at the desirability, requirement flexibility, and design complexity metrics as a means to make a decision.

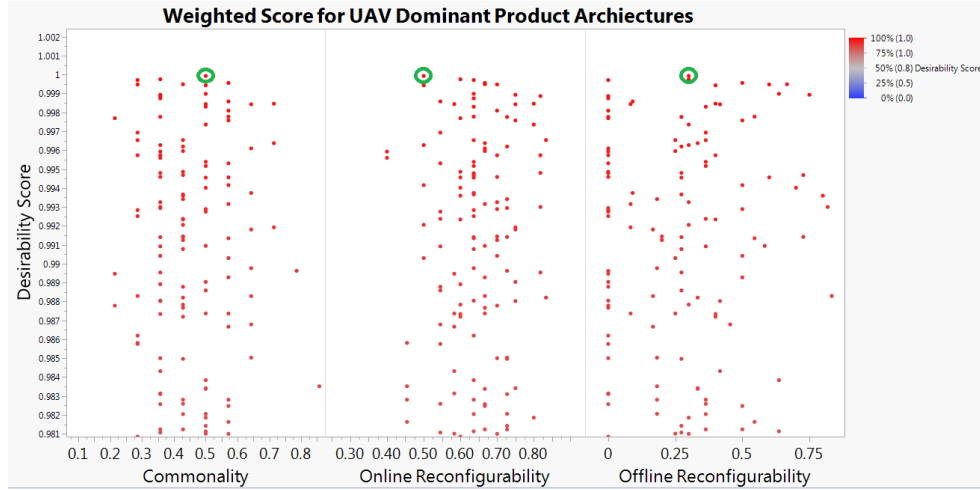


Figure 90: UAV Product Architecture Selection based on Desirability

The first case looked only at the desirability of the product architecture. Figure 90 displays the result of the case where the most favorable composition regarding this weighting consists of 50% commonality, 50% online reconfigurability, and 30% offline reconfigurability. The loss of requirement flexibility and design complexity weightings makes the most favorable product architecture have more commonality and a little less reconfigurability. Commonality reduces the predicted cost of the product line, but the selected product architecture's lower requirement flexibility and higher design complexity increase the risk of unforeseen errors during development and production. With respect to the weightings, the high commonality can make the vehicle difficult to upgrade and increase the number of constraints on the common components. These aspects require the design team to work together efficiently. Experience and the number of departments directly impact work cohesion. Therefore, the selected product architecture reflects the weightings in Section 3.3.

The first case looked only at the requirement flexibility of the product architecture. Figure 91 displays the result of the case where the most favorable composition regarding this weighting consists of 21% commonality, 75% online reconfigurability, and 0% offline reconfigurability. The loss of desirability and design complexity weightings makes the most favorable product architecture have more online reconfigurability

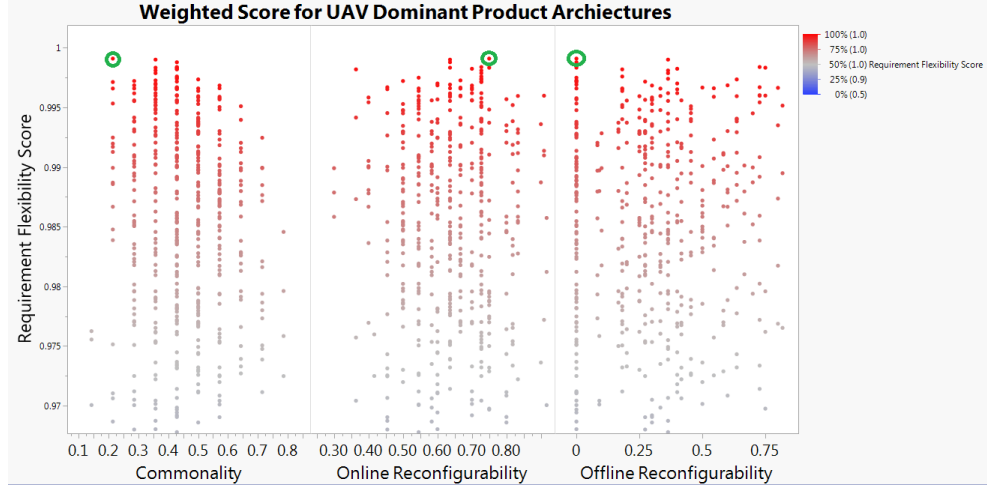


Figure 91: UAV Product Architecture Selection based on Requirement Flexibility

and no offline reconfigurability. Online reconfigurability essentially makes the product insensitive to changing performance requirements reflecting the drastic increase of online reconfigurability. With the respect to the weightings, if the product line is expecting to have a short lifetime, this product architecture will have no easy way to upgrade components with modular or offline reconfigurable parts. Also, online reconfigurable interfaces can be incredibly complex. The interfaces require multiple disciplines, expertise in controls, and a deep understanding of the design problem. Therefore, the selected product architecture reflects the weightings in Section 3.3.

The first case looked only at the design complexity of the product architecture. Figure 92 displays the result of the case where the most favorable composition regarding this weighting consists of 50% commonality, 64% online reconfigurability, and 18% offline reconfigurability. The loss of desirability and requirement flexibility weightings makes the most favorable product architecture have more commonality and a less offline reconfigurability. The increase in commonality decreases the number of components and the decrease in offline reconfigurability reduces the number of common interfaces. Both of the aspects reduce the design complexity of the system. However, with respect to the weightings, the increase in commonality and decrease in offline reconfigurability reduce the individuality of each product in the product line.

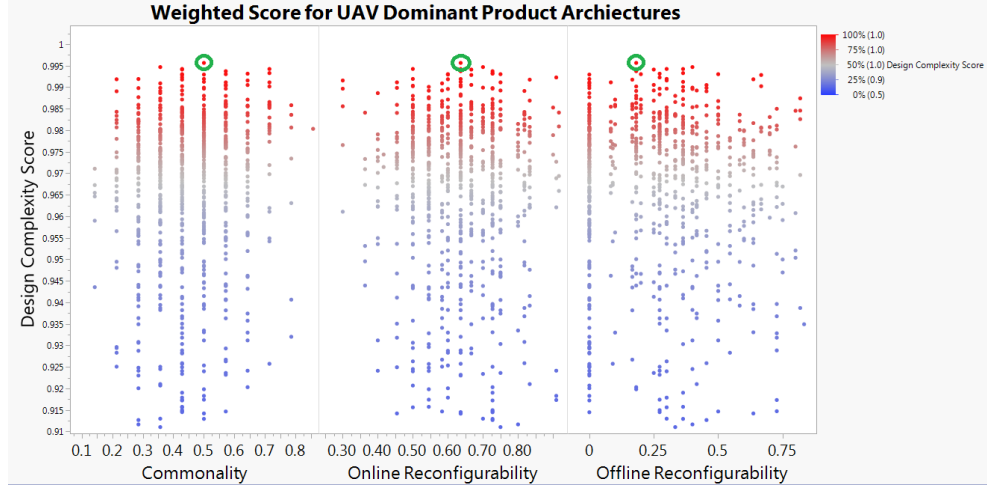


Figure 92: UAV Product Architecture Selection based on Design Complexity

This might be a negative trait according to the customer as all products might look similar and the products might not achieve the requirements as optimally. Also, the high commonality can make the vehicle difficult to upgrade making the product architecture much more sensitive to changing requirements. Therefore, the selected product architecture reflects the weightings in Section 3.3.

Each case in this experiment shows the impact of the weightings on the selected product architecture. Furthermore, it shows how the qualitative concepts of the problem impact the decisions made later on in the framework, proving the qualitative weightings are relevant concerning product architecture selection.

5.6.2 Experiment 3: Observing the Impact of Requirements on the Product Architecture Results and Conclusions

The first experiment analyzes the claim that a product architecture can be determined early in the design process by looking at a few specific requirements. Of those identified, they can be categorized by performance, production cost, technologies, and life cycle considerations. The experiment primarily focused on the performance and cost metrics since they are significantly applicable to the case study. Life cycle considerations were included in the cost model. Their primary driver is the number

of vehicles produced which relate to supporting the vehicles during their lifetime. Technologies were incorporated into the subsystems sizing. The subsystem sizing requires range and resolution for the radar and the EO sensor. Thus, altitude drives the weight and power required by the sensors since better range per resolution relates to more advanced electronic technologies.

During the generation of alternative product architectures, the requirements were varied as well. From the data, the extremely undesirable cases (ones that did not converge during sizing or missed the constraints severely) were ignored, and the product architecture compositions were plotted against the functional requirements in Figure 93.

The data suggests the functional requirements can be grouped into three categories power, energy, and cost. Takeoff distance, speed, and altitude act similarly and correlate with required excess power. Payload-range is a good representation of weight and the energy required to complete the mission. Finally, the number produced directly relates to cost since designing a UAV requires a lot of fixed cost. Producing more allows the fixed cost to be amortized over more vehicles. Hidden in these metrics are technologies and life-cycle considerations. The altitude and range influence the subsystem technologies. The radar and EO sensors' range versus resolution capabilities is dependent on these requirements. The capabilities determine the weight of the sensors, which drive the payload range required of the design, and the cost includes life-cycle considerations. Understanding how the technologies and life-cycle considerations influence the power, energy, and cost requirements can determine their impact on the product architectures. Combining the requirements reduces the dimensionality of the problem, simplifying the analysis required.

Figure 93 displays the individual trends between the functional requirements (negative takeoff distance, average altitude, payload-range, and the number produced) and the product architecture indices. The functional requirements include the average

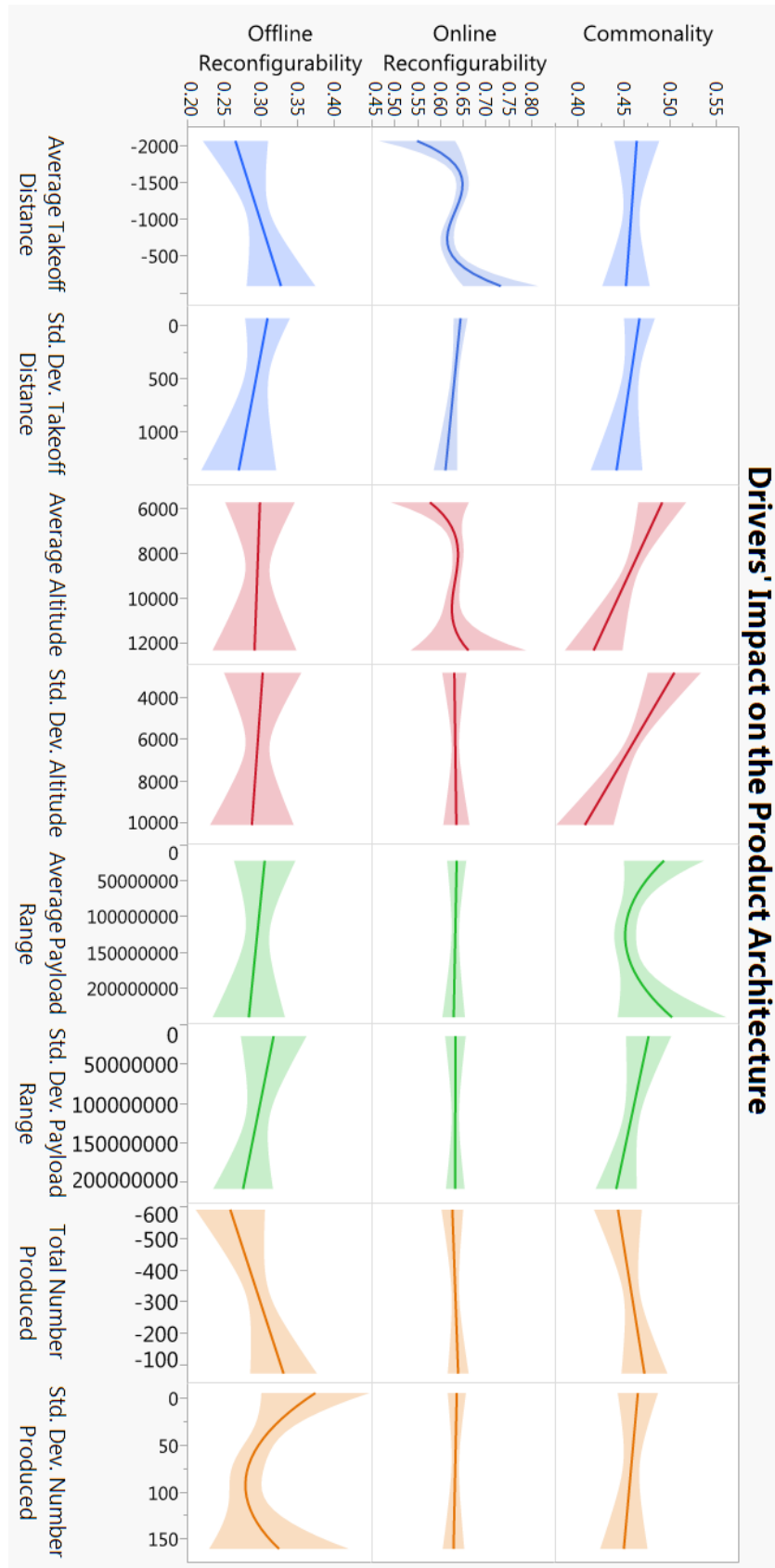


Figure 93: The Requirements' Impact on Product Architecture Evaluation Metrics

and standard deviation over the product line. Figure 93 suggests the requirements influence on the product architecture are primarily dependent on the design variables. However, at extremes of the requirement ranges, there is some impact on the product architecture.

First looking at takeoff field length, the shorter the takeoff distance, the more constraining the requirement becomes. Therefore, the short takeoff distances require the product to be more specifically designed for each mission, driving more online reconfigurability and offline reconfigurability. These trends make sense as the tight performance constraint drives the vehicles to favor flaps, variable sweep wings, and mission-specifically designed components. The standard deviation between takeoff distances for each product has relatively little impact on the product architecture.

Second, the average altitude requirements primarily impact the commonality of the product architecture. As more excess power is required by the system at higher altitude, the commonality's emergent detriment on performance becomes apparent. Therefore, as the average altitude requirements increase, the product architecture incorporates less commonality. The standard deviation of the altitude requirements also impacts the commonality of the product architecture. As the difference between altitude requirements increases the differences in performance constraints prefer more unique components to be incorporated in the design.

Third, the average payload range of the product line impacts the commonality and offline reconfigurability of the product architecture. At high and low payload ranges, the product architecture favors high commonality. When the average payload range is high or low, this suggests all of the product's payload ranges are high or low. This trend reflects the payload range standard deviation's impact on commonality. At low standard deviations the product architecture favors commonality.

Fifth, the total volume of production impacts the offline reconfigurability and commonality of the product architecture. As less vehicles are produced, the product

architecture favors more of both to counter the inability to spread costs over greater production volumes. The standard deviation of the volume of production for each product impacts the offline reconfigurability of the product architecture. At low and high standard deviations the product architecture prefers higher offline reconfigurability. This trend is hard to decipher, but at high standard deviations the increase of offline reconfigurability suggests the products are sharing the same platforms or fuselages to mitigate costs for the lower demand products. Therefore, the use of various subsystems create more common interfaces and offline reconfigurability. At low standard deviations, the same but opposite effect could occur. The uniform distribution of demand suggests the products are free to incorporate more individual and favorable components. Therefore, when the product line uses a common fuselage to offset the increase in costs then offline reconfigurability be high.

The requirements have some impact on each product architecture characteristic. To get a better understanding how the requirements impact the overall product architecture space, regressions of the requirements using the product architecture characteristic were created. The data was filtered down to 56 Pareto dominant designs. The Pareto frontier was formed from maximizing desirability, maximizing requirement flexibility, and minimizing design complexity. The 56 designs were used to fit third degree polynomial with second degree interactions. These regressions were used to create the driver impact mappings displayed in Figure 94. In each driver impact mapping, the top of the circle reflects offline reconfigurability, moving one third the circumference clockwise from the top reflects commonality, and moving one third the circumference counterclockwise from the top reflects online reconfigurability.

The driver impact mappings show hot spots where larger requirement metrics are favored in certain regions of the product architecture space. This shows if a requirement dominate the design process a which product architecture composition should be favored.

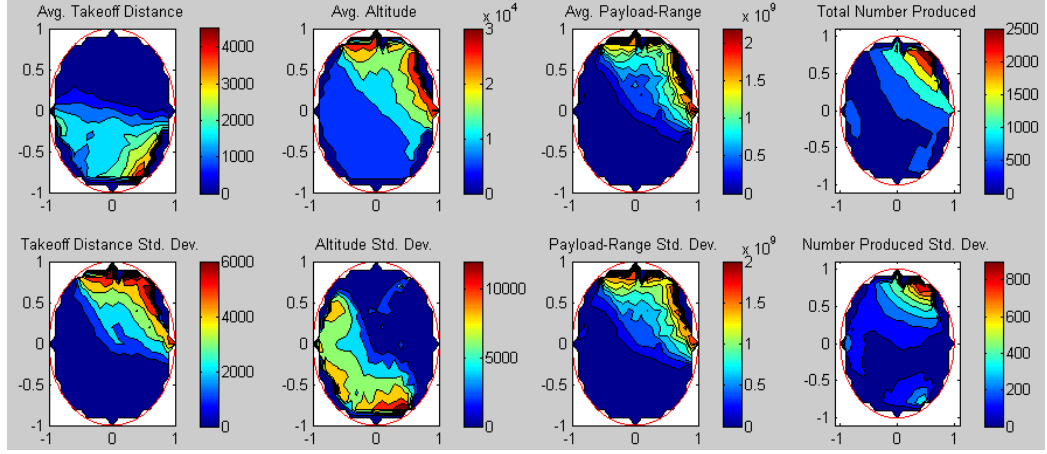


Figure 94: The Final Driver Impact Mappings for the UAV Case Study

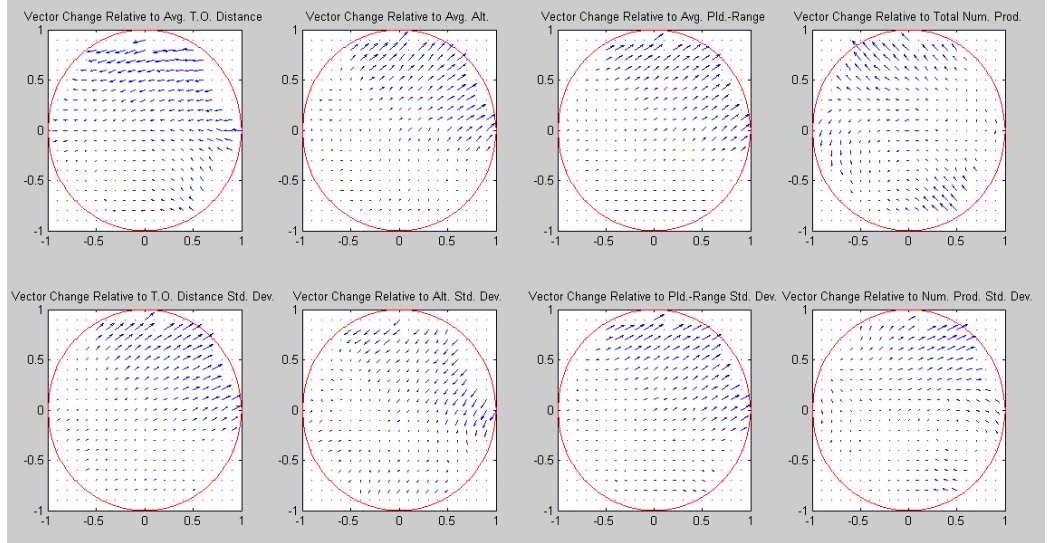


Figure 95: The Final Driver Sensitivity Mappings for the UAV Case Study

Figure 95 shows a different way of looking at the problem. Based on where the product architecture current lies, if a requirement changes, the driver sensitivity mappings shows how the product architecture should be modified to meet these new set of requirements.

The results shown in this experiment prove the way the requirements are structured impact the selection of the product architecture. Also, it shows how time can modify what would be considered the most favorable product architecture if the requirement are time sensitive, as they often are.

All of the drivers identified in Section 3.2.2 impact the implemented product architecture to a certain degree. The techniques used for this experiment could be applied to any product to understand the trends associated with the requirements and product architecture.

5.6.3 Summary of Experiments 1, 2, & 3

Experiments 1, 2, and 3 look at the product architecture's impact on system engineering metrics such as desirability, flexibility, and complexity, the sensitivity of the product architecture selection decision to the weightings, and the individual impact of requirements on the product architecture. The product architecture characteristics each exhibit varying degrees of desirability, flexibility, and complexity. Thus, not all product architectures are equally favorable. The weighting system for the product architecture evaluation metrics has an impact on the decision. The user of the framework should consider carefully what weightings should be given to each metric. The requirements in Figure 93 can be broken down into four categories: performance, cost, technologies, and life-cycle considerations. Each requirement exhibits specific characteristics and influences implementation of different product architecture characteristics. However, one of the key insights from this research is the dependence between the requirements and the product architecture. In certain regions of the product architectural space, certain requirements dominate while in other regions other requirements dominate. Overall, the product architecture selection problem requires careful analysis to determine the composition of product architecture characteristics the engineer should implement.

CHAPTER VI

CONCLUSIONS

The framework presented in this research approaches product architecture selection to determine the characteristics implemented in the design that determines how components interact. Implementing new product architectures, such as reconfigurability and product families, adds steps to the systems engineering process. Traditionally, engineers made assumptions to determine which product architecture to implement. However, the lack of clarity that exists in determining the product architecture's impact on vehicle performance and design requires structure approaches to tackle the problem. The framework implements characteristics of business strategy and systems engineering to provide a balanced understanding of the problem and how the problem architecture meets the customer's desires and fits in a manufacturer's agenda. The framework's structure involves:

1. Determining the customer's needs
2. Deriving a customer-oriented, product-based strategy to meet these needs
3. Deriving the functional requirements of the product line
4. Establishing what a "good" product line means
5. Generating alternative architecture
6. Evaluating the results
7. Choosing a composition of commonality, online, and offline reconfigurability

The choice in the composition does not tie down the engineers and allows them to make modifications during the design process as they see fit.

First, a UAV case study demonstrated the use of the framework. The UAV industry is emerging and has diverse requirements, driving its development. These include tighter fiscal constraints and demand for several mission and performance capabilities. However, the lack of standards in the industry made the case a challenging exercise to test the principles of the framework. The framework was then applied to the automobile industry in the late 1970s to provide validation. During this time in the car industry, Japanese firms had shifted the landscape of the industry and put the established manufacturers in a frenzy. However, the car companies that could adjust ended up flourishing in the late 1980s and early 1990s.

6.1 Contributions

The new framework provided many contributions to the systems engineering and aerospace field. The contributions include:

- **FA²UST Framework: A framework that facilitates the cooperation between the systems engineers and business management.** The framework presents a method of determining a new products needs. The method combines business management analysis with standard systems engineering practices. The approach aligns both the business and engineering sides of the firm ensuring an understood and common goal. Design processes often overlook the cooperation between both departments. However, it should be an essential part of the development of the product and has enormous implications on the product architecture implemented.
- **FA²UST Module: A product architecture analysis module that can be integrated into the design of any type of product.** During the development of this framework, a module was developed the allowed for the analysis of a product architecture. Alongside the module, a UAV sizing tool was created that utilizes historical data and multi-disciplinary analysis tools to determine a

UAV's performance and cost. The contributions of the framework were added to allow for product architecture analysis alongside the traditional performance-cost analysis. This tool is easy to use and conducts the analysis quickly allowing for the entire product architecture space to be analyzed. The UAV sizing tool integrates into the module allowing for the product architecture analysis of a UAV. The module can also analyze other products from other industries as seen in Appendix A. The flexible use of the module makes it useful in any field when considering developing a new product line and deciding what levels of commonality and reconfigurability it should have.

- **A numerical representation of the product architecture space.** Product architecture selection historically has been a qualitative problem where the systems engineers make certain assumptions that make their selection of product architecture seem logical. However, in complex product development, even the most experienced engineers can fail to develop the correct assumptions. By converting the problem into a quantitative one, more analysis can be conducted using conceptual design tools to challenge these assumptions. The quantitative space uses a product's components' characteristics including commonality and reconfigurability to create ratios and a quantitative space.
- **New ways to calculate a design or product architecture's requirement flexibility and design complexity.** Product architectures impact a product's desirability, requirement flexibility, and design complexity. Traditionally, these terms have often been vague or qualitative. Thus, new numerical representations of these terms provide a means to compare alternatives quantitatively. Since engineers implement a product architecture to reduce interactions among requirements and coupling between components or disciplines, the metrics use the sensitivities and dynamics of the problem to determine the requirements'

interactions and design variables' coupling.

- **A means to down-select and identify product architectures of interest.**

After the analysis, the framework provides a method that allows for the down-selection of product architectures and the final decision. The down-selection involves filtering designs that meet specific capabilities and combination of desirability, flexibility and complexity metrics.

- **Essential insights about the relations between a product architecture and its desirability, flexibility and complexity.** The two experiments and validation case conducted in the framework's verification and validation provided insights that are useful to general product development. The trends were consistent in both cases suggesting their universal usefulness.

The contributions this framework provides allowed for product architecture analysis. The analysis identified essential insights that could be used in product architecture analysis in the future.

6.2 Important Insights from the Research

The new framework provided many contributions to the systems engineering and aerospace field. The contributions include:

- **Requirements that drive product architecture selection.** The analysis of past industries identified four categories of product architecture selection drivers: performance, development and production costs, life-cycle considerations, and technologies. These categories can be summarized by overarching terms that can be applied to most products. They include power (rate), energy (capacity), and cost (economics). These metrics seem to capture trends between the specific requirements and product architecture universally. Therefore, their impact should be well understood in a new product's development.

- **Commonality's impact on a product architecture's:**

- *Desirability* Commonality tends to have a maximum desirability associated with it. This trend relates to the priorities set on the performance and cost of the final product line. Initially, commonality shows an increase in product desirability, but at a certain point, the cost savings do not outweigh the loss in performance of the product line. Therefore, there is a specific region where desirability reaches a maximum.
- *Requirement Flexibility* Commonality tends to decrease the product's requirement flexibility to changing requirements. This trend is due to the more significant number of constraints that are applied to a shared component, making it more sensitive to changing requirements.
- *Design Complexity* Commonality tends to decrease the design complexity of the product. This trend relates to the lower number of production processes. However, depending on the design problem the higher number of constraints added to the component's design can cause greater design complexity since it will create more coupling connections.

- **Online Reconfigurability's impact on a product architecture's desirability, requirement flexibility, and design complexity.** Online Reconfigurability's impact on the evaluation metrics is dependent on the design problem. The data provided by the analysis shows no trends between it and the desirability, requirement flexibility, and design complexity. However, as the requirements are filtered down, trends become apparent, demonstrating the characteristic's dependence on requirements and the definition of the design problem.

- **Offline Reconfigurability's impact on a product architecture's:**

- *Desirability* Offline reconfigurability tends to increase the overall desirability of a product. This trend is likely due to the combination of common characteristics and components unique to a task.
- *Requirement Flexibility* Offline reconfigurability tends to increase the requirement flexibility of a product. The ability to swap components based on the task required makes the components elusive and less sensitive to changing requirements.
- *Design Complexity* Though offline reconfigurability shows gains in desirability and requirement flexibility, there exists a trade-off. Offline reconfigurability increases the design complexity of a product due to the more components required for design the inclusion of interface constraints that allow end-users of the product to swap variants of the components quickly. In the past, this trend was not previously analyzed or fully understood. This framework brings that trend to light.

The insights provide universal considerations and trends an engineer can use to get a better understanding of the product architecture selection problem before initializing the process. The more information available earlier in the process will reduce the risk of mistake and misinformed decision making. The increase in design knowledge is a crucial part of product architecture selection since so much of the development and product development depends on the decisions made during this phase. Thus, the insights and contributions provided by this research should be a welcoming benefit for the systems and aerospace engineering fields.

6.3 *Final Thoughts*

Historically, implementing commonality and reconfigurability has been a vague step in systems engineering. Specific examples, such as the F-35, Littoral Combat Ship, and automobile industry, demonstrate cases where engineers made assumptions about the

product architecture's impact on the development of a new product without testing them. As design considerations and constraints from customers and the industry continue to increase, it is essential to test these assumptions and analyze the problem before haphazardly implementing a product architecture characteristic. If anything should be gained from this dissertation, moderation is key. Either implementing all three or maximizing one has the unwanted effect of decreasing the requirement flexibility or increasing the design complexity of a product line. If the design problem requires all three, the firm should have schemes in place to mitigate the possible increase in design complexity. However, if the firm conducts analysis presented by this framework, it will gain an understanding of what to expect and possible ways to mitigate the risk.

APPENDIX A

AUTOMOBILE CASE STUDY

The validation case study analyzes the automobile industry in the late 1970s to early 1980s. There are multiple points in the industry's history that present interesting dynamics and witnessed changes in the industry-wide, implemented product architecture. One example is the emergence of Henry Ford and his products that used standardized parts and standard processes to reduce the cost of an automobile. Another example would be the recent adoption of electronic and hybrid technologies. However, these instances follow clear trends in the industry. The case requires an understanding of the market and engineering practices. At the time, automobile development followed a trend of increasing commonality, modularity, and standards. Additionally, Toyota and Honda began to disrupt the industry with lower-cost vehicles with higher reliability in comparison to the competition. Automobile production methods were changing during this period. These changes had a direct impact on the product architectures implemented.

The automobile manufacturer, in this case, is an American Company who has to deal with the changing paradigm in the industry. The manufacturer had to formulate a new strategy and reflect that strategy in the product offered. The product architecture that dominated the industry following this period maintained common processes and standardized parts. However, the increased cost savings from the new manufacturing techniques allowed the manufacturers to offer products with more unique parts allowing for better performance. The following case study goes through the process to determine whether the framework can come to the same conclusion as the resultant product architecture implemented in the 1980s and 1990s.

A.1 Establishing the Need for a New Automobile Product

The first step in bringing a new product to life is determining the needs of the new product. Following the process detailed in Figure 34, the manufacturer can determine the customer needs, the resources or capabilities required to meet them, and product specific needs. The first step in the process is to analyze the industry and the internal dynamics of the firm. Together, they can help determine the producers place in the industry. From this analysis, the firm can choose a business strategy, down-select needs specific to the new product, and determine the product-based, customer-oriented strategy.

A.1.1 Automobile Industry External Analysis

The external analysis focuses on the dynamics that occur outside of the manufacturer in question. The two recommended frameworks to facilitate this analysis are the PESTEL and Five Forces. However, depending on the industry and global breadth of operations other frameworks can be introduced. However, for this case with an American car manufacturer in the late 1970s the PESTEL and Five Forces frameworks are sufficient.

A.1.1.1 PESTEL Analysis of Automobile Industry

The PESTEL Framework allows engineers and management to break down the external factors that influence the industry. The factors specific to the late 1970s, automobile industry, are as follows:

- *Political:* Automobile manufacturers have a historically close relationship with their nations' governments. The industry hires a considerably large workforce gaining influence over a city or region's politics (Detroit USA and Stuttgart Germany). Specifically, the automobile industry relies on government subsidized infrastructure investments. In the US, the 1916 Federal Aid Road Act, the 1921

Federal Highway Act, and the 1956 Federal-Aid Highway Act set in motion the adoption of cars in the 19th century, American society.

Politics also play a role in setting tariffs and trade agreements, often protecting their car industry in the process. However, governments' roles in economics have caused trouble for the automobile in the past. In the 1970s, oil prices skyrocketed as first OPEC implemented an embargo and Iran disintegrated into revolution. The automobile industry's sensitivity to the global commodities market requires governments to play a role in minimizing volatility, which comes to the automobile industry's benefit.

With an upcoming election in 1980, both of the prospective presidential candidates shared the same economic policy: reduced regulations cut in government spending, increase in interest rates, and reduced taxes. Of the four policies, three supported business growth and investment. The only one that would hurt the automobile industry would be the cut in government spending. However, the infrastructure investment required for the automobile industry had already run its course in the 1950s and 1960s. Therefore, the only threat was minimal.

- *Economic:* In the late 1970s, the American economy was in bad shape. High unemployment and high inflation stunted growth and investment in many of the country's markets. As a result, the American populous was spending money as soon as they obtained it, but margins were incredibly low, and volumes of goods exchanged were decreasing. The misery index reached its all-time high in the US, and the world economy did not fare much better. Thus, most customers had a tight budget but were willing to spend money.
- *Sociocultural:* The public's view of the automobile industry throughout the 19th century was vastly positive. The car was a foundation of American culture and stood as a symbol of individual freedom throughout the world. Collectors and

hobbyists sprung up around the industry. Furthermore, there were no signs this would change in the late 1970s.

- *Technological:* Due to the oil shocks in the 1970s, automobile manufacturers started investing in aerodynamic and fuel consumption technologies. In the past, cars were boxy to make the manufacturing process more modular, reducing the costs. Now, the emergence new aerodynamic frames challenged these practices and even drove up the price of production.
- *Ecological:* In the late 1960s, due to the abundant use of automobiles in urban centers some states, for example, California in 1966, started enforcing emission regulations. These regulations hoped to reduce the amount of smog and pollution present throughout the United States' cities. Eventually, the entire country adopted these principles in 1968. These regulations would continue to increase throughout the 1970s as the use of leaded gas was forbidden in 1975.
- *Legal:* The abundance of cars created an increase in risk for passenger safety. Though the car manufacturer cannot control how the customer drives the car; they do have some responsibility for how safe the vehicle is. Therefore, in the 1960 and early 1970s, governments passed regulations that demanded manufacturers outfit cars with seat belts and padded dashboards. However, there were still risks of parts failing during operations which could result in injury or death. These incidents incur legal risk for the manufacturer. The manufacturer could mitigate the risk by increasing the quality of parts.

The PESTEL analysis describes an industry that is extremely sensitive and dependent on political policies. It is also dependent on consumer confidence and commodity prices. The state of the industry in the late 1970s is not great, but it looks as though things will change. Due to the industry's dependence on government policy, manufacturers can view the possible change of government policy in 1980 as positive.

A.1.1.2 Five Forces of Automobile Industry

Following the analysis of the late 1970s, automobile industry's external considerations, it is essential to analyze the profitability of the industry. The Five Forces model analyzes the manufacturer's relative power within the industry. The manufacturer's power relates to its ability to negotiate favorable deals and increase profit margins. Furthermore, it can help determine possible business strategies the firm can implement in the industry. The Five Forces for the late 1970s, automobile industry, are as follows:

- *Bargaining Power of Buyers: (Moderate)* The customers in the automobile industry have sufficient power in the industry. Due to a large number of options present to the customers, the switching cost of changing cars is relatively low. However, factors such as brand loyalty and the need for a car reduce some of the customers' power.
- *Bargaining Power of Suppliers: (Low)* The suppliers mostly provide either raw materials or subcomponents with little value as a lone entity. Furthermore, many of these suppliers are tied to a specific manufacturer. This dependence strips the suppliers ability to switch manufacturers since such a large percentage of their business depends on the deal.
- *The Threat of New Entrants: (Low)* After Japanese companies entered the industry in the early 1970s, the market is pretty saturated. It also requires an extreme amount of capital to invest in the assets required of the industry. Without government subsidies the Japanese car companies would have never emerged.
- *The Threat of Substitute Products or Services: (Low-Moderate)* The main substitutes to cars are trains and air travel. In Europe and America, most of

the public uses trains and air travel respectively to travel between urban centers. However, most car owners use the vehicles around the place they live. So, while the threat is there, it is not as menacing.

- *Rivalry among Existing Competitors: (Very-High)* The entry of the Japanese car companies immediately started a price war, causing margins and profits to decrease steadily. The Japanese manufacturers implement new “lean” techniques which guaranteed lower costs and higher reliability. With slipping margins, the American and European companies are hastily copying these techniques.

The Five Forces Analysis depicts an industry where the competition and power of the customers are pushing prices and margins lower. Therefore, a manufacturer has two options: pursue price-leadership or focus on the brand-loyal or luxury-demanding customers and differentiate from the rest of the competition. The first option requires significant investment to catch up to the Japanese efficiency. The second option is a change for many of the American manufacturers’ past strategies. In the past, American car manufacturers presented the idea of “a car in every garage (Herbert Hoover).” With the help of government policy, manufacturers achieved this goal from the 1930s to the 1970s. The next step in the process is to analyze the internal dynamics of the manufacturer to determine which strategy is better.

A.1.2 Automobile Industry Internal Analysis

The internal analysis focuses on the resources, capabilities, and structure inherent of the firm in question. The VRIO framework and value chain analysis allow the firm to develop strategies around the strengths of the company. Furthermore, it identifies weaknesses that the business must bolster. There are other frameworks available, but the two methods are sufficient in determining high-level characteristics relevant to product development of the firm.

A.1.2.1 VRIO Analysis of Automobile Manufacturer

The car manufacturer in this case study is a standard American automobile producer in the late 1970s. At this point, Japanese efficiency and low prices are disrupting the industry. The disruption requires the manufacturer to reevaluate its position. The resources and capabilities available to the firm at this point are the experienced and loyal employee base, the ability to develop and produce a new model within a year, a loyal customer base, an established brand name, and political connections.

The VRIO analysis looks at a company's resources and capabilities to determine which provides the firm with a distinct advantage compared to their competitors. The capabilities and resources that are valuable, rare, hard to imitate, and the company is organized to capture their value should be leveraged in the new strategy. Table 65 displays the results from the VRIO analysis.

Table 65: Late 1970s Automobile Industry VRIO Analysis

Resources	Valuable	Rare	Un-Imitable	Organized
Employees	✓			✓
Loyal Customer Base	✓	✓	✓	✓
Brand Name	✓	✓	✓	✓
Political Connections	✓			✓

From the VRIO analysis, the two most valuable resources available to the manufacturer are its loyal customers and its brand name. These resources can provide a distinct advantage over its competitors. Therefore, the manufacturer should leverage them in a strategy to combat the threats posed to the industry.

A.1.2.2 Value Chain Analysis of Automobile Manufacturer

After, determining critical resources and capabilities available to the firm, value chain analysis looks at the internal structure of the organization. The firm can handle the supporting activities which include organizations infrastructure, human resources, and resource procurement. For primary activities, the firm can manage the inbound

and outbound logistics, marketing and sales, and service. However, there are some considerations concerning technology development and operations or production. The disciplines required for automobile production and the difficulty for the firm to develop subsystems and technologies in each domain are:

- *Aerodynamics: (Easy-Moderate)* Since the vehicles operate at lower speeds only stream-line analysis is required to analyze its effect on the design. The design's sensitivity to these considerations is minimal.
- *Mechanical Drive-Train: (Moderate)* The engine and mechanical drive train is critical to automobile production and performance. It is never an easy task to integrate an engine with the rest of the design, but especially difficult when developing a new engine. Many of the manufacturers set standards and practices to reduce the difficulty of the problem.
- *Structures: (Easy-Moderate)* Weight is a crucial metric that often determines the performance of the vehicle and the structures drive the weight. However, compared to the aerospace industry the safety factors are much higher allowing fundamental analysis in the structure sizing.
- *Production: (Moderate-Hard)* Production of automobiles requires extensive scheduling and coordination alongside the investment in bulky tooling. A large number of processes and components involved in the production creates issues for the manufacturer even if the firm has been in the industry for a while.

Production is the most challenging part of development and productions of a new automobile. It requires a lot of management and capital to produce all of the components required. Some of these parts are small and basic but still require specific processes or tools. As a result, the industry has utilized many external entities that produce parts based on the request. Thus, the firm should consider other approaches rather than vertically integrating these entities in the value chain.

The first option is to taper activities in the value chain. Tapering involves orchestrating external firms production of goods required by the product. The second is to outsource activities by purchasing goods required by the product. Tapering implies cooperation between the firm and the external entities, while outsourcing primarily acquires products previously developed by the external entities.

The factor that drives this decision is suppliers' power. In this case, the suppliers do not have much bargaining power. Therefore, the firm should taper activities creating standards and modules that allow external firms to produce lesser parts, and therefore reducing the difficulty and cost of production.

A.1.3 Selecting Automobile Industry Business Strategy

After analyzing the industry and the internal capabilities of the firm in question, the firm must formulate a business strategy. The firm in question is entering a highly competitive market with multiple market segments: economy, mid-sized, and luxury.

The Japanese brands dominated the low-cost segment. Their superior efficiency and reliability make it hard to compete in this segment without significantly cutting margins. The mid-sized market consists of customers with established jobs and families who need a vehicle to take the family places. In this segment, there are numerous competitors, but the Japanese have not been able to enter since their focus is on economy cars. Finally, the luxury or high-performance segment consists of wealthy customers who look for style, brand recognition, or performance. Only a few of the US and European manufacturers compete in this market since it requires specific intangible factors. Therefore, Figure 73 shows the proposed target segments for this case study.

The company should focus on the mid-sized and luxury segments. The company should try to differentiate itself from the economy segment to drive up margins and prestige. The strategy creates a strategic differentiation position and competitive

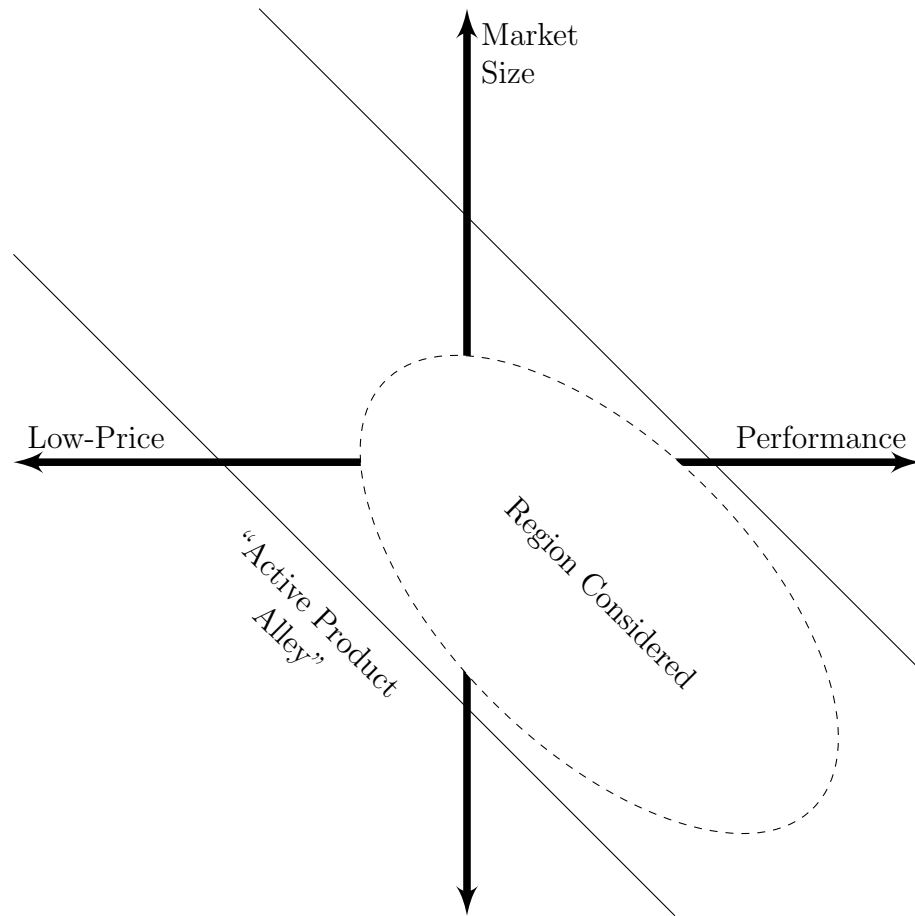


Figure 96: The Capability and Market Size Relational Space for the Automobile Industry

scope pairing (Figure 96). The inclusion of the mid-sized segment is to sustain the volume required for production, amortize the significant fixed costs. The firm can achieve this strategy by producing products for both segments on the same production line. Concurrent productions will take advantage of combining volumes from both segments and enforcing the higher quality standards to both. Furthermore, the ability to increase margins will allow the firm to invest in learning the practices used by the Japanese firms. Once on the same playing field, the American firm will be in a better position to combat the Japanese firms even if it takes more than a few years.

A.1.4 Extracting Customer Needs for New Automobile

Following the formation of the firm's business strategy, the firm must establish the needs for the product. The external analysis provided some insights on the safety and economic requirements of the new product. The value chain analysis identified the need to incorporate modular techniques with regards to the subcomponents and parts, since the firm should plan to taper their production. The choice of business strategy requires the firm to produce the higher performing products on the same production line. Concurrent production implies common components or processes which will achieve the volume required to meet the desired price points.

Now, the firm must derive the customer-specific needs. Figure 39 displays the options available for the firm. Concerning the formulated business strategy, the mid-sized and luxury segments can utilize the same approach. The customers in both segments have moderate power. Therefore, the company can use market analysis and customer surveys to determine all of the needs of the vehicle. The demand high performance suggests considering online reconfigurable components to maximize performance for the luxury segment but is not a priority, since it will add additional weight without gaining much value for the customer. The final needs of the product line are as follows:

1. **Modular and Outsourced Components:** The product line should incorporate modular techniques with regards to the subcomponents since the firm should plan to taper their production.
2. **Production Line:** The need for concurrent production implies common components or processes which will achieve the volume required for meeting the desired price points.
3. **Mid-Sized Market Segment:**
 - (a) *Performance:* The vehicle should be able to carry a family of five with minimal fuel consumption, while still providing the comfort desired by the customer.
 - (b) *Price:* The price point can be a higher than average in this segment since a differentiation should draw a higher price than the rest of the field.
4. **Luxury Market Segment:**
 - (a) *Performance:* The vehicle should be able to achieve high speeds and superior performance while still providing the comforts of luxury and prestige.
 - (b) *Price:* The price point can be a higher than average in this segment since a differentiation should draw a higher price than the rest of the field.

The needs identified using this approach form general descriptions of the tasks, missions, and capabilities required by the vehicle.

A.1.5 Final Automobile-Based, Customer-Oriented Business Strategy

The company should focus on satisfying the requirement of delivering a payload over a given distance. The manufacturer can achieve this capability over various distances and speeds requiring an automotive product line to satisfy the desired tasks. The firm should pursue the market targeting two market segments: mid-sized and luxury.

The pursuit of two different market segments suggests the production of a few vehicle variants on the same production line. Concurrent production and modular standards should save on cost without diminishing performance extensively. The uncertainty of the product architecture drives the need to explore the space and analyze the trade-offs between various vehicles. The next step for the firm would be to create concrete definitions of the tasks, missions, and capabilities required of the systems.

A.2 Defining the Automobile Design Problem

After establishing the needs for the new product line and the formulation of a customer-oriented, product-based business strategy, designers must go through the process of transforming the abstract needs to detailed functional requirements. Many of the facilitators found in Section ?? can help alongside the requirements analysis process found in Section 2.1.2 to create these requirements. For automobiles, the analysis forms range and design speeds which outline the capabilities of the vehicle. Furthermore, their technical requirements can provide benchmarks of expected costs, speeds, reliability, technology level, or other metrics. In this case study, the primary concerns are performance and cost.

The first step in this process is defining the technical parameters that engineers can directly trace from the product's needs. In this case, the cost, gas mileage, payload capacity, and range associated with each vehicle are:

Table 66: Technical Requirements of Automobile Case Study

Vehicle	Acq. Price (1970US)	Gas Mileage (MPG)	Payload Capacity (lbs)	Accel. to 60mph (s)	Range (mi)
Mid-Sized	\$8,000	25	500	9	300
Luxury	\$14,400	25	300	7	250

The next step is to form drive cycles that define the ranges, grades, and speeds

which an automobile must be able to complete. The design missions provide inputs to the simplified models (Section 2.1.4.3) which size the vehicle and estimate performance.

A.2.1 Decomposition of Tasks Required of Automobile

An Automobile must be able to perform a standard drive cycle, which consists of varying and number of speeds and grades. Since both vehicles are for civilian use, they go through the same tests. Section 4.2.2.2 outlines these drive cycles and Figure 70 shows the various operating speeds for these drive cycles.

The other element of the automobiles' function is the ability to take cargo from place to place. Therefore, the development of these vehicles must include additional payload considerations.

A.2.2 Relating Design Missions to Automobile Configuration

Since the drive cycles are the same, the only functional requirement that will differ between the two vehicles' is the payload or cargo capacity. Thus, all civilian cars tend to have the same configuration, which includes a chassis, engine, fuel tank, four wheels, steering system, and transmission. The steering component and the transmission control the vehicle's two degrees of freedom while the rest of the configuration is there to carry cargo across a distance. Looking throughout the industry's history, manufacturers have implemented a few other configurations. Two examples are a convertible roof and three-wheeled vehicles. The first configuration is primarily implemented to meet customers' desires for open-air luxury vehicles. Though, the additional bonus in luxury, the configuration is not applicable in this case since it is hard to measure the luxury bonus the convertible top provides. The second configuration was a dud in sales. Customers primarily viewed these cars as cheap or poorly made, since many of them were not stable and easily tipped over. Thus, these vehicles only found a home in third-world countries. Thus, this case only considers the configuration with

a chassis, engine, fuel tank, four wheels, steering system, and transmission.

A.2.3 Conclusions from Defining the Automobile Design Problem

The straightforward standards set in the automobile industry leave little room for creativity while approaching the design. Due to the dependence of the vehicle's performance and controls on the steering and engine, there is limited room for more online reconfigurability. Furthermore, the standards and customer expectations in the 1970s, there is little need to incorporate sophisticated or revolutionary or technologies to the design. Therefore, this study will primarily be determining what levels of commonality and offline reconfigurability the engineers should incorporate in the product line.

A.3 Establishing a “Valuable” Automobile Product Architecture

The next step requires the determination of weighting between metrics. For this case, desirability, flexibility, and complexity are the three-primary metrics. First, desirability depends on the customers' desires and needs. Also, flexibility and complexity require the gradient and Hessian of a pseudo-objective function which is a combination of the desirability and penalty functions associated with various constraints of the design. Therefore, the derivation of desirability must come first.

A Quality Function Deployment (QFD) produces an objective function by combining the importance of customer needs, functional requirements and the relations between them [122]. The mid-sized and luxury automobile design missions are structured the same way. Though, the magnitudes of the needs and functional requirements differ, the relations stay relatively constant. Therefore, the objective functions for both vehicles can be derived from the same QFD. Table 67 depicts the QFD analysis of the mid-sized and luxury vehicles.

In the QFD for the automobile, the primary customer demands are fuel efficiency,

Table 67: Mid-Sized and Luxury Automobile QFD

	Import.	Cargo Weight	Total Weight	Range	Accel. to 60mph	MPG.	MSRP
Fuel Efficiency	5	0.6	1		0.1	1	0.8
Range	2		0.5	1		0.6	0.3
Cargo	1	1	1	1	0.3	0.6	0.1
Speed	3	0.1	0.2		1		0.3
Cost	4		0.4	0.1	0.1		1
Mid-Sized Target		500 lbs.	4000 lbs.	300 mi.	10 sec	25	\$8,000 (US 1970)
Luxury Target		300 lbs.	3500 lbs.	250 mi.	7 sec	25	\$14,400 (US 1970)
Absolute Importance		4.3	9.2	3.9	4.2	6.8	9.6
Relative Importance		0.11	0.24	0.1	0.11	0.18	0.25

range, cargo weight, speed, and cost, and the functional requirements are the cargo weight, total gross weight, range, time to acceleration to 60mph, fuel efficiency, and the acquisition cost. After providing weightings to each of the customers needs and providing sensitivities among the customer needs and function requirements absolute and relative importance of each functional requirement can be calculated. Combined with targets the objective functions for the two vehicles can be created.

Mid-Sized Vehicle:

$$\phi = 0.11 \frac{W_C}{500} + 0.24 \frac{4000}{W_T} + 0.10 \frac{R}{300} + 0.11 \frac{Acc}{10} + 0.18 \frac{MPG}{25} + 0.25 \frac{8,000}{MSRP} \quad (176)$$

Luxury Vehicle:

$$\phi = 0.11 \frac{W_C}{300} + 0.24 \frac{3500}{W_T} + 0.10 \frac{R}{250} + 0.11 \frac{Acc}{7} + 0.18 \frac{MPG}{25} + 0.25 \frac{14,400}{MSRP} \quad (177)$$

The next step in the process is determining the weightings for the three-primary product architecture evaluation metrics. As shown in Section 3.3, the designer should ask the following six questions to provide general weightings. These weightings act more as guides to depict the general area where to search for product architectures.

- Desirability

1. How much power do the customers have? (**High Power - 3**) As stated in Section A.1.1.2, the customers have a sufficiently high amount of power since they can switch brands easily. Furthermore, since the strategy focuses on the upper-end of the industry, these customers expect their performance and luxury requirements met.
2. How many requirement thresholds must the product achieve? (**Low Number - 1**) Standards for satisfying requirements and regulations have been set allowing new products to flow smoothly through the firm's development process. As long as these standards do not change the difficulty to achieve the number of requirements, the priority on minimizing complexity is minimal.

- Flexibility

1. How long is a product's traditional life span in the industry? (**Very Short - 1**) A typical lifespan for a new product is only a few years. Afterwards, the product mostly becomes obsolete as new models come off the production line.
2. What is the cost to develop and produce a new product? (**High Cost - 3**) There is sufficient cost every time an automobile manufacturer develops a new product, mainly when producing high performance or luxury vehicles.

- Complexity

1. What is the manufacturer's novelty producing a product? (**Low Novelty-1**) Since this case focuses on an established American manufacturer, the novelty of the engineering team is low.
2. How many domains are associating with developing a new product? (**Moderate Number - 2**) There are only a few disciplines that drive automobile design and development. They primarily are drive-train development and production.

The results of this analysis provide the following weightings for the three metrics: Desirability - 0.36, Flexibility - 0.36, and Complexity - 0.28. These are general directions and should act as guides. The product architecture space acts entirely different when compared to typical design problems. Thus, it can be concluded in this case the highest priority should be towards desirability followed by flexibility and complexity.

A.4 Generating Alternative Automobile Product Architectures

Producing alternative product architectures is not as easy as selecting index values of commonality, online, and offline reconfigurability. Instead, this process requires the possible production combinations. Each design requires a configuration, and each configuration requires a chassis, engine, fuel tank, wheels, transmission, and mechanical steering system. The process enforces commonality by giving each component a number varying from one to two. If the two designs' chassis possess the same frame number, an equality constraint makes all the dimensions and characteristics of the two chassis the same. Online reconfigurable components are the mechanical steering system and transmission. During each drive cycle, the controls for the transmission are optimized to create the best performance for the vehicle throughout the cycle. Offline reconfigurability is assumed to be two different subcomponents (engines) sharing the same interface with a common platform (chassis).

The design variables are varied as well to provide distributions of the evaluation metrics relating to the different product architectures. The ranges of the design variables originate from past automotive designs. The ranges attempt to capture the overall design space. Table 68 displays the design variables considered and their minimum and maximum values.

Table 68: Automobile Design Variable Ranges

Design Variable	Minimum	Maximum
Drag Coef.	0.261	0.319
Wheel Base (ft.)	8.25	10
Fuel Storage Power (hp)	1.34	6.71
Fuel Storage Energy (ft-lb)	412	504
Engine Power (hp)	178	218
SFC (lb/hp-hr)	0.3	0.36
Wheel Radius (ft)	0.96	1.07
Rolling Coef.	0.00765	0.00935
Cargo Weight (lbs)	270	330

Combinations of various inputs create product architectures throughout the design space. The distributions resulting from varying the design variables will be used to determine the relationships amongst a product architecture's characteristics, the drivers and the metrics of interest. The next section will detail the tools used to size the vehicles and calculate the metrics of interest.

A.5 Evaluating Alternative Automobile Product Architectures

The evaluation of product architectures requires evaluations of many designs and configurations with the same architecture. The product architecture does not define the system's performance or cost but does influence it. The product architecture also has a relationship with the requirements. Switching the product architecture influences how constrained the space is relating to the flexibility and complexity of the system.

Therefore, to capture the effects of the product architecture on the performance and cost of the product line, the Future Automotive Systems Technology Simulator (FASTSim) was used. The baselines chosen for this case study were the 2011 Toyota Avalon for the mid-sized automobile and the 2011 BMW 335d for the luxury automobile. Engine efficiencies were dropped by 2-5% to make the fuel consumption and the power train efficiencies more realistic to 1970s values. Otherwise, most of the variables were varies around the baseline initial values.

The drive cycles used to size the vehicle are the same as those listed in Section A.2.1. Though FASTSim provides multiple databases of other drive cycles, these are standard EPA tests and are relevant to 1970s design [26].

A.5.1 Validation of Framework’s Consistency in Evaluating Metrics Summary

Validation of the framework requires two steps determining whether grouping the requirements have a similar impact on the product architecture and whether the product architecture has a similar impact on the evaluation metrics.

Figure 97 displays the functional requirements zero-to-sixty times, payload-range, gas mileage, and the number produced in a box plot graph where the points are the medians and the lines are the first and fourth quartiles, excluding outliers. Furthermore, it considers the average and standard deviation of both vehicles.

Compared to Figure 93, the zero-to-sixty requirement for an automobile acts similarly to the takeoff distance, speed, and altitude requirements for a UAV. In the automobile case, the gas mileage behaves opposite as the payload range because the gas mileage is a form of efficiency requiring less energy to complete a mission. Therefore, the payload-range and gas mileage in the automobile case acts similarly to the payload range in the UAV case. Finally, the number produced in the UAV case should act opposite as cost in the automobile case. However, this is not necessarily true because the number produced does not capture the fixed costs associated with

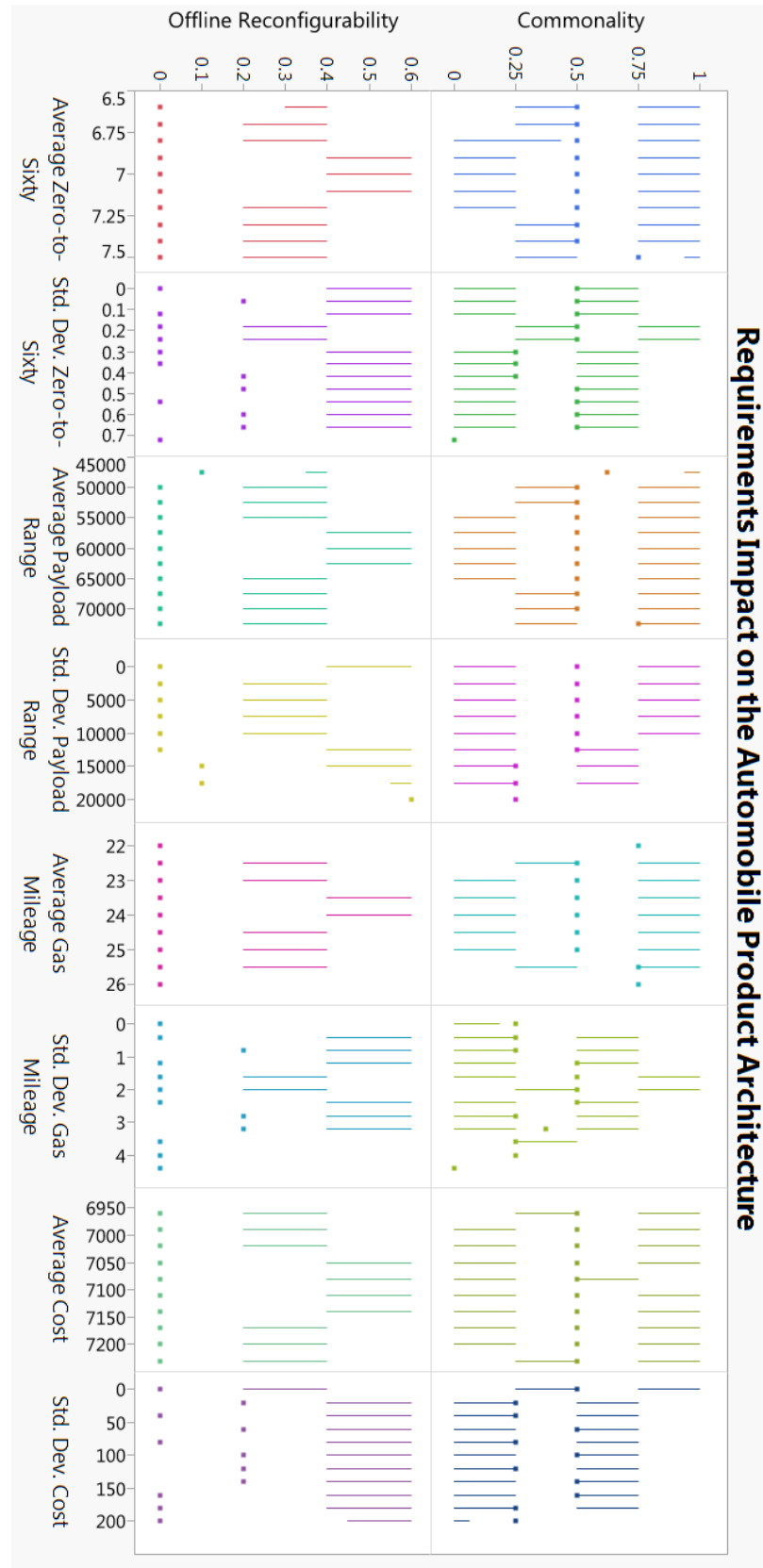


Figure 97: Validation of FA²UST Framework Product Architecture Drivers

the design. However, the general trends hold true.

The fact that both cases act similarly suggests that combining requirements into power, energy, and cost considerations is an appropriate way to approach the problem. Though the trends are not the same, exacerbated by the lack of online-reconfigurable options in the automobile case, the overall trends seem to hold. Though the product architecture composition cannot be directly derived from the requirements, at least the designer can gain some insights on the behavior of the design space.

The results from the automobile case were compared against the results from the UAV case in Figure 98 to validate the evaluation metrics. Figure 5 shows the evaluation metrics results in a box plot where the point is the expected value and the lines are the seventy-fifth percentile to upper, expected range of variance or twenty-fifth percentile to lower, the expected range of variance. Though the fields differ, there are definite trends among the design spaces.

Figure 98 compares the automobile case with the design-feasible cases from the UAV case. For both the automobile and UAV case, increasing commonality shows an initial increase in desirability, but at a point, the desirability begins to decrease. This trend is likely due to commonality's ability to reduce cost, but at a certain point, the performance begins to deteriorate. There seems to be no trend between desirability and offline reconfigurability, as costs and benefits appear to balance each other out.

For both cases, commonality tends to decrease the flexibility of the product architecture as seen in the medians and spreads of the flexibility in Figure 98. The trend is likely due to the additional constraints placed on each common component as commonality increases. The addition of constraints makes the component much more sensitive to changing requirements. Offline reconfigurability tends to increase the flexibility of the product. As the plurality of components reduces each component's sensitivity to changing requirements. Though one might say Figure 98 disputes this, the high flexibility when offline commonality is zero in the automobile case is

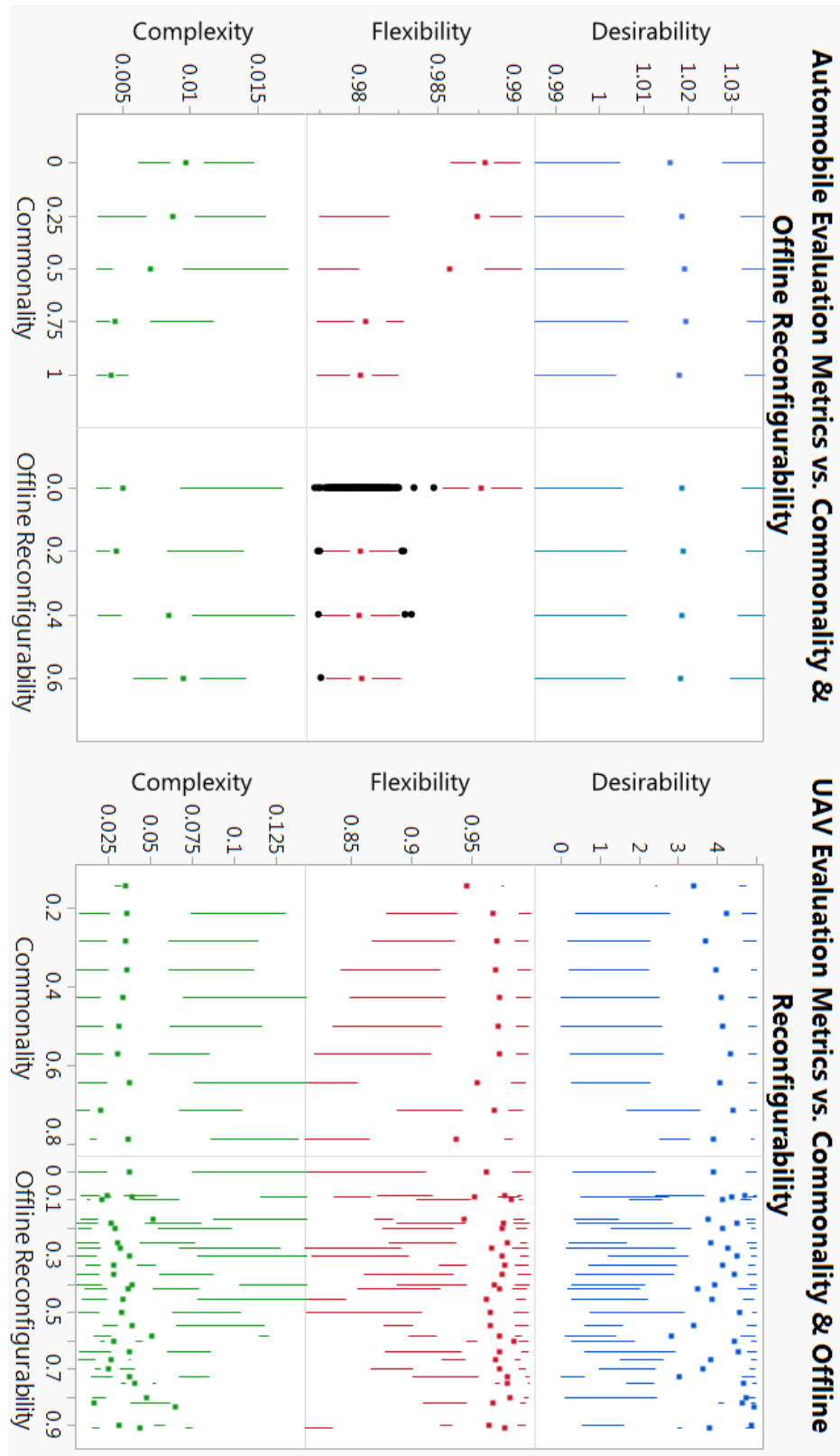


Figure 98: Validation of FA²UST Framework Evaluation Metrics

due to fixed characteristics. The fixed components create a second higher mode which threw of the distribution of offline reconfigurability at zero. A fixed architecture can occur at commonality at zero or one, both times, making offline reconfigurability zero. The automobile case had a higher probability of creating fixed architectures than the UAV case due to the limited number of components. The rest of the space shows a slight upward trend.

For both cases, commonality decreases complexity due to complexity in numbers. The fewer components, the less work in design is required. Since the design of a UAV is more difficult than an automobile, there is an increased risk of higher complexity at higher commonality. At a certain point, high commonality puts many constraints on the shared components, making the component highly sensitive to design changes. The automobile design problem is less complicated, and the study only considers two designs, reducing the risk of higher complexity at high commonality.

For both cases, increasing offline reconfigurability increases complexity again for the opposite reason as increasing commonality. Offline reconfigurability increases the number of interface constraints, creating a greater need for more design work.

Online reconfigurability could not be compared due to the limited nature in the 1970s-automobile design space. A 1970s automobile does not have much room for online reconfigurability since hybrid, and reconfigurable air spoilers had not been created yet. However, the results seem to suggest the overall consistency of the framework's ability to evaluate alternative product architectures.

A.6 Final Decision of Automobile Product Architecture to Implement and Comparison to Historical Case

The weights provided in Section A.3 can now be used to create an overall objective score. Based on the customer desires and functional requirements stated earlier in the process, the designer can filter the product architecture space as shown in Figure 99.

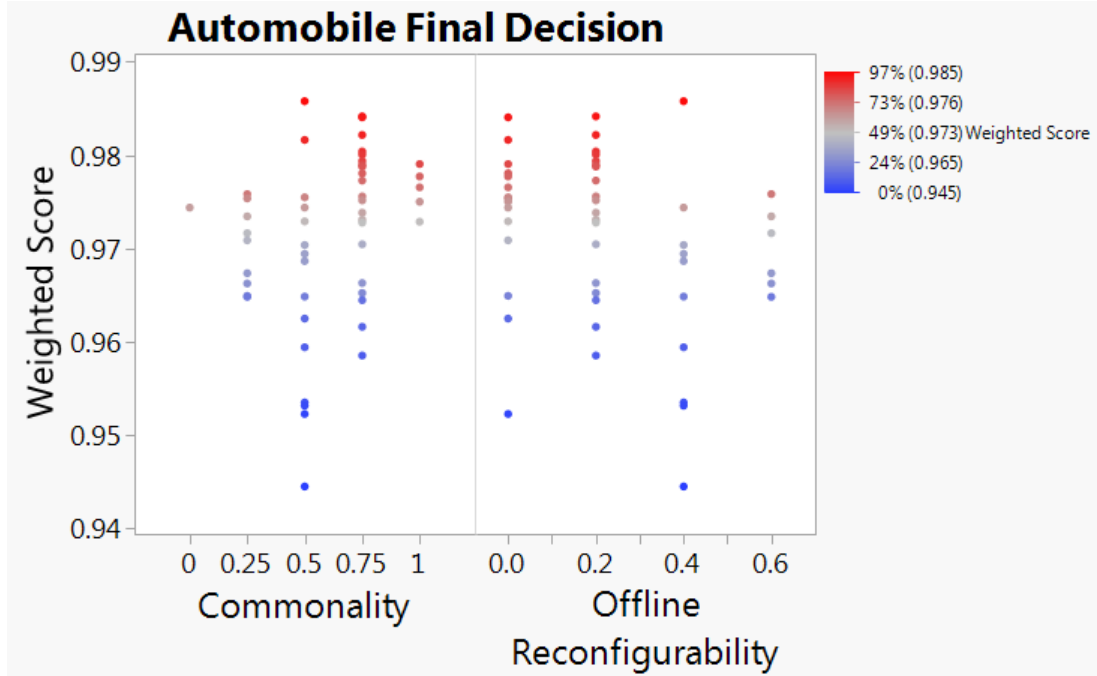


Figure 99: Validation of Automobile Historical Case

The results show the designer should choose to make the vehicles 60% common and 30% offline reconfigurable, but it is important to understand how the results compare against the historical case.

In the late 1970s, Japanese auto-makers disrupted the American automobile industry. They produced vehicles at a much lower cost and higher reliability than their competitors. Therefore, they took over the economy market. The American manufacturers had two options: find new ways to reduce costs or try to move into the higher performing market. The companies that chose to reduce cost did so, by continuing to increase the amount of commonality and offline reconfigurability of their products. However, this severely deteriorated the vehicle's performance [144, 27, 78]. The cannibalization of the products created problems for these manufacturers. For example, Chrysler in the 1980s became obsessed with commonality by continually implementing the K-car platform in their cars causing a lack of distinct new products [132]. The companies that moved into the higher performance segment did not make

the products more common or offline reconfigurable. Instead, they focused on increasing the reliability and operations surrounding the development and production of the vehicle, while increasing the product's performance. This case shows the analysis did not favor higher commonality and offline reconfigurability options. Instead, nominal commonality and offline reconfigurability achieved the higher scores.

REFERENCES

- [1] “Design parameter.” BusinessDictionary.com. WebFinance, Inc. Electronic. <http://www.businessdictionary.com/definition/design-parameters.html>, 2015.
- [2] “Faa: Unmanned aircraft systems civil operations (non-governmental).” Electronic. https://www.faa.gov/uas/civil_operations/, 17 March 2015.
- [3] “Stockholm international peace research institute: Sipri military expenditure database.” Electronic. http://www.sipri.org/research/armaments/milex/milex_database, 2015.
- [4] “U.s. department of commerce: Bureau of economic analysis - national economic accounts.” Electronic. <http://www.bea.gov/national/>, 28 August 2015.
- [5] “Exponential growth of system complexity.” AVSI. The Texas A&M University. College Station, TX. Electronic. <http://savi.avsi.aero/about-savi/savi-motivation/exponential-system-complexity/>, 2016.
- [6] *Cambridge Academic Content Dictionary*, ch. Definition of Product. Cambridge University Press, 2017.
- [7] “Computer programmer salary.” Electronic. <https://money.usnews.com/careers/best-jobs/computer-programmer/salary>., 2017.
- [8] *Merriam-Webster Dictionary*, ch. Definition of Component. Merriam-Webster, 2017.
- [9] *Merriam-Webster Dictionary*, ch. Obsolescence. Merriam-Webster Incorporated, 2017.
- [10] *Oxford Dictionary*, ch. Customization. Oxford University Press, 2017.
- [11] ADS-NEWS, “Cost overruns put global hawk at risk.” Electronic. <https://www.adsgroup.org.uk/articles/10233?i=10233&t=0>, 20 April 2006.
- [12] AGLE, D. C., “The gutless cutlass,” tech. rep., Smithsonian Air & Space Magazine, August 2012.
- [13] AIRFORCE-TECHNOLOGY.COM, “F-14 tomcat, united states of america.” Electronic. <http://www.airforce-technology.com/projects/f14/>.
- [14] AIRFORCE-TECHNOLOGY.COM, “Predator rq-1 / mq-1 / mq-9 reaper uav, united states of america.” Electronic. <http://www.airforce-technology.com/projects/predator-uav/>.

- [15] ALBERS, A., SEDCHAICHARN, K., SAUTER, C., and BURGER, W., “A method to define a product architecture early in product development using a contact and channel model,” in *International Conference on Engineering Design: Design Methods and Tools* (NORELL BERGENDAHL, M., GRIMHEDEN, M., LEIFER, L., SKOGSTAD, P., and LINDEMANN, U., eds.), vol. 5, (Palo Alto, CA, USA), pp. 241–252, Institute of Product Development, University of Karlsruhe, 27 August 2009.
- [16] ALLISON, J. T., “Complex system optimization: A review of analytical target cascading, collaborative optimization, and other formulations,” Master’s thesis, The University of Michigan, 2004.
- [17] AMERICANAUTOMOBILES.COM, “The winton automobile & the winton motor carriage co..” Electronic. <http://www.american-automobiles.com/Winton.html>, 2012.
- [18] ANDERSON, J. D. J., *Aircraft Performance and Design*. Boston, MA: WCB McGraw-Hill, 1999.
- [19] AUSTIN, R., *Unmanned Aircraft System: UAVS Design, Development and Deployment*. The Atrium, Southern Gate, Chichester, West Sussex, UK: John Wiley & Sons, Ltd., 2010.
- [20] BEARDEN, D., “Perspectives on nasa mission cost and schedule performance trends,” tech. rep., Aerospace Corp., 2008.
- [21] BECZ, S., PINTO, A., ZEIDNER, L. E., KHIRE, R., BANASZUK, A., and REEVE, H. M., “Design system for managing complexity in aerospace systems,” in *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, (Fort Worth, TX), AIAA, 13-15 September 2010.
- [22] BERRY, S. and PAKES, A., “The pure characteristics demand model,” *International Economic Review*, vol. 48, pp. 1193–1225, November 2007.
- [23] BLOOMBERG.COM, “Markets: Energy.” Electronic. <https://www.bloomberg.com/energy>, 2017.
- [24] BOEING.COM, “F/a-18 hornet fighter: Historical snapshot.” Electronic. <http://www.boeing.com/history/products/fa-18-hornet.page>.
- [25] BOEING.COM, “Boeing 737 max.” Electronic. <http://www.boeing.com/commercial/737max/>, 2015.
- [26] BROOKER, A., GONDER, J., WANG, L., WOOD, E., LOPP, S., and RAMROTH, L., “Fastsim: A model to estimate vehicle efficiency, cost, and performance,” in *SAE 2015 World Congress & Exhibition*, (Detroit, MI), SAE International, 21-23 April 2015.

- [27] CAMERON, B. G. and CRAWLEY, E. F., “Crafting platform strategy based on anticipated benefits and costs,” *Advances in Product Family and Product Platform Design*, pp. 49–70, 2014.
- [28] CAMERON, D., “F.a.a. drone laws start to clash with stricter local rules,” in *The Wall Street Journal*, October 5 2017.
- [29] CANIS, B., “Unmanned aircraft systems (uas): Commercial outlook for a new industry,” tech. rep., Congressional Research Service, Washington, DC, 9 September 2015.
- [30] CAPACCIO, T., “Pentagon says northrop’s global hawk drone isn’t ‘effective’,” electronic news. <http://www.bloomberg.com/news/articles/2011-06-06/pentagon-says-northrop-s-global-hawk-drone-isn-t-effective->, Bloomberg Business, New York, NY, 6 June 2011.
- [31] CASTELLS, M., *The Rise of the Network Society*, vol. 1. Chichester, West Sussex, United Kingdom: John Wiley and Sons, Ltd., 2 ed., 2010.
- [32] CHEN, W., ALLEN, J. K., and MISTREE, F., “The robust concept exploration method for enhancing concurrent systems design,” *Concurrent Engineering: Research and Applications*, vol. 5, no. 3, pp. 203–217, 1997.
- [33] CLEVELAND, C. J., “The encyclopedia of the earth: De rivaz, francois isaac.” Electronic. <http://www.eoearth.org/view/article/151661/>, 24 August 2008.
- [34] COLLIER, D. A., “The measurement and operating benefits of component part commonality,” *Decision Sciences*, vol. 12, p. 8596, January 1981.
- [35] COLLIS, D. J. and RUKSTAD, M. G., “Can you say what your strategy is?,” *Harvard Business Review*, April 2008.
- [36] COMMISSION, E., “Notice of proposed amendment 2017-05 (a): Introduction of a regulatory framework for the operation of drones,” tech. rep., European Aviation Safety Agency, 2017.
- [37] COMMISSION, E., “Communication from the commission to the european parliament and the council,” Communication 207, European Commission, Brussels, Belgium, 8 April 2014.
- [38] CORPORATE.FORD.COM, “Our history: Company timeline.” Electronic. <https://corporate.ford.com/company/history.html>.
- [39] COURTNEY, H., KIRKLAND, J., and VIGUERIE, P., “Strategy under uncertainty,” *Harvard Business Review*, December 1997.
- [40] CRAWLEY, E., DE WECK, O., EPPINGER, S., MAGEE, C., MOSES, J., SEERING, W., SCHINDALL, J., WALLACE, D., and WHITNEY, D., “The influence of architecture in engineering systems,” *MIT Engineering Systems Division*, 29-31 March 2004.

- [41] DEBORD, M., “The us auto industry may surprise everyone in 2017,” tech. rep., Business Insider, 2017.
- [42] Defense Acquisition University Press, Fort Belvoir, VA 22060, *System Engineering Fundamentals*, January 2001.
- [43] Department of Defense, 2530 Loop Road West, Wright-Patterson AFB, OH 45433, *Department of Defense Standard Practice: Glossary of Definitions, Ground Rules, and Mission Profiles to Define Air Vehicle Performance Capability*, a ed., 14 February 2003.
- [44] Department of Defense, 6000 Defense Pentagon Washington, D.C., *The DoDAF Architecture Framework Version 2.02*, August 2010.
- [45] DEVNANI-CHULANI, S., BRADFORD, C., and BOEHM, B., “Calibrating the cocomo ii post-architecture model,” tech. rep., Computer Science Department, University of Southern California, Los Angeles, CA 90098, 1998.
- [46] DIAZ-CALDERON, A., PAREDIS, C. J. J., and KHOSLA, P. K., “A composable simulation environment for mechatronic systems,” in *1999 SCS European Simulation Symposium*, (Erlangen, Germany), 26-28 October 1999.
- [47] DIAZ-CALDERON, A., PAREDIS, C. J. J., and KHOSLA, P. K., “Organization and selection of reconfigurable models,” in *Winter Simulation Conference 2000*, 10-13 December 2000.
- [48] DODGE, Y., *The Oxford Dictionary of Statistical Terms*, ch. Interaction. Oxford University Press, 2003.
- [49] DOMAN, D. B., “Rapid mission planning for aircraft thermal management,” tech. rep., American Institute of Aeronautics and Astronautics, Kissimmee, Florida, 2015.
- [50] DOMERCANT, J. C., *ARC-VM: an architecture real options complexity-based valuation methodology for military systems-of-systems acquisitions*. PhD thesis, Georgia Institute of Technology, November 14 2011.
- [51] DONLEY, M. B. and SCHWARTZ, N. A., “United states air force: Unmanned aircraft systems flight plan 2009-2047,” tech. rep., United States Air Force, Washington, DC, 18 May 2009.
- [52] DORR, LESS, J. and DUQUETTE, A., “Press release faa asks for public input on uas test site selection,” tech. rep., Federal Aviation Administration, 800 Independence Avenue, SW Washington, DC 20591, 7 March 2012.
- [53] DOWER, G., “Ridek modular electric vehicles.” Electronic. <http://gordondower.com/ridek-modular-electric-vehicles/>, 9 May 2000.

- [54] DWYER, A. and O'BRIMSKI, F., "Aircraft design for carrier operations," tech. rep., NAVAIRSYSCOM: Conceptual Design Division, November 1997.
- [55] EVANS, P. and FORTH, P., "Borges' map: Navigating a world of digital disruption," *BCG Perspectives*, April 2015.
- [56] FAHLSTROM, P. G. and GLEASON, T. J., *Introduction to UAV Systems*. The Atrium, Southern Gate, Chichester, West Sussex, UK: John Wiley & Sons, Ltd., 4th ed., 2012.
- [57] FISCHMAN, L., MCRITCHIE, K., and GALORATH, D. D., "Inside seer-sem," tech. rep., Galorath Inc., El Segundo, CA, April 2005.
- [58] FRANK, C. P., *A Design Space Exploration Methodology to Support Decisions under Evolving Uncertainty in Requirements and its Application to Advanced Vehicles*. PhD thesis, Georgia Institute of Technology, August 2016.
- [59] FREEMAN, D. F., *A Product Family Design Methodology Employing Pattern Recognition*. PhD thesis, Georgia Institute of Technology, December 2013.
- [60] GERMAN, B., "Ae 8803 ger: Optimization for the design of engineered systems: Multidisciplinary design optimization (mdo)." Lecture, 2013.
- [61] GIPSON, L., "Unmanned aircraft systems integration in the national airspace system (uas in the nas) project," tech. rep., National Aeronautics and Space Administration, 13 July 2016.
- [62] GOYAL, K. K., JAIN, P. K., and JAIN, M., "A novel approach to measure machine reconfigurability in reconfigurable manufacturing system," in *Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium* (KATALINIC, B., ed.), vol. 22, (Vienna, Austria, EU), p. 959, DAAAM International, DAAAM International, 2011.
- [63] GUDMUNDSSON, S., *General Aviation Aircraft Design: Applied Methods and Procedures*. 225 Wyman Street, Waltham, MA 02451: Butterworth-Heinemann, first ed., 2014.
- [64] GUMASTA, K., GUPTA, S. K., BENYOUSSEF, L., and TIWARI, M., "Developing a reconfigurability index using multi-attribute utility theory," *International Journal of Production Research*, vol. 49, pp. 1669–1683, 15 March 2011.
- [65] HALL, N., "The beginner's guide to aerodynamics," tech. rep., NASA Glenn Research Center, 5 May 2015.
- [66] HASKINS, C., ed., *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. No. 3, INCOSE, June 2006.
- [67] HICKMAN, K., "World war ii: Grumman f6f hellcat." Electronic. <http://militaryhistory.about.com/od/worldwarii/aircraft/p/f6f-hellcat.htm>, 2015.

- [68] HÖLTTÄ, K., SUH, E. S., and DE WECK, O., “Tradeoff between modularity and performance for engineering systems and products,” in *15th International Conference on Engineering Design*, pp. 449–450, ICED, 2005.
- [69] IEEE, *Systems and Software Engineering - Vocabulary*, 2010.
- [70] ISRAELI-WEAPONS.COM, “Scout.” Electronic. <http://www.israeli-weapons.com/weapons/aircraft/uav/scout/Scout.html>.
- [71] JIAO, J. R., SIMPSON, T. W., and SIDDIQUE, Z., “Product family design and platform-based product development: a state-of-the-art review,” *Journal of Intelligent Manufacturing*, vol. 18, pp. 5–29, February 2007.
- [72] JOINER, S., “What couldn’t the f-4 phantom do?: A tribute to mcdonnell’s masterpiece fighter jet,” tech. rep., Smithsonian Air & Space Magazine, March 2015.
- [73] JOSHI, D., “Commercial unmanned aerial vehicle (uav) market analysis industry trends, companies and what you should know,” tech. rep., Business Insider, New York, NY, 8 Aug. 2017.
- [74] KANG, C., “Drone registration rules are announced by f.a.a.,” in *N.Y. Times*, December 14 2015.
- [75] KANG, C., “F.a.a. drone laws start to clash with stricter local rules,” in *N.Y. Times*, December 27 2015.
- [76] KAPURCH, S. J. and RAINWATER, N. E., *NASA Systems Engineering Handbook*. NASA, Washington, D.C. 20546, revision 1 ed., December 2007.
- [77] KHANA, P. and KHANA, A., “The evolution of technology,” in *Big Think Edge*, Big Think, Inc., 2016.
- [78] KIM, K. and CHHAJED, D., “Commonality in product design: Cost saving, valuation change and cannibalization,” *European Journal of Operational Research*, vol. 125, pp. 602–621, 16 September 2000.
- [79] KINGERY IV, V. U., “The state of automotive: 2017 outlook,” tech. rep., GE, 2017.
- [80] KOTA, S., SETHURAMAN, K., and MILLER, R., “A metric for evaluating design commonality in product families,” *Journal of Mechanical Design*, vol. 122, pp. 403–410, June 1998.
- [81] LEE, S., TITCHKOSKY, L., and BOWEN, S., “Software cost estimation,” tech. rep., Department of Computer Science, University of Calgary, 2002.
- [82] LEVIS, A., *Handbook of Systems Engineering and Management*, ch. System Architectures, pp. 427–454. New York, NY: John Wiley & Sons, 1999.

- [83] LOCKHEEDMARTIN.COM, “The multi-variant, multirole 5th generation fighter.” Electronic. <https://www.f35.com/about>.
- [84] LONGSTREET, D., “Fundamentals of function point analysis,” tech. rep., Longstreet Consulting Inc., Blue Springs, MO, 2005.
- [85] LUKE, E. A., “Defining and measuring scalability,” in *Scalable Parallel Libraries Conference*, (Mississippi State, MS), IEEE, 6-8 October 1993.
- [86] MALER-SPEREDELOZZI, V., KOREN, Y., and HU, S., “Convertibility measures for manufacturing systems,” *CIRP Annals Manufacturing Technology*, vol. 52, no. 1, 2003.
- [87] MALONE, P., APGAR, H., STUKES, S., and STERK, S., “Unmanned aerial vehicles unique cost estimating requirements,” in *2013 IEEE Aerospace Conference*, (Big Sky, MT), IEEE, 2-9 March 2013.
- [88] MARTIN, M. V. and ISHII, K., “Design for variety: Development of complexity indices and design charts,” in *ASME Design Engineering Technical Conferences*.
- [89] MAVRIS, D., “Fixed wing aircraft design i: Classical design methods.” School of Aerospace Engineering, Georgia Institute of Technology. Lecture, August 2011.
- [90] MAVRIS, D., “Unified trade-off environment (ute).” School of Aerospace Engineering, Georgia Institute of Technology. Lecture, 15 January 2013.
- [91] MAVRIS, D. N., GRIENDLING, K., and DICKERSON, C. E., “Relational oriented systems engineering and technology tradeoff analysis (rosetta) framework,” in *6th International Conference on System of Systems Engineering Conference*, (Albuquerque, NM), pp. 49–54, Aerospace Systems Design Laboratory, Georgia Institute of Technology, IEEE, June 27-30 2011.
- [92] MCGIBBON, T., “Modern empirical cost and modern empirical cost and schedule estimation tools,” tech. rep., ITT Industries - Systems Division, 775 Dae-dalian Drive Rome, NY 13441, 20 August 1997.
- [93] MEOLA, A., “Drone market shows positive outlook with strong industry growth and trends,” tech. rep., Business Insider, 2017.
- [94] MEYER, M. and LEHNERD, A., *The Power of Product Platform - Building Value and Cost Leadership*. New York, NY: Free Press, 1997.
- [95] MILITARY.COM, “Hh-60g pave hawk.” Electronic. <http://www.military.com/equipment/hh-60g-pave-hawk>.
- [96] MILITARY.COM, “Uh-60a/l black hawk.” Electronic. <http://www.military.com/equipment/uh-60a-l-black-hawk>.

- [97] MILITARYFACTORY.COM, “Northrop grumman rq-4 global hawk unmanned aerial vehicle (uav) / drone (2001).” Electronic. http://www.militaryfactory.com/aircraft/detail.asp?aircraft_id=40, 29 June 2015.
- [98] MILITARYFACTORY.COM, “Sikorsky hh-60 / mh-60t jay hawk medium range search & rescue / interdiction helicopter (1990).” Electronic. http://www.militaryfactory.com/aircraft/detail.asp?aircraft_id=282, 2 July 2015.
- [99] MILITARYFACTORY.COM, “Vought f-8 crusader carrier-borne naval fighter (1957).” Electronic. http://www.militaryfactory.com/aircraft/detail.asp?aircraft_id=181, 18 January 2015.
- [100] MORRIS, P., “Pierce-arrow motor car company history.” Electronic. <http://www.pierce-arrow.org/history/index.php>.
- [101] MUSEUMOFFLIGHT.ORG, “The fieseler fi 103 (v1) german ‘buzz bomb’.” Electronic. <http://www.museumofflight.org/exhibits/fieseler-fi-103-v1>.
- [102] NAVAIR, “Unmanned carrier launched air-borne surveillance and strike system.” Electronic. <http://www.navair.navy.mil/index.cfm?fuction=home.display&key=A1DA3766-1A6D-4AEA-B462-F91FE43181AF>.
- [103] NAYAK, R. U., CHEN, W., and SIMPSON, T. W., “A variation-based methodology for product family design,” in *ASME 2000 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, (Baltimore, Maryland), ASME, ASME, 13 September 2000.
- [104] ODASA-CE Technomics, *Unmanned Aerial Vehicle System Acquisition Cost Estimating Methodology*, 37th DoD Cost Analysis Symposium, October 2015.
- [105] OF AMERICAN HISTORY, N. M., “Winton automobile.” Electronic. http://amhistory.si.edu/onthemove/collection/object_1277.html, 2017.
- [106] OF AMERICAN HISTORY STAFF, N. M., “Ford model a automobile.” Electronic. http://amhistory.si.edu/onthemove/collection/object_1319.html, 2017.
- [107] OPGI, “Decoding general motors body style designations,” in *Original Parts Group Incorporated*, Original Parts Group Incorporated, 28 May 2013.
- [108] PARKER, S. P., *McGraw-Hill Dictionary of Scientific & Technical Terms*. The McGraw-Hill Companies, Inc., 6th ed., 2003.
- [109] PARKIN, R., WILK, R., HIRSH, E., and SINGH, A., “2017 automotive trends,” tech. rep., PwC, 2017.

- [110] PATE, D. J., PATTERSON, M. D., and GERMAN, B. J., "Optimizing families of reconfigurable aircraft for multiple missions," *Journal of Aircraft*, vol. 49, no. 6, pp. 1988–2000, 2012.
- [111] PATTERSON, M. D., PATE, D. J., and GERMAN, B. J., "Performance flexibility of reconfigurable families of unmanned vehicles," *Journal of Aircraft*, vol. 49, pp. 1831–1843, December 2012.
- [112] PBS-NOVA, "Pre-aviation uavs: Perley's aerial bomber (usa)." Electronic. http://www.pbs.org/wgbh/nova/spiesfly/uavs_01.html.
- [113] PERCY, T. K., "Simplifying complex problems with systems engineering tools: A lunar architecture analysis case study," in *Space Systems Engineering Conference*, vol. 1, Atlanta, GA: Georgia Institute of Technology, 10 November 2005.
- [114] PHADKE, M., *Quality Engineering Using Robust Design*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
- [115] PIKE, J., "F-14 tomcat variants." Electronic. <https://www.globalsecurity.org/military/systems/aircraft/f-14-variants.htm>, April 2016.
- [116] PIRMORADI, Z., WANG, G. G., and SIMPSON, T. W., *Advances in Product Family and Product Platform Design: Methods and Applications*, ch. Chapter 1: A Review of Recent Literature in Product Family Design and Platform-Based Product Development, pp. 1–36. Springer, 2014.
- [117] PORTER, M. E., *Competitive Advantage: Creating and Sustaining Superior Performance*. 1230 Avenue of the Americas, New York, NY: The Free Press - Simon & Schuster Inc., 1985.
- [118] PORTER, M. E., "The five competitive forces that shape strategy," *Harvard Business Review*, January 2008.
- [119] PRABHAKAR, A. and WALKER, S. H., "Breakthrough technologies for national security," tech. rep., DARPA, 675 North Randolph Street Arlington, VA 22203-2114, March 2015.
- [120] PUTNAM, L. H., "A general empirical solution to the macro software sizing and estimating problem," *IEEE Transactions on Software Engineering*, 1978.
- [121] RAYMER, D. P., *Aircraft Design: A Conceptual Approach*. Blacksburg, VA: AIAA, fourth ed., 2006.
- [122] REVELLE, J. B., MORAN, J. W., and COX, C. A., *The QFD Handbook*. 605 Third Ave. New York, New York: John Wiley & Sons, Inc., 1998.

- [123] RICHARDS, D., MCKAY, B. D., and RICHARDS, W. A., "Collective choice and mutual knowledge structures," *Advances in Complex Systems*, vol. 1, pp. 221–236, 1998.
- [124] ROGERS, D. F., "Propeller efficiency: A rule of thumb," tech. rep., NAR Associates, 2010.
- [125] ROTHARMEL, F. T., *Strategic Management: Concepts*. 2 Penn Plaza New York, NY: McGraw-Hill Education, 3rd ed., 2017.
- [126] SALEH, J. H., HASTINGS, D. E., and NEWMAN, D. J., "Flexibility in system design and implications for aerospace systems," *Acta Astronautica*, vol. 53, pp. 927–944, December 2003.
- [127] SALEH, J. H., MARK, G., and JORDAN, N. C., "Flexibility: a multi-disciplinary literature review and a research agenda for designing flexible engineering systems," *Journal of Engineering Design*, vol. 20, pp. 307–323, June 2009.
- [128] SHALAL-ESA, A., "Cost of flying northrop's global hawk down over 50sources," electronic news. <http://www.suasnews.com/2013/09/25052/cost-of-flying-northrops-global-hawk-down-over-50-sources/>, sUAS News, 14 September 2013.
- [129] SIDDIQI, A. and DEWECK, O. L., "Modeling methods and conceptual design principles for reconfigurable systems," *Journal of Mechanical Design*, vol. 130, no. 20, 2008.
- [130] SIDDIQUE, Z., ROSEN, D., and WANG, N., "On the applicability of product variety design concepts to automotive platform commonality," in *ASME Design Engineering Technical Conferences*, (Atlanta, GA), ASME, 13-16 September 1998.
- [131] SIMPSON, J. and WEINER, E., eds., *Oxford English Dictionary*, ch. Logistics. 2001 Evans Road Cary, NC 27513: Oxford University Press, 2nd ed., 2015.
- [132] SIMPSON, T. W., SIDDIQUE, Z., and JIAO, J. R., *Product Platform and Product Family Design: Methods and Applications*, ch. 1, pp. 1–15. New York, NY: Springer, 2005.
- [133] SINHA, R., PAREDIS, C. J., and KHOSLA, P. K., "Interaction modeling in systems design," in *2001 ASME Design Engineering Technical Conferences*, (Pittsburgh, PA), ASME, 9-12 September.
- [134] STAFF, H., "Model t." Electronic. <http://www.history.com/topics/model-t>, 2010.
- [135] STAFF, H., "History of the huey." Electronic. http://www.huey.co.uk/history_huey.php, 2017.

- [136] STAFF, I., “Economies of scale,” 2017.
- [137] STAFF, P. A. M., “Pierce arrow history.” Electronic. <http://www.pierce-arrow.com/history>, 2013.
- [138] SULLIVAN, M. J., “Defense acquisitions: Assessments of selected weapon programs,” report to congressional committees, United States Government Accountability Office, March 2014.
- [139] SUMMERS, J. D. and SHAH, J. J., “Mechanical engineering design complexity metrics: Size, coupling, and solvability,” *Journal of Mechanical Design*, vol. 132, January 14 2010.
- [140] TARANTOLA, A., “The ryan firebee: Grandfather to the modern uav.” Electronic. <http://gizmodo.com/the-ryan-firebee-grandfather-to-the-modern-uav-1155938222>, 27 August 2013.
- [141] THEVENOT, H. J. and SIMPSON, T. W., “Commonality indices for product family design: A detailed comparison,” *Journal of Engineering Design*, vol. 17, pp. 99–119, April 2006.
- [142] THEVENOT, H. J. and SIMPSON, T. W., *Product Platform and Product Family Design: Methods and Applications*, ch. Commonality Indices for Assessing Product Families, pp. 107–129. University Park, PA, 16802: Springer, 1 ed., 2006.
- [143] THEVENOT, H. J. and SIMPSON, T. W., “A comprehensive metric for evaluating component commonality in a product family,” *Journal of Engineering Design*, vol. 18, pp. 577–598, December 2007.
- [144] ULRICH, K., “Fundamentals of product modularity,” *Management of Design*, pp. 219–231, 1994.
- [145] ULRICH, K., “The role of product architecture in the manufacturing firm,” *Research Policy*, vol. 24, no. 3, pp. 419–440, 1995.
- [146] UPTON, E., *An Intelligent, Robust Approach to Volumetric Aircraft Sizing*. PhD thesis, Georgia Institute of Technology, 7 May 2007.
- [147] US Air Force, Air Force Special Operations Command, 229 Cody Ave. Suite 103 Hurlburt Field, FL 32544-5312, *USAF Raven Fact Sheet*, November 2009.
- [148] USNAVY, “Sh-60 sea hawk helicopter.” Electronic. http://www.navy.mil/navydata/fact_display.asp?cid=1200&tid=500&ct=1, 24 August 2012.
- [149] VALERDI, R., “Cost metrics for unmanned aerial vehicles,” tech. rep., AIAA, Cambridge, MA, September 2005.

- [150] WACKER, J. G. and M., T., “Component part standardization: An analysis of commonality sources and indices,” *Journal of Operations Management*, vol. 6, pp. 219–244, February 1986.
- [151] WALDEN, D. D., ROEDLER, G. J., FORSBERG, K. J., HAMELIN, R. D., and SHORTELL, T. M., eds., *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Wiley, 4th ed., July 2015.
- [152] WALLACE, D., “2.009 product engineering processes: Product architecture.” Electronic. http://web.mit.edu/2.009/www/lectures/19_ProductArchitecture.pdf, 20 October 2014.
- [153] WEGERT, E., *Visual Complex Functions: An Introduction with Phase Portraits*. Birkhäuser, 2010.
- [154] WINGFIELD, N., “Regulators propose a drone registration system,” in *N.Y. Times*, October 20 2015.
- [155] YU, J. S., GONZALEZ-ZUGASTI, J. P., and OTTO, K. N., “Product architecture definition based upon customer demands,” *Journal of Mechanical Design*, vol. 121, pp. 329–335, September 1999.
- [156] ZEIDNER, L. E., ROCK, B. E., DESAI, N. A., REEVE, H. M., and STRUASS, M. P., “Application of a technology screening methodology for rotorcraft alternative power systems,” in *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, (Orlando, FL), AIAA, 4 - 7 January 2010.
- [157] ZEIDNER, L. E., REEVE, H. M., KHIRE, R., and BECZ, S., “Architectural enumeration & evaluation for identification of low-complexity systems,” in *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, (Fort Worth, TX), AIAA, 13 - 15 September 2010.