ELICITATION AND FORMAL SPECIFICATION OF RUN TIME ASSURANCE REQUIREMENTS FOR AEROSPACE COLLISION AVOIDANCE SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Kerianne L. Hobbs

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

Georgia Institute of Technology

May 2020

ELICITATION AND FORMAL SPECIFICATION OF RUN TIME ASSURANCE REQUIREMENTS FOR AEROSPACE COLLISION AVOIDANCE SYSTEMS

Advisory Committee:

Dr. Eric M. Feron, Co-Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Glenn Lightsey, Co-Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Moriba K. Jah
School of Aerospace Engineering and Engineering Mechanics
University of Texas at Austin

Dr. Joseph H. Saleh
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Alwyn E. Goodloe
Safety Critical Avionics Systems Branch
National Aeronautics and Space Administration

Date Approved: March 13, 2020

If a machine is expected to be infallible, it cannot also be intelligent.

Alan Turing

To Mom and Dad.

ACKNOWLEDGEMENTS

There is an African proverb that says it takes a village to raise a child. Similarly, I couldn't have completed my PhD dissertation without a large community supporting me.

My dissertation would not have happened without the faith and support of my advisor, Eric Feron. Thank you for taking a chance on me as a non-traditional graduate student balancing completing degree requirements within constraints of a job I love over 500 miles away. Your chaotic genius pushed me further than I thought possible. From the courses I took with you to the years of research discussions, my research vision is forever colored with a richer perspective.

Thank you to my diverse committee members who each challenged and supported me in their own way. Glenn Lightsey helped me understand how I could contribute in the space domain and stepped in to co-advise while Eric was on sabbatical. Moriba Jah inspired me to work on a space controls problem that might contribute to solving space traffic management challenges and connected me with a community who influenced the research direction. Alwyn Goodloe's years of mentoring have pushed me to consider some of the more challenging technical details in the computer science domain, helped point me towards the applicable computer science research, and kept me honest in my formal methods research. Joseph Saleh's attention to detail as well as ability to go from a systems-level perspective down to deep technical details was also exceptionally helpful in early revisions of this dissertation.

Thank you to the many technical experts who shaped my understanding of practical collision avoidance, formal methods and the space domain. Thanks to Amy Burns, Don Swihart, Finley Barfield, Mike Coy, Mark Skoog, and many others for mentoring me throughout my time on the Automatic Collision Avoidance Technologies (ACAT) program. Thank you to Lucas Wagner and Jen Davis for working with me in SpeAR over the years, helping me better understand formal requirements, and coming to the rescue when I was stuck using SpeAR in this research. Thanks to Mike Whalen for introducing me to assume-guarantee contracts and architecture design considerations. Thank you to Stan Bak for coaching me in hybrid systems and reachability and his helpful edits on this dissertation. Thank you to Guillaume Brat, Alwyn Goodloe, Natasha Neogi, Misty Davies, Paul Minor, Cesar Munoz, Kristin Rozier, Ewen Denny, Ganesh Pai, Ben Hocking, Jonathan Rowanhill, Nancy Leveson, and others whose discussions have helped inform my understanding of

formal methods, hazards, and faults in aerospace. Thank you to Matt Dilsaver for introducing me to airworthiness certification and helping me shadow for a couple weeks to better understand that world. Thank you to Darren McKnight, Don Greiman, Jess Sponable, Jodi Goldberg, Charles Finley, Ed Quinonez, Joe Iungerman, Ed Lu, Charles Fischer, Mike Gabor and others who helped me get up to speed on the state of Space Traffic Management. Special thank you to Kendra Lang, Sean Phillips, Michelle Simon, Scott Erwin, Chris Petersen, Ryan Weisman, and Alex Collins for providing feedback on the space portions throughout my research.

Thank you to all my mentors and supervisors along the way who supported me throughout my graduate studies. Thank you to Howard Emsley for helping me navigate the process of starting classes towards my Master's from 2012-2014 and letting me use my lunch hour to attend courses. Thank you to Jeff Tromp for supporting my continued coursework from 2015-2016. Thank you to Matt Clark who introduced me to verification and validation in 2012, showed me that the Automatic Ground Collision Avoidance System (Auto GCAS) was a run time assurance system in the same year, worked with me to develop a verification and validation strategy for autonomy in 2014, served on my Master's Thesis committee, and supported my application for Long Term Full Time training that enabled me to attend Georgia Tech for a year in residence to take 39 credits of coursework. Thank you to Jim Overholt for helping me figure out that I did want to do a PhD and for his studying advice that changed how I studied during my PhD. Thank you to Elizabeth Beecher for helping me figure out I wanted to do an Aerospace Engineering PhD rather than a Systems Engineering PhD. Thank you to Joe Nalepka for his mentoring to help me figure out I wanted to stay on a technical track. Thank you to David Casbeer, Misty Davies, and Natasha Neogi for introducing me to the professional society technical committee community. Thank you to Daryl Ahner for being my go-to testing, design of experiments, and statistics guru, and to Eric Swenson for his early mentoring in aerospace research. Thank you to Bryan Cannon, Laura Humphrey, and Chris Greek for supporting my research focus the last couple years.

Thanks to Jason Bowman, Dave Doman, David Casbeer, Isaac Weintraub, Eloy Garcia, Alex VonMoll, Adam Gerlach, Jose Camberos, Raphael Cohen, Alexandra Long, and many others who helped me push the limits of my controls and aircraft design knowledge as I prepared for my qualifying exams, one of the most challenging and intimidating experiences of my life.

Many thanks to Dr. Steve Ruffin, Oksana Gomez, Daurette Joseph, and Kamaria Richards

for helping figure out my coursework requirements and strategy on a short timeline, as well as to navigate the paperwork to complete my degree remotely while I was working full time at AFRL. Thank you to Alysia Watson for helping me make all the deadlines in 2017. Thank you to Jennifer Lynch, Garrison Lindholm, Kim Blanks, and Kim Ford for helping me with the AFRL paperwork for my year at Georgia Tech.

Finally, I'm exceptionally grateful to my family and friends who helped me stay sane over the last several years. Thank you to my parents Maryanne and Christopher as well as my sister Christine for fielding all the overwhelmed phone calls and encouraging me to keep going over the four years of undergrad and eight years of grad school. Also thanks to my friends Teressa Specht, Gretchen Hawkins, Sean Mahoney, Lee Ann Rutledge, Soleil Verse, Kevin Price, Harsh Shrivastava, and others who listened as I struggled and reminded me there was more to life than work. A special thanks to Dan Berrigan whose support as a friend and mentor is woven in every sentence of this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

One of the greatest challenges preventing use of advanced controllers in aerospace is developing methods to verify, validate, and certify them with high assurance. Traditional test and simulation-based approaches evaluate system behavior at design time in a subset of the total state space. Results from simulation and testing cannot be interpolated over systems with large state spaces, systems with nonlinear dynamics, systems that learn or degrade over time, systems operating under high uncertainty, or systems in complex and adversarial environments.

Run Time Assurance (RTA) systems are proposed as a complementary verification approach to facilitate near-term certification of advanced aerospace decision and control systems. RTA systems monitor the state of a cyber-physical system (CPS) online for violations of predetermined boundaries that trigger a switch to a simple, safety remediation controller. For example, automatic collision avoidance systems are RTA systems that monitor the CPS state for violations of proximity constraints and switch to a backup controller that assures safe separation. Design of RTA systems is generally ad hoc and specific to application, although common design elements and requirements of RTA systems cross applications and domains.

This research elicits, formally specifies, and analyzes RTA-based collision avoidance system requirements for a conceptual spacecraft conducting autonomous close-proximity operations. First, the Automatic Ground Collision Avoidance System developed for aircraft is studied to identify common design elements and requirements of RTA last-instant collision avoidance systems that cross the air and space domains. Second, formal requirements specification templates are developed for a generalized RTA architecture that extends the simplex architecture by accounting for human interaction, system faults, and safety interlocks. Third, formal requirements are elicited through the process of formal specification as well as from common design elements and requirements, spacecraft guidance constraints in the literature, and a structured hazard assessment. Finally, the requirements are analyzed using compositional reasoning and formal model checking verification techniques.

CHAPTER 1
INTRODUCTION

Automated collision avoidance systems are a form of run time assurance (RTA) that continually verify safety during operations. RTA systems go a step beyond alerting systems that inform human operators that an undesired condition is imminent. When undesired conditions are detected, RTA systems automatically react with a simple, assured response. An RTA monitor and set of automatic backup controllers, like automatic collision avoidance, assure operations adhere to desired properties regardless of the form of the primary controller. When bounded by a well-designed RTA system, the primary controller could theoretically be anywhere on a spectrum from human pilot to a fully autonomous decision and control system. RTA is an enabling technology for verification of increasingly autonomous systems for which certification criteria do not yet exist.

Development of specific criteria for the certification of new aerospace control systems lags the development of state-of-the-art aerospace control systems. For example, proportional-integral-derivative controllers were developed in the 1920s [1], while frequency response methods such as the Nyquist and Bode plots that yield the gain and phase margins describing stability weren't developed until more than a decade later [2, 3]. More complex and capable control systems design like adaptive control began to emerge in the 1950s [4, 5], while Lyapunov stability and [6] other verification approaches like run time assurance [7] appeared much more recently. Since the adaptive controller introduction, additional active research areas like robust, optimal, and more recently intelligent control have emerged pushing the state of the art in control design.

Despite significant advances in control theory, 1930s era gain and phase margins are given as criteria for certification [8]. In some cases, linearization about fixed points provides sufficient evidence [8]. In other cases, system developers can work directly with certification authorities to determine appropriate certification evidence for control systems designs on a case by case basis; however, this is a very long and costly approach. While alternative certification criteria for state-of-the-art control approaches may one day be developed, the certification process presents a signification barrier to operational use of many advanced control techniques.

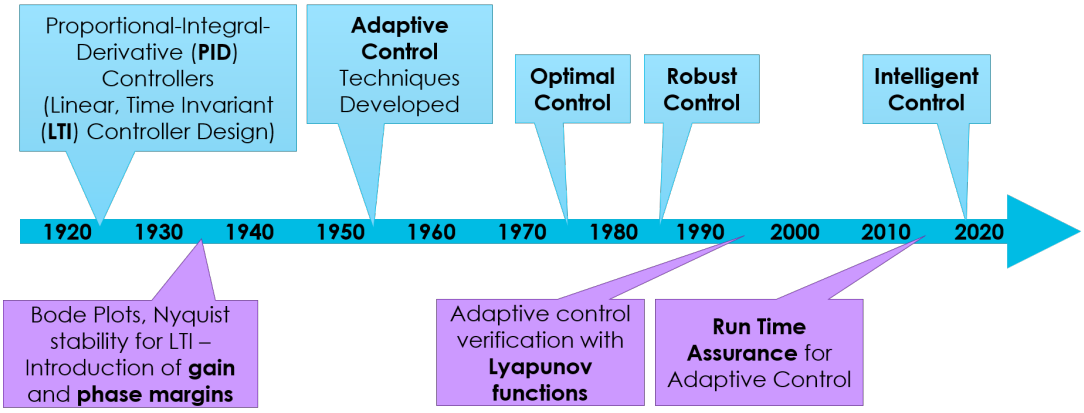


Figure 1.1: Timeline showing the introduction of control techniques in blue with verification approaches in purple lagging behind.

In addition to a lack of specific certification criteria, we've known for decades that test-based approaches are insufficient to completely verify correct system behavior to the levels required in aerospace [9]. This is echoed in other industries like autonomous vehicle development, where one study found that autonomous vehicles would have to be driven hundreds of billions of miles to be sufficiently tested [10].

Run time assurance (RTA) is a promising online verification approach to enable certification of complex control designs for which certification criteria do not exist yet, and is beginning to gain recognition in recent standards [11]. Ensuring decision and control systems always exhibit safe behavior is becoming an increasingly daunting task as the complexity of the controllers and the systems under control increase. While several approaches to RTA have been developed and successfully demonstrated, the design and verification of these systems is ad hoc and specific to the application.

1.1 Approach

This dissertation seeks to evaluate the following hypotheses through case studies in legacy and novel aerospace collision avoidance applications:

1. Common design elements and requirements for RTA systems cross the air and space domains.

2. Abstraction techniques, systems-theoretic accident analysis [12], and formal specification and analysis [13, 14, 15] provide a structured, rigorous, and generalizable approach to the

development and verification of high-level RTA system requirements before detailed design phases.

The first hypothesis is evaluated by development of a generalized RTA approach described in Chapter 4, informed by the Auto GCAS case study in Chapter 3, and demonstrated in Chapter 5 where automatic spacecraft collision avoidance requirements are traced to inspiration from the aircraft domain. The second hypothesis is evaluated through a case study in Chapter 5.

The research in this dissertation has three principle phases. First, a case study is conducted on the research and development of the Automatic Ground Collision Avoidance System (Auto GCAS) to determine key factors contributing to the success of a real-world RTA system. Auto GCAS is a particularly interesting case study because it is the product of more than three decades of research and prototypes, has been studied extensively by psychologists as a human-automation teaming example, and is unique as a system that takes control from a pilot in life or death situations [16, 17]. These key factors to success are identified in Chapter 3 and form a foundational set of common design elements and requirements for RTA systems that are expanded in Chapters 4 and 5.

Second, these common design elements and requirements are developed into design specification and requirements for a generalized, conceptual aerospace RTA. The RTA system verification problem is expressed as a set of finite state machine design specifications and linear temporal logic requirements, allowing the use of compositional model checking verification techniques. The general design specifications and requirements presented in Chapter 4 extend simplex architecture-based RTA systems by including human interaction, failure monitoring, and interlock monitoring components in the design. The design specification and requirements elicitation process builds on the foundational design elements identified in Auto GCAS by examining standards and guidance reviews, literature reviews, and systems-theoretic hazard analysis [12]. The process of formal specification and incremental analysis elicits additional previously unidentified requirements. Where applicable, reusable design specification and requirements patterns are identified.

Third, the approach is employed in the early specification and analysis of a novel, hypothetical, last instant collision avoidance system for a hypothetical future autonomous spacecraft. The space operations culture relies heavily on human operators for monitoring and human-coded commands [18, 19, 20, 21, 22, 23]. The approach presented in this research makes progress towards safe inte-

gration of autonomous capabilities onboard spacecraft. This collision avoidance system is scoped to prevent collisions in scenarios where two space objects feature low relative velocities (e.g. during autonomous rendezvous, proximity operations, and docking (ARPOD) missions). The automatic collision avoidance system monitors and intervenes when a black-box controller violates safe separation boundaries. Formal verification evidence is provided for the RTA-based design. Gaps in the formal verification capabilities and the appropriateness of various verification techniques are discussed.

1.2 Research Contribution

This research works towards combining aerospace RTA and formal methods, as well as filling a gap in the development of requirements for automatic and autonomous spacecraft maneuvering. "Specification is difficult, unglamorous, and arguably the biggest bottleneck facing verification and validation of aerospace, and other, autonomous systems [24]." This research seeks to make progress in the specification challenge. RTA systems and formal specification and analysis have been applied successfully in a variety of applications [7, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39] as is discussed in greater detail in Chapter 2. However, the ad-hoc implementations of RTA and formal methods by experts are application-specific and lack a generalized, repeatable approach. While recent standards provide some guidance to on the use of RTA [11] and formal methods [40, 15], the standards have not proposed formal specification and analysis of RTA. While widespread adoption of formal specification and analysis is partially hindered by the lack of design templates for implementation [41], this work is presented in the form of specification patterns and provides a conceptual example application. Some standards provide guidance on spacecraft systems engineering [42, 43], and space autonomy requirements engineering [44], and regulations are placed on satellite communication frequencies [45]. However, no standard or requirements exist to ensure safety of autonomous spacecraft operations. In addition, the White House [46], European Space Agency [47], and others have called for the development of a Space Traffic Management (STM) system, and collision avoidance has been identified as a top priority for success of STM [48]. However, no international body has published requirements for automatic collision avoidance in the context of STM. The contributions of this dissertation are as follows:

- a case study of the Automatic Ground Collision Avoidance System (Auto GCAS) for aircraft to inform critical components of a general aerospace automated collision avoidance RTA;
- generalized formal requirements for a decision module of an RTA architecture that includes a fault monitor, interlock monitor, and human-machine interface components;
- safety requirements to bound automated or autonomous maneuvering of spacecraft elicited from top-down system safety analysis;
- first development of a set of requirements for last-instant automatic collision avoidance system for spacecraft; and
- an evaluation of the application of two requirements analysis tools.

1.3 Content Overview

Chapter 2 provides background on practices in systems engineering with context for the use of formal methods and run time assurance approaches. In Chapter 3, Auto GCAS is studied for key factors contributing to the success of a real-world RTA system. These well-studied success factors are then used to construct a novel RTA design and verification approach in Chapter 4, presented in the form of general patterns and templates for design specifications, requirements, and hazard analysis. The RTA approach is then evaluated through application to a hypothetical spacecraft automatic collision avoidance system in Chapter 5. Finally, Chapter 6 discusses conclusion of the work and recommendations for research to improve upon the initial approach.

CHAPTER 2
CURRENT STATE OF VERIFICATION AND VALIDATION FOR AEROSPACE CONTROL SOFTWARE

Several unique concepts come together to create the specification and analysis approach for RTA presented and evaluated in this dissertation. Section 2.1 provides context for the proposed approach by defining verification, validation, certification and assurance. Then Section 2.1 introduces various models for systems engineering, discusses where design errors are introduced and identified, and describes a new systems engineering model for increasingly autonomous systems that features the approach presented in this dissertation. Practical considerations for using formal verification are discussed in Section 2.2, including the state space explosion that hinders the use of formal methods, methods to abstract systems to facilitate formal analysis, approaches to systems-level metamodeling and metalevel analysis, necessary concepts to develop formal specifications, formal specification patterns that facilitate easier reuse, the capabilities of automated formal analysis approaches, and current requirement specification and analysis tools. Third, context and discussion of decision and control system hierarchies in Section 2.3 provides insight into the appropriate place to use run time assurance techniques discussed in Section 2.4. This research is not the first use of RTA, RTA in aerospace, or RTA and formal methods and Section 2.4 previous work. One of the important aspects of specifying RTA systems is formally specifying safe boundary conditions, and Section 2.5 discusses hazard-analysis based techniques for requirements elicitation. Finally, as a formal RTA approach isn't appropriate universally, unique characteristics of aerospace systems development that make them a particularly attractive application of RTA are discussed in Section 2.6.

2.1 Verification, Validation, Certification and Assurance

Developing methods to verify, validate, certify, and assure advanced automated and autonomous decision and control systems presents one of the largest barriers to operational use of autonomy. Verification investigates whether a system meets requirements [49], validation assesses whether a system meets the needs of the end user [50], certification determines whether a system conforms to

a set of criteria or standards for a class of similar systems [8], and assurance is justified confidence that the system functions as intended with limited vulnerability to threats and hazards based on evidence generated through development activities [51]. Verification, validation, certification, and assurance activities intersect at early requirements and architecture development phases and correct design relies on early rigor.

In this dissertation, verification, validation, certification, and assurance are all addressed at their intersection in early system design. The proposed approach formally verifies the RTA design meets requirements, while at the same time utilizing a formal, unambiguous requirements development process that facilitates validation that the requirements constrain the design as intended. Where applicable, guidance from certification standards is considered. The proposed approach generates evidence as part of assurance activities.

2.1.1 Systems Engineering Models

Many systems engineering models such as the "V," waterfall, spiral, and twin peaks models are used to describe the stages of the systems engineering development process and interactions between the stages. One of the many instantiations of the Systems Engineering "V" is shown in Fig. 2.1. This model shows the steps in the development of a new system from concept to operations. While there are some slight variations between sources [52, 53, 54], the systems engineering "V" includes the following core steps: development of a concept of operations, requirements elicitation, high level architecture design specification, detailed design, hardware and software implementation, unit testing, subsystem verification, system level verification, system level validation, and transition to operational use and sustainment until retirement.

One of the core issues with the systems engineering "V" is the linear timeline it generally assumes. A similar method called the Waterfall model, shown in Fig. 2.2, also depicts systems engineering phases as a process where progress flows down through the phases, but emphasizes that feedback should also flow back up to iterate and refine the products in each phase [55]. Another influential systems engineering model that rejects the notion of sequential design phases is the spiral model [56]. The spiral model depicted in Fig. 2.3 emphasizes that phases should be developed concurrently and constantly refined throughout the systems engineering process. Problem frames in conjunction with the Twin Peaks model [57] as depicted in Fig. 2.4 emphasizes iterating between

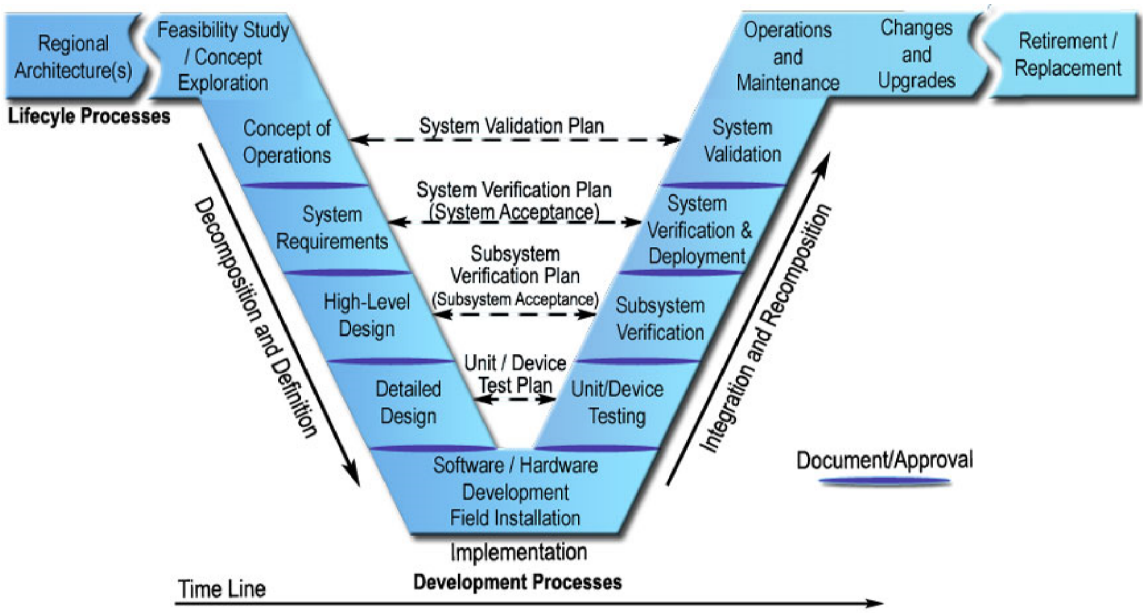


Figure 2.1: Systems Engineering "V" [53]

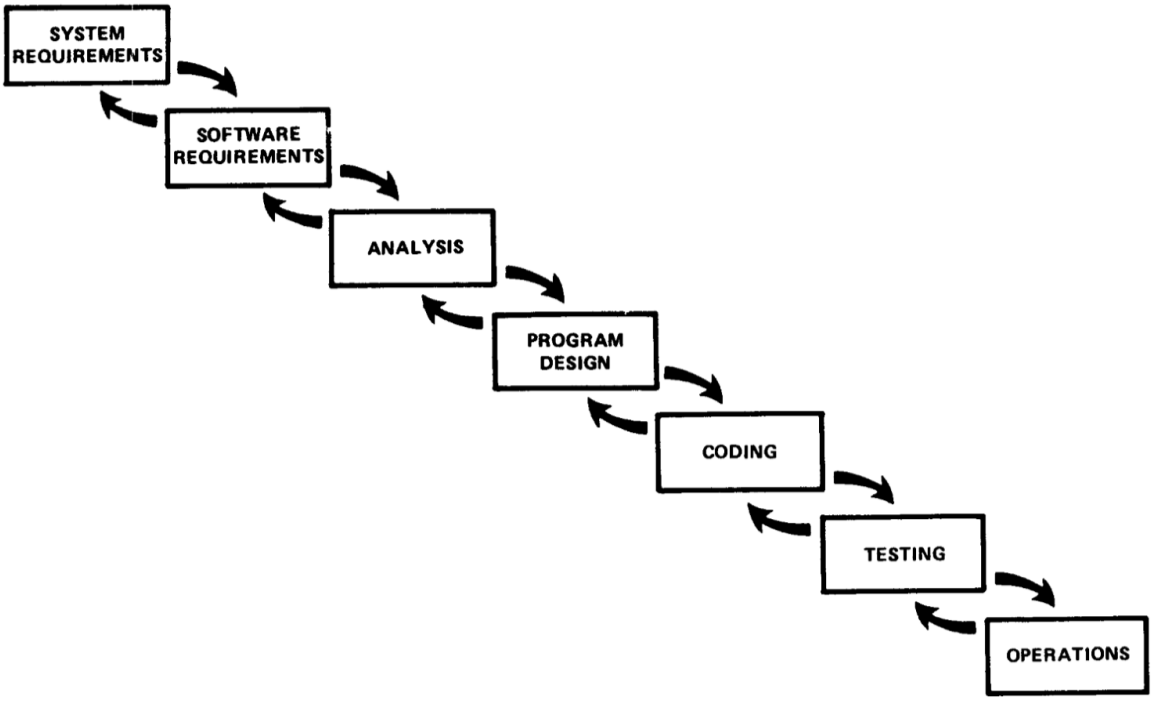


Figure 2.2: Systems Engineering Waterfall Model [55]

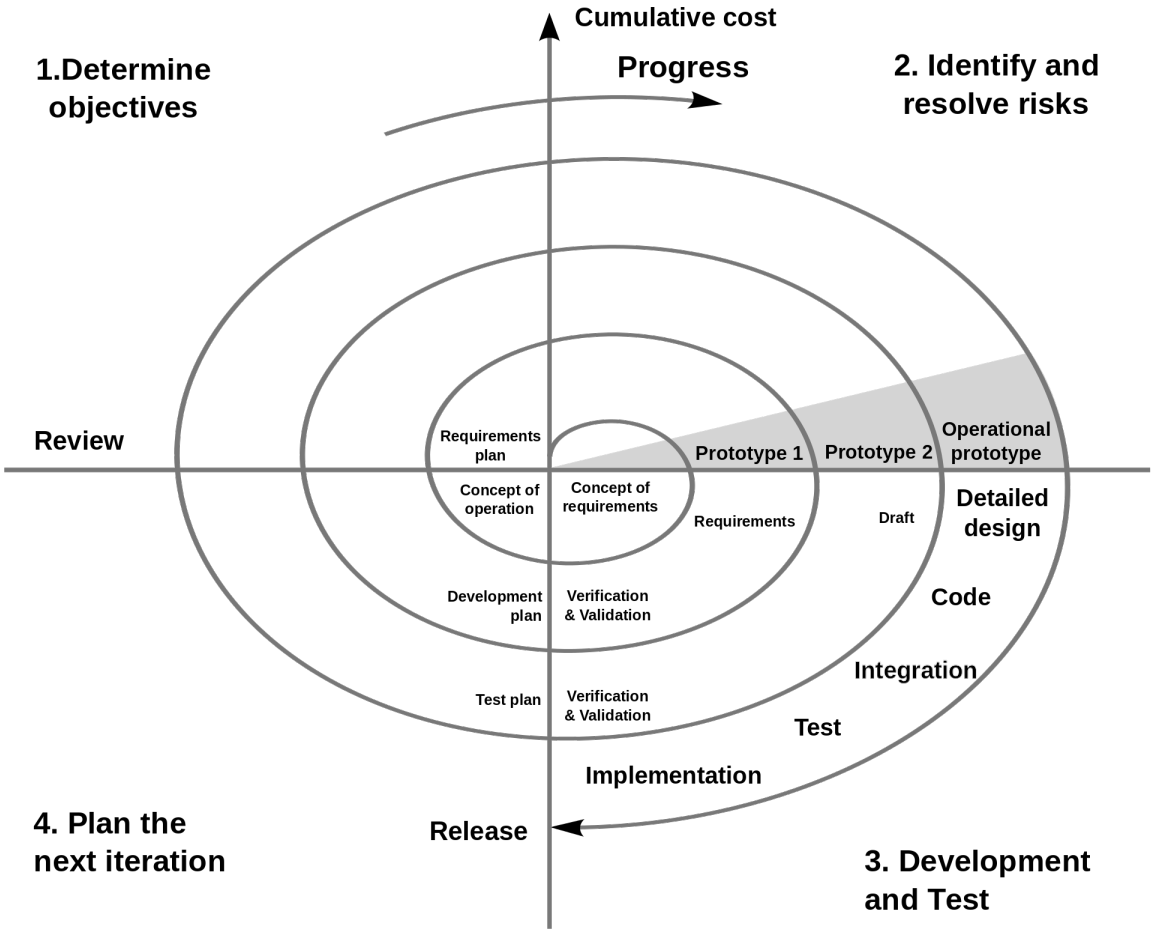


Figure 2.3: Systems Engineering Spiral Model [56]

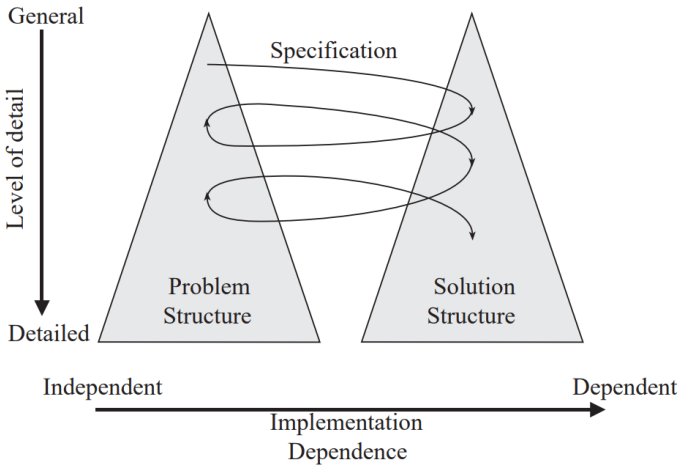


Figure 2.4: Twin Peaks Software Systems Engineering Model [57]

the problem and solution spaces so that architectural considerations are included in the requirements elicitation process. This model relates requirements and architectures via problems frames which start with abstract behavioral requirements and architectures which are refined together.

In this dissertation The design approach presented in this may be implemented in an iterative fashion where feedback flows up through levels. The approach provides a rigorous, formal foundation for continual refinement and iterative verification throughout the design process.

2.1.2 Introduction and Identification of Design Errors

In the software systems engineering process, 70% of faults are introduced in the requirements and architecture development phases, while 80% are found after integration in the verification, validation, or operations when the cost to repair them can be as much as 1000 times higher than if it was found earlier in the design process [58, 59, 60, 61]. As a result, in addition to employing a more iterative design process, there is a signification opportunity for increased analysis early in the systems engineering design cycle. This dissertation presents and evaluates a approach that focuses on the early design specification, requirements, architecture development, and hazard analysis phases.

2.1.3 Connection to Certification and Assurance

At its core, RTA ensures that a system stays within a set of acceptable states while it operates. This is different from design time verification, which generally relies on showing that an carefully selected set of possible states always yield an acceptable response. For example, MIL-HDBK-516C Airworthiness Certification Criteria [8] specifies that a controller demonstrate a gain margin of 6 dB and a phase margin of 45 degrees, which apply to linear time invariant controllers like PID controllers, but do not apply to nonlinear, adaptive, or other more complex systems designs. This certification criteria attempts to cover the entire state space of the controller by evaluating the system response to a sinusoidal input across wide ranges of frequencies and magnitudes. While similar metrics may one day be developed for nonlinear and adaptive control systems that currently represent the state of the art, the *development of certification criteria will likely lag behind the development of decision and control systems for the foreseeable future*.

Complexity of the control system designs which promise superior performance has passed a point where traditional design time verification techniques can scale to cover the entire space of

possible states. When flight test can cost tens of thousands of dollars an hour or the risk of losing the test asset is too high, the range of points that can be tested decreases, and simulation-based verification techniques fill some gaps. However, stochastic simulation approaches like Monte Carlo still leave coverage gaps in the evaluation. Set based simulation and reachability evaluation methods provide some promise of greater coverage across a continuous range of states; however, even these methods struggle to scale to the number of states or for long periods of time. Run time verification approaches, such as the architecture proposed in this dissertation, present a way to leverage the state of the art in control system design while assuring safety online by examining a specific set of states for a limited time horizon and adapting in real time to preserve system safety. Offline verification approaches analyze, simulate, or test system components prior to deployment, often looking at results after the verification task is completed. By contrast online verification approaches continuously evaluate a system while it is running.

2.1.4 A New Systems Engineering Model for Increasingly Autonomous Systems

In an age of increasingly complex and autonomous systems, a new paradigm for systems engineering is needed. Through years of research and a series of workshops with industry, academia, and government verification and validation researchers and practitioners [62], a new systems engineering model co-developed by the author of this dissertation is proposed as depicted in Fig. 2.5. This system engineering model is theoretical and while portions have been successfully implemented, it is not yet a fully mature process.

As discussed in Section 2.1.2, 70% of errors are introduced in requirements and architecture phases, so the approach places heavy emphasis on activities in these phases. Planning and analysis in the early design stages is critical to understand how the system will demonstrate conformance to requirements, forms the foundation of the assurance profile, and guides development of the system with incremental verification and validation activities.

An integrated design and assurance profile generation is comprised of two phases conducted in parallel. The first early design phase is *requirements elicitation*, which can be divided into four separate interconnected and iterative activities:

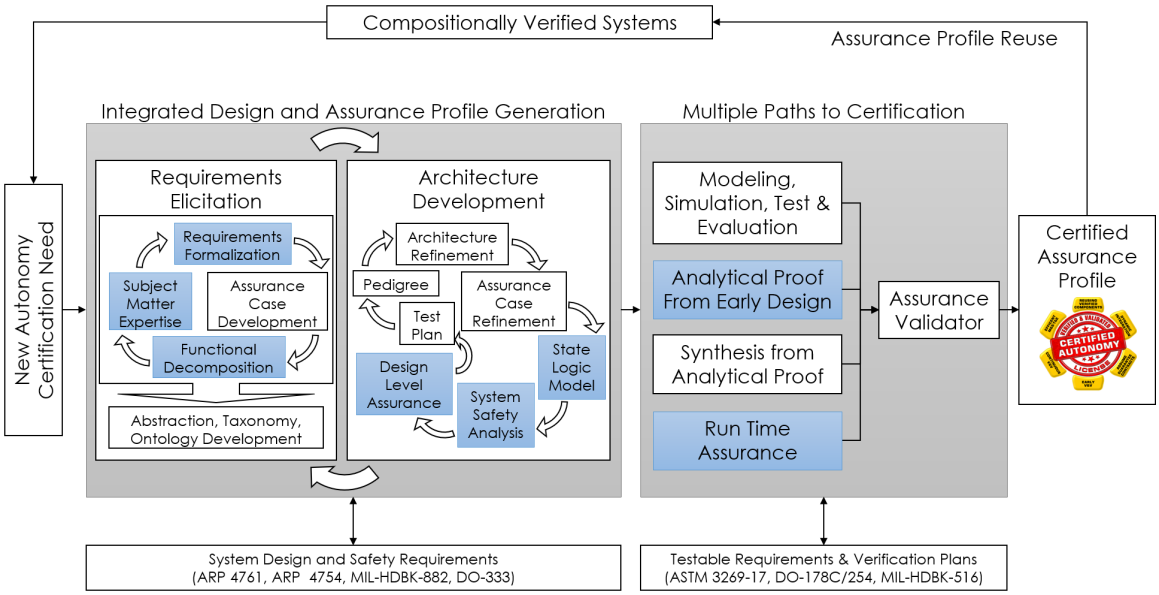


Figure 2.5: Systems engineering for increasingly complexity and autonomy, with items addressed in this research in blue [63, 64, 65, 66].

1. The *subject matter expertise (SME)* activity includes talking to operators (pilots, mission controllers, etc.) to develop an intent specification [67] and mission threads and vignettes that elicit and document SME input.

2. The *requirements formalization* activity includes extracting and formally defining requirements to meet mission capabilities identified in the SME activity.

3. The *assurance case development* activity includes building the skeleton of a structured argument in the form of an assurance case that the system will meet requirements.

4. The *functional decomposition* activity includes decomposing the requirements into functional components that form the baseline of a conceptual architecture in the first phase of the architecture development.

Throughout the process of requirements elicitation, complex functions are abstracted until they can be further refined in the design, components are named and classified in a taxonomy, and relationships between components and concept are defined in an ontology. This process of abstraction, taxonomy creation, and ontology development may be implicit (kept in the heads of the specifiers and designers) or explicit (formally described so that it can be reasoned on).

The second early design phase is *architecture development*, an iterative refinement process which includes development of important artifacts for the assurance profile. The initial architecture of a system emerges from the design specifications in the requirements phase. The following are important activities in the architecture development stage:

1. The *architecture refinement* activity includes incremental refinement of the architecture from conceptual, to reference, to objective, to system architecture.

2. The *assurance case refinement* activity grows the assurance case in an iterative and incremental process to incorporate more refined functionality, component definitions, and interface definitions as design decisions are made in architecture refinement.

3. The *state logic model* activity utilizes models such as state machines to formally define system and component-level states. This formal specification enables analysis techniques such as model checking to ensure that a section of the requirements is met by the design.

4. The *system safety analysis* activity applies appropriate analysis methods incrementally as the system is developed and may include Functional Hazard Assessments (FHA), Preliminary System Safety Assessment (PSSA), Failure Modes and Effects Analysis (FMEA), Fault Tree Analysis (FTA), Systems Theoretic Accident Model and Processes (STAMP), and Systems Theoretic Process Analysis (STPA).

5. The *design level assurance* activity identifies system interlock conditions or safety boundaries informed by the system safety analysis. This activity includes development of design time and runtime monitoring and mitigation strategies.

6. The *test plan* activity incrementally develops a test plan to evaluate system functionality.

7. The *pedigree* or heritage activity documents previous uses of the component including previous verification evidence and assurance profiles as appropriate. This activity may pull in evidence or analysis from previous implementations under the appropriate context for reuse in the analysis of this architectural implementation.

In the later design phases, multiple paths to certification are envisioned. *Modeling, simulation, and traditional test and evaluation* are expected to be part of the certification process for many years

to come. However, traditional simulation and testing approaches alone are likely insufficient and infeasible as the primary source of verification evidence for autonomy. For example, in 2016 the RAND Corporation published a study that autonomous vehicles would need to be driven hundreds of billions of miles to demonstrate their reliability and concluded that new methods and adaptive regulations are needed to provide sufficient evidence of vehicle safety [10]. While the presence of these traditional verification activities will not change, it is possible the selection of what to simulate and test may be guided and supplemented by new methods. One form of supplemental evidence is *analytical proof* generated from the requirements elicitation and architecture development phases. This analytical proof topic is the focus of Section 2.2. Another supplemental evidence path to certification is to *synthesize* software automatically from formal specifications [68]. Finally, in cases where analytical proof or traditional verification is not possible, *run time assurance* paradigms may be utilized to a monitor behavior online and switch controllers as described in Section 2.4. These multiple paths are combined by an assurance validator, which may be a human or in the future an automated system into an assurance profile, a complete argument of assurance made up of heterogeneous evidence. One of the most challenging problems will be generating evidence and assurance arguments in a modular way that enables arguments and evidence from a variety of sources to be composed into a reusable library.

The design and assurance phases should be conducted in accordance with system design and safety standards such as Aerospace Recommended Practice (ARP) from the Society of Automotive Engineers (SAE) International, Radio Technical Commission for Aeronautics (RTCA) guidance, American Society for Testing and Materials (ASTM) standards, as well as military and federal standards, handbooks, and regulations. Where appropriate, the following standards, handbooks, and guidance should be consulted:

- ARP4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment [69]
- ARP4754: Guidelines For Development Of Civil Aircraft and Systems [70]
- RTCA DO-254: Design Assurance Guidance for Airborne Electronic Hardware [71]
- RTCA DO-178: Software Considerations in Airborne Systems and Equipment Certification

[40]

- RTCA DO-333: Formal Methods Supplement to DO-178C and DO-278A [15]
- ASTM 3269-17: Standard Practice for Methods to Safely Bound Flight Behavior of Unmanned Aircraft Systems Containing Complex Functions [11]
- MIL-HDBK-516C: Airworthiness Certification Criteria [8]
- MIL-STD-882E: System Safety [72]
- FAA's Standard Airworthiness Certification Regulations [73]

In this dissertation, as a part of the requirements elicitation activity, SMEs are consulted in the areas of collision avoidance, requirements are extracted from previous collision avoidance programs and formalized, and the functions of the run time assurance system are decomposed into functional components forming the baseline of the proposed high-level RTA architecture. In the architecture development phase, design specifications describe functional system behavior in the form of state logic models, and system safety analysis in the form of STAMP and STPA is used to define design level assurance. Together these selected phases of requirements elicitation and architecture development were used to create the run time assurance approach, design specifications, requirements, and hazard analysis presented and evaluated in this dissertation. The development of an assurance case in either the requirements or architecture phase, as well as the development of a test plan or pedigree analysis, is beyond the scope of this dissertation.

2.2 State Space Explosion, System Abstraction, and Model-based Formal Verification

2.2.1 State Space Explosion Problem

One of the major challenges of verification is scaling for systems with large numbers of states. Autonomous systems may need to include a high number state variables as inputs to their complex or intelligent decision making algorithms. The *state space explosion problem* is that for N variables with k possible values, the number of possible states grows exponentially to k^N [13]. Complete verification of these systems requires checking that a system meets requirements under all k^N states. The graph of possible states can be extremely large or infinite, especially in systems with continuous

dynamics. Certain statements about infinite state spaces are usually undecidable, i.e. a decision problem where it is impossible to construct a computer algorithm that will always return a correct answer [13]. *Cardinality* is the number of states in a transition system model and the main factor in the *verification algorithm runtime* [13]. In this dissertation many continuous variables are used to describe the dynamics of the system (position, velocity, orientation, etc.), as well as discrete and enumerated variables describing conditions of the system.

2.2.2 Formal Methods

Formal methods are defined as applied mathematics for modeling and analyzing systems with mathematical rigor [13] and include proof-based techniques, static analysis, and run time approaches. Proof-based formal methods techniques for verification can be divided into axiomatic and semantic approaches [74]. Axiomatic approaches reduce the verification problem to a proof problem include methods rooted in Hoare logic [75], as well as theorem proving approaches. Theorem-proving techniques are complex with a step learning curve and require human interaction and ingenuity to formulate and solve. Semantic problems use exhaustive search techniques to evaluate all possible program executions for violations of formal specifications and include model checking approaches [13]. Model checking produces a proof that system meets desired properties or a counter example. Model checking can be more automated, but may require abstraction to use on very large systems. Static analysis and run time approaches are also often considered formal methods. Static analysis methods, such as abstract interpretation [76], screen software for errors by inferring properties without executing program but are vulnerable to false alarms[77]. Run time approaches are discussed in Section 2.4. Model checking is used in this research.

2.2.3 System Abstraction

One method of dealing with the state space explosion problem is conducting verification at higher levels of abstraction. An *abstraction* is a model of a system that removes implementation details while preserving properties of the system to be analyzed [13]. Models may exist at multiple levels of abstraction where lower levels refine the abstract model with more implementation details.

All design processes use some form of abstraction. Abstractions are useful approximations of the design that facilitate understanding of the system and enable early analysis. For example, writing

an algorithm in a programming language is working with an abstraction of the actual 1's and 0's of the binary that the computer uses to execute the algorithm. Abstractions are not constrained to the digital domain either. In physics, a free body diagram is an abstract representation of the forces and moments acting on an object that assumes locations at specific points. Designs can also be described at various levels of abstraction. For example, at a high level of abstraction, it may be assumed that the system orientation may rotate within a range of rates. Lower levels of abstraction may describe selection of a type of actuator such as a momentum exchange device. Greater refinement might include the actual mass and angular acceleration and velocity characteristics of a specific reaction wheel array attitude control actuator.

Two important tools in system abstraction are atomic propositions and labeling functions. *Atomic propositions* are formal expressions of temporal characteristics that can be evaluated as true for false (e.g. $x \leq 10$). A *labeling function* L of a state s, denoted $L(s)$ is a form of abstraction where a Boolean variable can be used as a substitution for a set of atomic propositions which are satisfied by a state. A propositional logic formula Φ satisfies s if and only if it satisfies $L(s)$ [13].

A popular system model abstraction in computer science is a *transition system* directed graph, where states are nodes and transitions are edges [78, 13]. The *state* of a system is a collection of variables describing its behavior at a specific moment in time. For example, the state could include the power setting (an enumerated and discrete on or off value) variable and the distance from the origin variable (a continuous real numbered value), as well as many other variable descriptions. The *transitions* of a system describe the evolution from one state to another; for example, from on to off or a change in distance, etc.

Definition 1. Transition System. A transition system TS is a tuple $(S, Act, \rightarrow, I, AP, L)$ where

S is a set of states

Act is a set of actions

$\rightarrow \subset S \times Act \times S$ is a transition relation

$I \rightarrow \subset S$ is a set of initial states

AP is a set of atomic propositions, and

$L : S \rightarrow 2^{AP}$ is a labeling function.

TS is finite if S, Act and AP are finite.

In this dissertation, design specifications are captured in the form of transition systems and requirements are captured in the form of atomic propositions. Labeling functions are used in this dissertation to abstract atomic propositions when their complexity is enough that the specification becomes difficult to interpret, or as placeholders where more detailed knowledge of the system design is needed in later design iterations to generate an atomic proposition.

2.2.4 Formal Specification

Formal methods are a collection of notations and techniques based on mathematical theories, logic, automata or graph theory for describing and analyzing systems [79]. Three goals of formal methods are specification, analysis, and synthesis. Formal specification facilitates, a common, unambiguous understanding of the system between users, designers, programmers, and verifiers. Formal analysis helps to find errors and increase reliability of the system. Formal synthesis integrates formal methods into the development process to convert a design to implementation with some level of automation. Formal specification is described in this section, formal analysis is described in Section 2.2.8, and formal synthesis is beyond the scope of this dissertation.

Specification of design and requirements in formal logic enables automatic analysis of a system design early in the design phases to find design flaws and conflicts between requirements at multiple levels in the design. Formal *design specifications* describe the transition behavior of systems between states, a definition derived from a computer science definition of *transition systems*. Formal *requirements* describe atomic constraints or conditions on the system behavior. In this case, *atomic* means that the statements are evaluated as true or false and cannot be divided into smaller statements.

In computer science communities, design specifications are often referred to as "requirements" and requirements are referred to as "properties." In computer science, properties (called requirements in this dissertation) may describe *functional correctness* (system performs as expected), *reachability* (set of states reachable from an initial set of states), *invariance* ("something bad never happens," i.e. a safety property that is always true), *liveness* ("something good eventually happens," i.e. a property that will eventually be true), *fairness* (ensuring balanced, proper repetition of events), and *real-time* properties (acting as expected in time) [13, 78]. In this dissertation, most of the requirements fall into the category of functional correctness, reachability, invariant, and real-time.

2.2.5 Logic Notation and Specification

Formal specification may use a variety of different logics and languages. *First order logic* uses quantifiers (such as "for all" and "there exists") and relations (such as inequalities, addition, and multiplication) to reason over variables in a predefined domain like reals or integers, and is widely used in calculus and geometry as well as databases and artificial intelligence [79]. If-then-else and while loops are special kinds of first order logic. Logic symbols used in first order logic are listed in Table 2.1, and the precedence (order of operation) of the operators is: $\neg$, $\wedge$ (and $\bigwedge_{i=1,n} \mu_i$), $\vee$ (and $\bigwedge_{i=1,m} \nu_i$), $\forall$, $\exists$, $\rightarrow$.

Propositional logic is a simpler formalism than first order with a more restrictive set of symbols that does not include quantification ($\forall$ or $\exists$) or function or relation (including equivalence $\equiv$) symbols [79].

Table 2.1: Logic Symbols

Description	Symbol
In	$\in$
Implies	$\rightarrow$
Not	$\neg$
Or	$\vee$
And	$\wedge$
Relation	rel
Equivalent to	$\equiv$
For all	$\forall$
There exists	$\exists$
Satisfies	$\models$
Conjunct	$\bigwedge_{i=1,n} \mu_i$
Disjunct	$\bigwedge_{i=1,m} \nu_i$
Equal to (outside logic)	=
Variant	$a[d, v](u)$

Modal logic extends propositional logic with two additional connectives, $\Box$ ("necessarily") and $\Diamond$ ("possibly"). Note, in temporal logic, the square and diamond correspond to slightly different definitions of "always" or "globally" and "eventually," respectively. Modal logic is used to create and reason about finite state machines or finite transition systems. In this research, the design specifications create a model or transition system, as described in Section 2.2.3. One way to describe these state models is with Kripke semantics.

Definition 2. Model. A model $\mathcal{M}$ is defined by [80]:

a set W, whose elements are called worlds (i.e. states of the system);

a relation R on W ($R \subset W \times W$), called the accessibility relation

a labelling function $L : W \rightarrow \mathcal{P}(\texttt{Atoms})$.

$R(x, y)$ denotes that (x, y) is in R. The set W is often drawn as a set of circles, containing a labelling function L, with arrows between the circles corresponding to the relations R.

Temporal logic extends propositional logic with operators that represent and reasons about systems over time [13]. A variety of formal languages have been developed to model systems in temporal logic. Linear temporal logic (LTL) models sequences of events in a linear progression along a single path. Computational tree logic (CTL) models branching time, where each branch represents different decisions, and includes operators to define statements such as "there exists a path" and "along all paths" [81, 13]. Timed computational tree logic (TCTL) extends CTL with continuous time propositions and expresses properties of timed automata. Another popular temporal logic used is metric temporal logic (MTL) which uses integer time constraints [82].

In some cases temporal logics are constrained to only reason about future cases, as is the case with LTL. Other logics, such as *past time linear temporal logic* (ptLTL), use temporal operators to describe the past states of an execution trace relative to the current point of reference [83]. Where an LTL statement might say state a implies in the next step that state or condition b is true ($a \implies \bigcirc b$), ptLTL might instead say that if in the last step a is true, it implies b is now true ($\bigcirc^{-1} a \implies b$). An example of this is found in the decision logic design specification DL02 which states that "Once the system enters the failed state (F), the system shall remain in the failed state (F) if a failure condition (f) is still detected," expressed in ptLTL as $(\bigcirc^{-1} F \wedge f) \implies F$. In a model checking tool used in this research as described in Section 2.2.9, this requirement is written `DL02 : (prevstate == FAILED and system_state.f) implies (op_state == FAILED)`. This "last state and a current condition implies current state" form is in contrast to saying that if the system is in a current state and condition, then the next state will be a specific state. The LTL version of DL02 would be written $(F \wedge f) \implies \bigcirc F$. In 2003 it was proven that ptLTL can be exponentially more succinct that LTL [84]. A list of ptLTL symbols is listed in precedence order in Table 2.2. In this dissertation, ptLTL is used to specify requirements.

Table 2.2: Summary of Past Time Linear Temporal Logic (ptLTL) Symbols used in this Dissertation.

Symbol	Description	Translation	Alternative Symbols
$\Box$	historically/always	"always"	$[*]$
$\bigcirc^{-1}$	previous step	"previously"	
$\neg$	negation	"not"	!
$\cup$	until	"until"	U
$\wedge$	conjunction	"and"	
$\vee$	disjunction	"or"	
$\Rightarrow$	implication	"implies"	$->$
$\Leftrightarrow$	logical equivalence	"is equivalent to"	$<->$

Definition 3. Semantics of ptLTL. A linear structure π over a finite set of propositions $\mathcal{A}$ is a function $\pi : \mathbb{N} \rightarrow 2^{\mathcal{A}}$ [85]. Let π be a linear structure over $\mathcal{A}$, let φ and ψ be ptLTL formulas, and let $i, j, k \in \mathbb{N}$. φ is true in π, written $\pi \models \varphi$ if and only if (iff) $(\pi, 0) \models \varphi$. Then φ holds in π at time i, written $(\pi, i) \models \varphi$, is inductively defined as follows [85]:

$(\pi, i) \models p$ iff $p \in \pi(i)$

$(\pi, i) \models \Box\varphi$ iff $\forall j \leq i, (\pi, j) \models \varphi$

$(\pi, i) \models \bigcirc^{-1}\varphi$ iff $i > 0$ and $(\pi, i-1) \models \varphi$

$(\pi, i) \models \neg\varphi$ iff $(\pi, i) \not\models \varphi$

$(\pi, i) \models \varphi \cup \psi$ iif $\exists j \leq i, ((\pi, j) \models \psi$ and $\forall k : j < k \leq i, (\pi, k) \models \varphi$

$(\pi, i) \models \varphi \wedge \psi$ iff $(\pi, i) \models \varphi$ and $(\pi, i) \models \psi$

$(\pi, i) \models \varphi \vee \psi$ iff $(\pi, i) \models \varphi$ or $(\pi, i) \models \psi$

2.2.6 Tabular Requirements Formats

In this research, formal requirements are documented in tables. While these tables do not follow a standard convention, several different conventions have been created over the years. An early case of requirements specifications tables is Parnas tables [86, 87], which separate functional and relational expression cases into rows and columns. There are several types of Parnas tables including normal tables (headers are values and inner cells are expressions), inverted normal tables (headers are expressions and inner cells are value assignments), decision tables (for tuple domains), vector tables (for tuple ranges), and several others [86]. Parnas tables proved very useful in development of the U.S. Naval A-7E aircraft [88], particularly in the software requirement specifications, which

aided subject matter experts and pilots in funding hundreds of detailed errors. Lessons learned from the program were incorporated in the Navy's Software Cost Reduction (SCR) requirements method [89].

2.2.7 Patterns

Often a system specification will use a limited set of requirements *patterns*, defined as sets of logical statements with the same syntax. Perhaps the most influential work on formal property patterns was that by Dwyer et. al. [90, 91]. Motivated by a hypothesis that practitioners did not use formal methods because formal languages are difficult to write correctly, read or understand, they created a specification pattern system that is a "collection of parameterizable, high-level, formalism-independent specification abstractions," that cover 95% of 500 examples of property specifications surveyed. Just as is done in this research, Dwyer et. al. [91] model systems as finite transition systems and properties as formal constraints on the transition behavior. The patterns contain five primary kinds of *scopes* that describe the conditions over which the specification must hold (beginning with the starting state up to but not including the ending state):

- global (throughout the entire program execution),
- before (up to a specific state or event),
- after (following a specific state or event),
- between (after one state or event and before another), and
- after-until (a modification of between where the condition continues even if the second state or event never occurs).

They also introduced eight primary patterns, which the authors will refer to as *predicates*, that describe what condition shall occur in the given scope, including:

- absence (state or event does not occur),
- existence (state or event must occur),
- bounded existence (state or event must occur at least, at most, or exactly k times),

- universality (state or event occurs for entire scope),
- precedence (state or event P is always preceded by state or event Q),
- response (state or event P is always followed by state or event Q),
- chain precedence (a sequence of events $P_1, ..., P_n$ must be preceded by a state of events $Q_1, ..., Q_m$), and
- chain response (a sequence of events $P_1, ..., P_n$ must be followed by a state of events $Q_1, ..., Q_m$).

More recently, a set of robotic mission patterns have been developed in linear temporal logic [92, 93]. These patterns are organized in categories of core movement patterns, avoidance patterns, and triggers and are evaluated across six experiments for coverage in real world missions and correctness.

In this dissertation, it is hypothesized that many formal specifications for collision avoidance will fall into a finite set of patterns. The results are described in Section 4.2.

2.2.8 Automated Formal Analysis

Formal analysis of design specification items and requirements evaluates logical entailment, logical consistency, realizability, and traceability.

- *Logical entailment* analysis shows that a metamodel defined by design specifications meets the atomic safety requirements. This is completed by evaluating that a conjunction of the design specifications and assumptions satisfy requirements ($A_1 \wedge A_2 \wedge ... \wedge A_n \wedge D_1 \wedge D_2 \wedge ... \wedge D_m \implies R_i$). Logical entailment is usually shown by proof. In some cases proof can be generated automatically using tools like model checking. When the expression is too complex interactive proof tools like theorem provers are typically employed. *Model checking* [81, 94] is an automatic verification technique that shows a model of a concurrent system satisfies properties in temporal logic. Several techniques may be applied to do this including binary decision trees [95], k-induction [96], and property directed graphs [97]. The logical entailment in this work is accomplished with k-induction model checking.
- *Logical Consistency* checks that the design specification items and assumptions are free of conflicts for N time steps.

- *Realizability* checks that the design specification can coexist for all possible input conditions that satisfy the requirements ($A_1 \wedge A_2 \wedge ... \wedge A_n \wedge D_1 \wedge D_2 \wedge ... \wedge D_m$). The realizability algorithms used in this work are based on the work of [98].

- *Traceability analysis*, based on the techniques in [99] shows which design specifications fulfill each of the requirements. In complex state machines, traceability analysis may be used to analyze the system logic to determine if there is a reduced set of logic may be used to describe the system transitions. The benefit of reducing the logic with logic equivalence is that it may be easier to implement and verify in a complete system. The disadvantage is that the intuition is lost as to why the logic is written a certain way. During early system analysis, an intuitive set of logic may be constructed and proved equivalent to a simpler set of logic which is implemented. One way to reduce to a smaller, logically equivalent set is to evaluate whether the design specifications which define the state machine are unnecessary to meet the desired requirements.

As described in Section 2.1.2, the majority of errors are introduced in early design phases. Introducing formal methods analysis is best in early design phases because the design is still relatively small and errors are less expensive to fix [79]. *Compositional verification* helps to verify or test portions of code separately and then make conclusions about the system as a whole, which is helpful in modern coding practices where coding tasks are divided over large teams [79]. Compositional verification includes statements such as: if property p_1 for subsystem A and property p_2 for subsystem B both hold, it implies property p holds for the entire system.

In this dissertation the verification approach conducts logical entailment to ensure the design specifications meet the desired safety properties, logical consistency checking of the design specifications to ensure they are free of conflicts, realizability checking to ensure that all the design specifications can coexist for all possible input conditions that satisfy assumptions, traceability analysis to show that the design specifications tie to each of the requirements, and compositional verification to show that design specifications and requirements in lower level functional components satisfy higher system-level design specifications and requirements.

2.2.9 Requirements Specification and Analysis Tools

A variety of methods and tools are used in practice to document and analyze requirements, however they differ on how formal they approach requirements specification and whether they enable analysis. The least formal and most popular tools to document requirements used are spreadsheets and word processors. From there, the Systems Modeling Language (SysML) [100, 101] and IBM's DOORS [102] are examples of tools that offer slightly more structured ways to document requirements in the context of Model Based Systems Engineering (MBSE) and requirements management, respectively. However, MBSE tools like DOORS and SysML do not enforce formal, mathematically or logically precise requirements specifications that can be automatically analyzed.

Another class of tools conduct some form of basic analysis on natural language requirements. QVScribe [103, 104, 105, 106] conducts entry level analysis on natural language requirements to find things like vague words (i.e. using "it" rather than referring to a specific component), consistency in units and terms, negative imperatives ("shall not" rather than "shall"), and universal quantifiers (such as "no"). It is worth noting that the requirements community is somewhat divided over the use of negative imperatives and universal quantifiers and removing ambiguity in requirements is an area of active research [107]. There has also been development in basic analysis techniques for DOORS and SysML such as checking that natural language requirements conform to requirements templates in DOORS [108], automated analysis of requirements change impact in SysML [109], and verification of formalized SysML requirements on activity diagrams formalized as hierarchical colored petri nets [110, 111, 112]. Another similar tool is Jama [113], which helps organize and track changes in requirements, risk and test cases. Program management tools like Atlassian's JIRA and Confluence tools can also be used to manage requirements, design changes, and test plans [114].

Going beyond searching for presence of specific words or phrases to formal specification, natural language processing tools like the Automatic Requirements Specification Extraction from Natural Language (ARSENAL) [115, 116] find formal specifications in natural language requirements. One of the challenges of extracting formal requirements from natural language requirements is that the natural language requirements are often not complete enough to find all the required information to generate a mathematical or logical statement.

Recognizing the need for complete specifications, other approaches focus on the creation of

structured natural language requirements with various levels of analysis support. NASA's Formal Requirements Elicitation Tool (FRET) tool [117] uses a graphical user interface to guide subject matter experts in specifying requirements that can be translated to temporal logic. While FRET does not currently support analysis, it creates an ontology that may be especially useful in knowledge graph-related artificial intelligence and machine learning [118]. In addition, FRET's requirements can be used to formally analyze compliance of Simulink models using CocoSim [117, 119]. Formal Simulink model analysis tools in the same class as CoCoSim include QVTrace [120, 121], Simulink Design Verifier [122], and others [123].

A final class of tools offers formal analysis of system requirements and design specifications. The EARS-CTRL tool analyzes requirements specified using Easy Approach to Requirements Syntax (EARS), a structured natural language [124, 125] and offers basic realizability checking. The two most capable constrained, near-natural language requirements specification and formal analysis tools are the Specification and Analysis of Requirements (SpeAR) tool [126] and the Analysis of Semantic Specifications and Efficient generation of Requirements-based Tests (ASSERT) tool [127, 128, 129]. SpeAR is an open source tool that allows users to specify their requirements in

Table 2.3: Comparison of SpeAR and ASSERT Formal Requirements Specification and Analysis Tools

	SpeAR	ASSERT
Specification Format	ptLTL	set theory and subset of first order logic
Specification Language	SpeAR	SADL Requirements Language
Translated Output (1)	Lustre FSM, ptLTL requirements	Prolog first order logic if-then
Translated Output (2)	JKIND k-induction model checker	ACL2s Lisp representation
Analysis Engine	Z3, CVC4, SMTInpterpol, or Yices 2 SMT Solver	ACL2s Theorem Prover
Availability	Open Source	Proprietary

a constrained natural language with the formal semantics of ptLTL. SpeAR's realizability analysis algorithms are based on the work of [98]. SpeAR conducts logical entailment analysis in three steps. First, SpeAR outputs a Lustre [130] model with design specifications in the form of a finite state machine and requirements in ptLTLT. Next JKIND [131], a k-induction [96] model checker [13] is used to explore the state space of the requirements. Third, JKIND calls a Satisfiability Modulo Theory (SMT) solver [132] to prove or falsify requirements in parallel. SpeAR supports use

of the Z3 [133], CVC4 [134], SMTInterpol [135], and Yices 2 [136] SMT solvers. If the design specification violates a requirement, SpeAR creates a counterexample showing a trace of the violation. Traceability analysis in SpeAR is based on the techniques in based on the techniques in [99]. In ASSERT, requirements are captured using a combination of set theory and a subset of first order logic (based on the Web Ontology Language (OWL)) [137] using an extension of the Semantic Application Design Language (SADL) [138], called the SADL Requirements Language (SRL) [128]. Then requirements are translated into if-then first order logic in Prolog [139]. From there, the requirements are translated to Lisp [140] so that they may be analyzed for contingency, conflict, completeness, independence and surjectivity [141] using the ACL2s Theorem Prover [142].

In this dissertation, QVScribe and SpeAR are used in the analysis. QVScribe is used to analyze an initial set of natural language requirements. SpeAR is used to formally document the requirements in ptLTL and then analyze them for realizability, logical entailment, logical consistency, and traceability.

2.2.10 Metamodeling and Metalevel Analysis

Metamodels are abstract models of complex systems that give an all-inclusive picture of a system or process, and enable *metalevel* analysis, which is at a higher and more abstract level than a system is typically analyzed. Metamodeling is proposed as a method to conduct system-level analysis under the *state-space explosion problem*. One way to deal with state-space explosion is through abstraction (metamodeling), where a system analyst might:

- represent the graph with *atomic propositions*,
- represent sets of states and sets of transitions rather than single states and single transitions,
- conduct proofs at the highest level of abstraction reasonable, and/or
- apply lessons from *abstract interpretation*, which focuses on sound approximations of semantics in computer programs and uses *static analysis* to automatically extract possible executions of computer program without actually executing the program.

In this dissertation, metamodels and metalevel analysis are used in the formal specification and analysis of design specifications, requirements, and products of the hazard analysis.

2.3 Hierarchy of Decision and Control Functions

When discussing decision and control functions, there are a variety of forms depending on where the control is in a larger hierarchy as described in Figures 2.6-2.7. At the lowest level, often called *inner loop control*, a controller sends commands to actuators based on sensed state information. In practice, inner loop control is often done by a lower level autopilot function. At the next level, sometimes called *outer loop control*, a controller is sending higher level commands like heading, speed and altitude that are translated by the inner loop control into actuator commands. *Navigation* is sometimes used to describe monitoring of a trajectory and determining the control required to stay on the desired track. When there is no feedback in the controller that indicates the current state, it is referred to as *open loop control*. *Guidance* is the determination of a desired trajectory at a higher level that may be communicated as waypoints or path primitives.

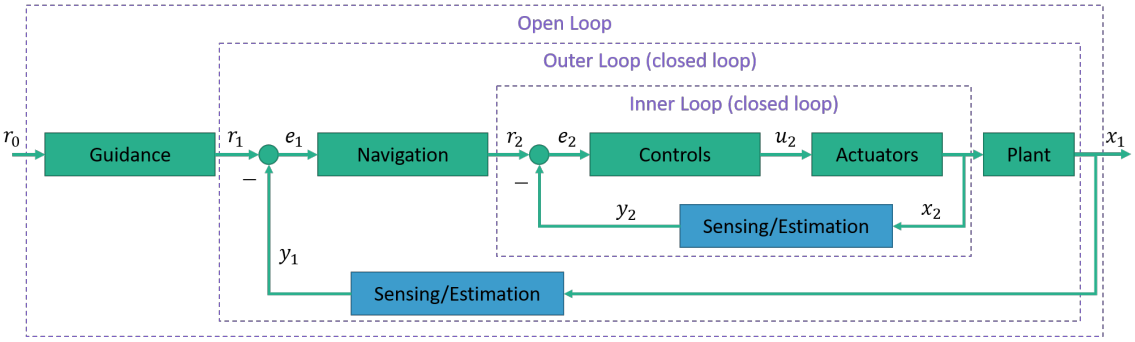


Figure 2.6: Inner, Outer, and Open Loop Control

In this dissertation, RTA is conducted at the guidance and navigation levels, rather than at the inner loop control levels. RTA is used to ensure that higher level system states are acceptable.

2.4 Run Time Assurance

Current certification approaches rely on offline verification methods to evaluate every possible state space scenario, which is an impossible task for non-deterministic, adaptive, or near-infinite state algorithms. To supplement and complement offline verification methods, Runtime Verification (RV) methods monitor and analyze the behavior of software and hardware systems online [143]. Run Time Assurance (RTA) systems go one step beyond RV methods by acting on the results of RV

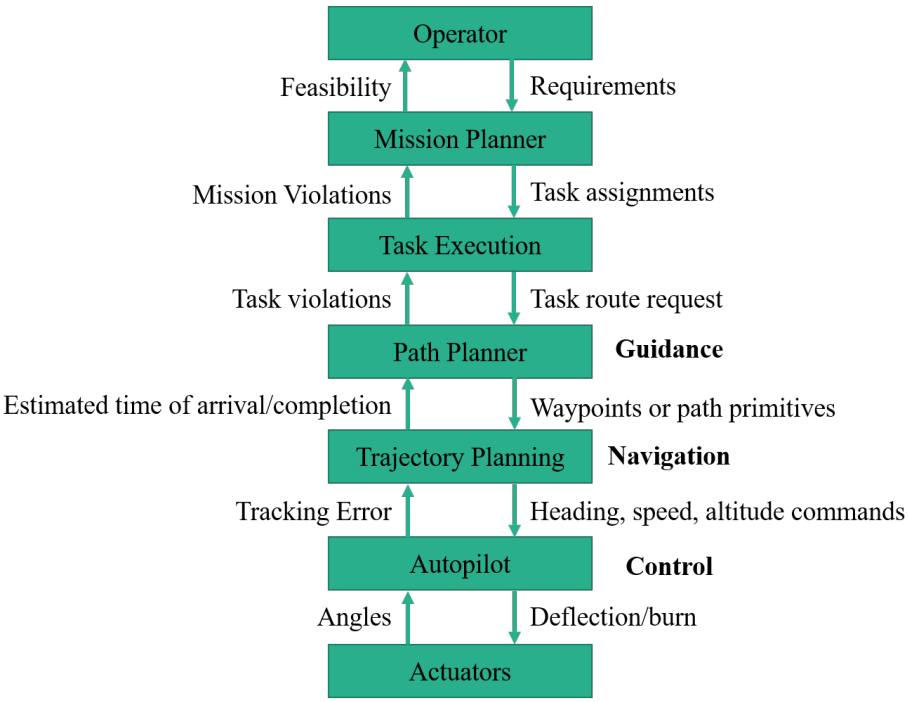


Figure 2.7: Hierarchy of Controls Functions

online. The two core functions of RTA systems are to:

- monitor and detect potential safety boundary violations
- invoke a recovery or switching mechanism to ensure system safety.

Online verification methods like RV and RTA present a path to enhance system performance with non-deterministic, non-stochastic, adaptive, near infinite state components that promise greater performance but cannot be verified with existing methods. Assurance of a system within an RTA approach is predicated on offline verification of the bounding, monitoring, and recovery mechanisms. The backup controller goes by many names in the literature, including the *backup controller*, *safety controller*, *high-assurance controller*, *recovery controller*, *safety remediation controller*, and *reversionary controller*, which will be used interchangeably in this dissertation.

Run time assurance has evolved over the last several decades from research straddling the aerospace and computer domains. There are several approaches to run time assurance including barrier functions that gradually influence the control to prevent safety violations [144, 145] and hybrid systems that switch between discrete modes with distinct control approaches governing con-

tinuous dynamics[146]. However, this dissertation builds on an architectural approach to run time assurance, known as the *simplex architecture*, which arose out of research in the computer science and cyber physical systems.

2.4.1 Runtime Verification

While this research focuses on RTA, RV is worth a brief introduction because significant work has been accomplished using formal methods to verify online monitors and alerting systems. RV is also sometimes called runtime monitoring, trace analysis, and dynamic analysis [147]. Runtime verification can be used for testing, verification, and debugging before deployment and to ensure reliability, safety, robustness and security after deployment. RV reasons about *explicit properties* describing system behavior in a specification language, as well as *implicit properties* such as deadlock avoidance, memory safety, and bounds checking covered by specific RV algorithms and not necessarily in the specification [147]. Reasoning about implicit properties is beyond the scope of this research, so the rest of this section focuses on RV of explicit properties.

Explicit properties describe the system *behavior*, which is how the system changes over time through updates to internal *state* or some *actions* taken to interact with the environment [147]. Specifically, RV gives precise information about a single execution *trace* of a system at runtime [148]. A *trace* is a behavior abstraction for a single run of a system as a finite sequence of events [148]. An *event* is any observation about the system, and often an abstraction such as a light turning on or a temperature limit is reached [147]. A *property* of a system is a set of traces while a *specification* is a textual object that describes and denotes a set of traces. One property may have many specifications [147].

RV can be used on software, hardware, cyber-physical systems, sensor networks, or any other application where system behavior is observed. RV is particularly attractive in aerospace applications because aerospace systems are safety critical with reliability requirements on the order of the 10^{-9} catastrophic fault rate for civil aircraft avionics [149], and it has been shown that testing alone cannot verify systems to that level of criticality [9].

Aerospace systems are distributed hard real-time systems and feature unique RV challenges [150]. In distributed systems, computations take place in different locations or nodes connected by communication channels, or directed edges in a graph. For example, subsystems may conduct

computations at lower levels that impact the global system behavior. Hard real time systems process information as it is received in small fractions of a second, and these temporal constraints make them difficult to test and debug. Significant progress has been made developing an RV framework for distributed hard real-time systems called Copilot [151]. Copilot is a monitor programming domain specific language embedded in the Haskell functional programming language. The benefit of using a Haskell-based language is that it is inherently type-safe, Turing-complete, and keeps the DSL small, along with other benefits. Verification of monitors in Copilot is accomplished using k-induction model checking [152]. Runtime verification of distributed hard real-time systems is beyond the scope of this research.

One of the closest sets of RV research to this dissertation is the 2015 dissertation of Aaron Kane [153]. Kane's research focuses on RV for embedded systems, while this research focuses on RTA for abstractions of a specific class of systems (aerospace collision avoidance systems). Both Kane's research and this research develop an architecture to accomplish the RV or RTA tasks, respectively. Both dissertations use formally specified monitors, however Kane's dissertation focuses more on the monitoring algorithm proof, while this research focuses on system-level proofs. Both dissertations identify requirements patterns for RV of the embedded system or RTA of the abstract system description. Finally, both dissertations show feasibility of the approach on a specific system.

2.4.2 The Simplex Architecture

The development of control architectures resembling run time assurance can be traced back work in the mid-1990s at Carnegie Mellon University. Initially these systems were envisioned to test an upgraded controller before completely replacing the legacy controller. This is like the popular software testing concept of sandboxing [25], where new untested and untrusted code is isolated from critical resources. The concept of using redundant control software running separately, but in parallel with heterogeneous designs for high-performance or high reliability, was introduced in [26]. Building on this concept, the necessary technologies for an architectural solution to upgrade software while a system was running were described in [154]. This architectural solution was matured to become the *simplex architecture*, introduced a few months later in [27], and included many of the core components still used today. As shown in Fig. 2.8, the 1996 version of the simplex architecture was intended as a way to add incremental control improvements and included a baseline

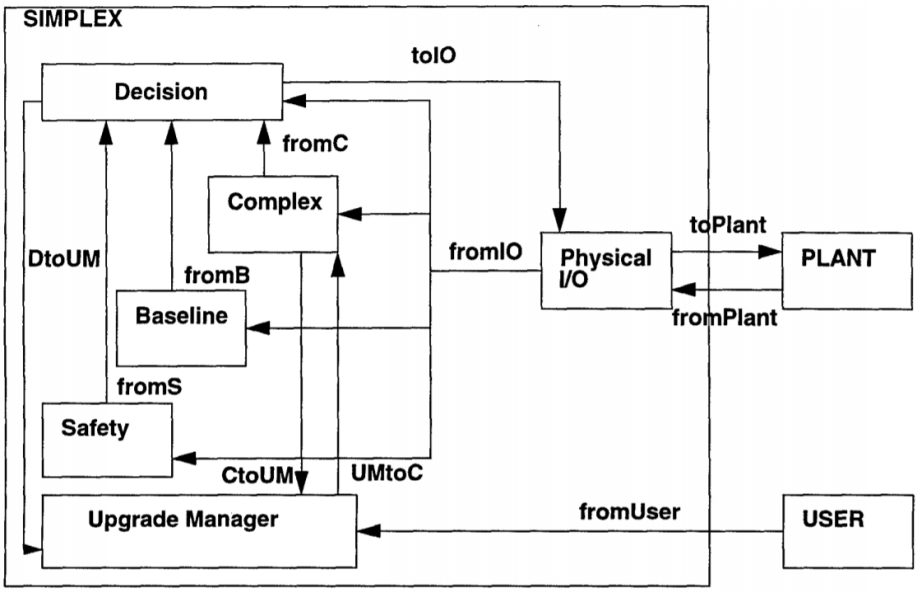


Figure 2.8: Simplex Architecture presented in [27]

controller (i.e. the legacy controller design), a complex controller (i.e. new, higher performance design), and a safety controller (which returns the system to a safe state if it is being pushed outside the safety region). The decision module determined which of the control outputs were applied to the plant. This model also included a limited form of user interaction, where the user could change the complex controller through an upgrade manager function. Additional refinements of this concept were published in [28, 29], including a refined image of the simplex architecture depicted in Fig. 2.9. The work by [29] focused on the use of RTA to detect and tolerate timing and semantic faults. Timing faults occur when a controller did not complete computation of its control command by the end of a sampling period and could be the result of improper selection of the sampling period or other coding faults such as divide by zero or infinite loops. Semantic faults occur when a controller generates a command that violates the control specification for the application-domain specific physical system (i.e. when the state of the system is approaching an unsafe region of the state space). This work also formally specified switching conditions, admissible states, and admissible controls in terms of dynamics and control theory definitions. The simplex architecture was evolved again to utilize formal methods verification of a high-performance controller, a high-assurance controller, and decision logic with application to an aerospace example [30]. This work also highlighted one of the disadvantages of the simplex architecture:the total system performance

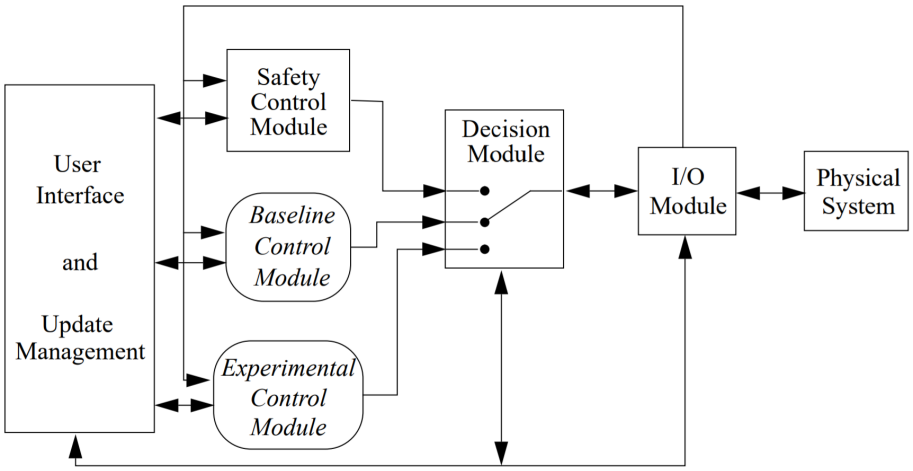


Figure 2.9: Simplex Architecture presented in [29]

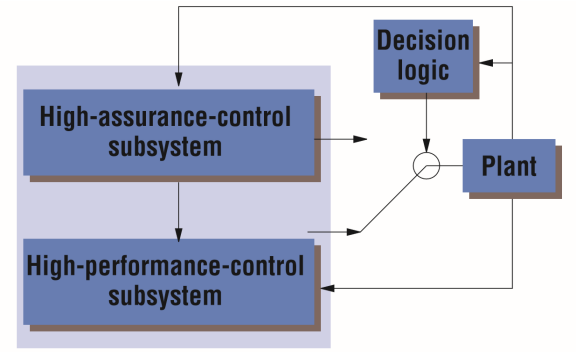


Figure 2.10: Simplex Architecture using a formally verified high-performance controller inside the operating envelope of a high-assurance controller that could be switched to as a safety backup[30]

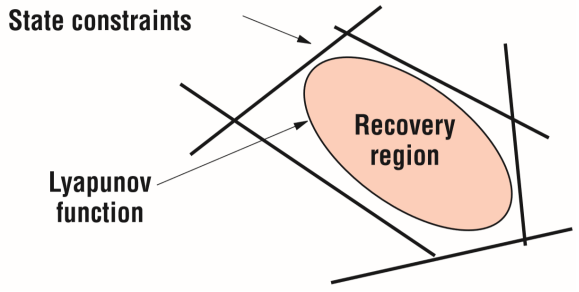


Figure 2.11: Lyapunov function ellipsoidal boundary inside operational constraint polytope [30]

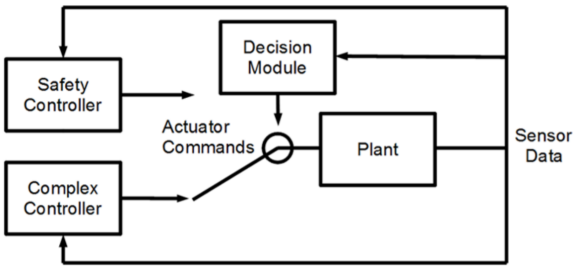


Figure 2.12: Simplex Architecture with a physical plant, a verified safety controller, a verified decision module, and an unverified complex controller [32].

is limited by the operational envelope of the secondary (high assurance, backup) controller. As an example, the work cited the triple redundant Boeing 777 flight control system featuring two different controllers: a normal controller optimized for the 777 and a secondary controller based on the 747 control laws that had 25 years of heritage. One disadvantage of the design was that the normal controller (the 777 controller) was restricted to flight inside the secondary controller's (747) smaller flight envelope. In this version of the simplex architecture, the constraints of the system (including physical, environmental, safety, and operational requirements) were represented as a *polytope*, an n-dimensional figure with hyperplane faces, where states inside the polytope were admissible. The recovery region was a smaller space inside the admissible states defined by a Lyapunov function (geometrically represented as an n-dimensional ellipsoid).

While the work in [30] suggested that all components be verified, nearly a decade later work in [31, 32] returned to looking at the simplex architecture as an enabler of unverified code. This approach predicated system safety on offline verification of the decision module and safety controller for cases when verification of a complex controller is cost prohibitive or impossible. In [33], the decision module switching was computed using reach set computations for a restricted class of hybrid automata based on [34, 35, 36], and then expanded to address some of those restrictions in [32].

2.4.3 Run Time Assurance for Aerospace

While some interest in run time assurance for aerospace was present as early as Sha's 777 flight control system example in 2001 [30], interest grew tremendously in the 2010s. In [155] a run time verification and validation wrapper is introduced that monitors high-risk algorithms for unacceptable

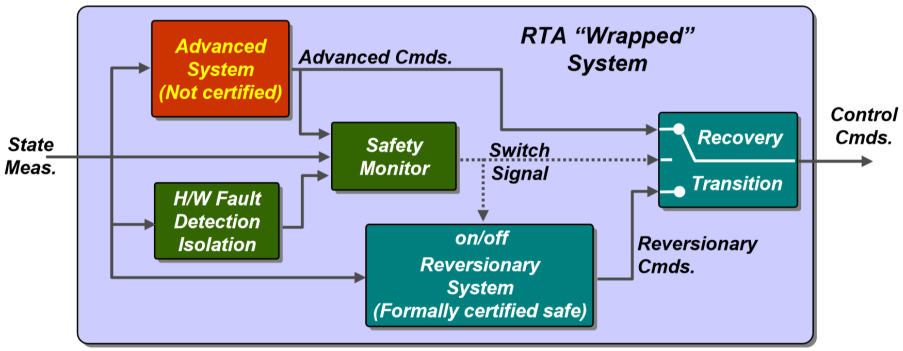


Figure 2.13: Run time assurance wrapper concept including fault detection and isolation for advanced controller [37].

errors, reverts to a simple backup, and is demonstrated on a multiple control layers including inner loop control. A fault detection and isolation component was also introduced in [155] to detect whether unacceptable errors were occurring because of control surface failures, sensor failures, or an error in the software. Work in [37] built on the concept of a run time assurance wrapper, shown in 2.13, for aerospace systems echoing the importance of developing a fault detection and isolation (FDI) capability concurrently with the RTA. However, the focus of the FDI in was on detecting faults in the advanced controller, not other system wide faults. This paper echoed [30] by discussing the importance of defining the reversionary safety envelope (RSE) of the reversionary controller, state that the aircraft is only guaranteed to be safe under the reversionary controller if it is within the RSE. Work in [7] demonstrate and evaluate one of the primary motivations of RTA systems: enabling safe integration of adaptive control on aircraft that cannot be verified using traditional verification. In [7], an RTA system is developed to enable adaptive control on a quadcopter. A baseline controller using proportional-integral-derivative (PID) linear control schemes that can be verified using traditional techniques is employed as a backup controller. Similar RTA concepts were employed on highly adaptive flight control systems [156], propulsion systems [157] including turbofan engine control [158], and geofencing UAVs [159].

Within the last five years progress has been made towards developing certification criteria for run time assurance [160] resulting in the publication of a recent ASTM standard for the use of run time assurance in unmanned aircraft [11].

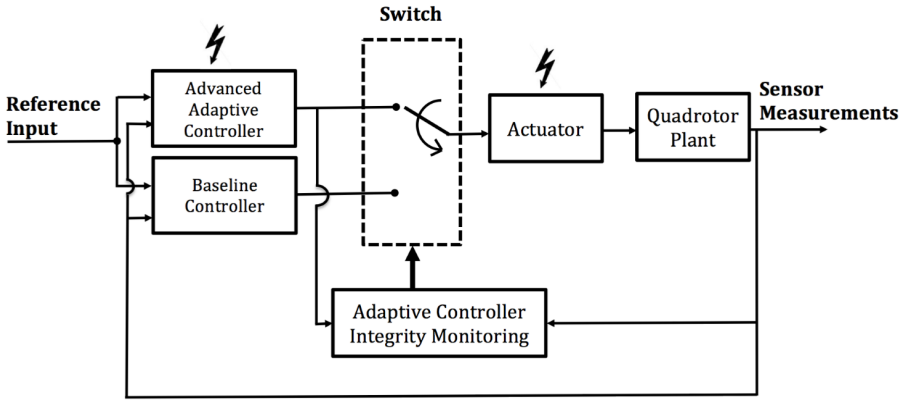


Figure 2.14: Run time assurance control architecture to enable nonlinear adaptive control of a quadrotor with a backup PID baseline controller [7].

2.4.4 Multi-Monitor Run Time Assurance

While much of RTA research focuses on the use of a single simple backup controller, multi-monitor run-time assurance (MMRTA) architectural approaches [39] have multiple backup controllers depending on the type of boundary violation prediction. These approaches emphasize a modular framework with minimal to no interfaces between backup controllers, enabling easier verification. In initial research backup controllers have been created for ground collision avoidance and geofencing, and envisioned future functions include midair collision avoidance, separation assurance, and weather avoidance. Each backup controller operates independently and an executive flight module deconflicts between the systems, interfaces with the autopilot to conduct the maneuver, and uses a risk-based decision logic "moral compass" to prioritize backup approaches. In practice it was found that the structure of both the ground collision avoidance and geofence backup controllers was almost entirely the same, differing by just 30 lines of code [39]. One consequence of this approach is that in responding in priority order, it may be possible for activation of one recovery function to violate a different boundary. For instance, if the system decides that a ground collision is a higher risk than a geofence boundary violation, the ground collision avoidance recovery maneuver may violate the geofence boundary. One way to deal with this possibility is to make the boundary conditions more conservative at the cost of restricted performance. Rather than treating each backup system independently, an alternative approach is integrating multiple backup systems together so that each is aware of the strategy of the other, such as integration of ground and midair collision avoidance

in the integrated collision avoidance system [161, 162]. The complexities and challenges of this approach are discussed further in Section 6.2.1.

2.4.5 Formal Methods and Run Time Assurance

Formal methods has been used to define and verify monitors, decision modules, and backup controllers in previous work. In [163], a runtime monitor is used to enforce behavior based on a formally defined concept of operations for an unmanned system. By contrast, this dissertation uses requirements informed by previous work, standards, and hazard analysis rather than concepts of operation. In addition, the high-level RTA architecture in this dissertation is different. In [31], model checking was used to verify correctness of discrete system switching logic, but the high-level architecture and types of properties proven were different than that presented in this dissertation. The use of formal methods based a limited set of formal requirements for runtime monitoring of system behavior is presented in [164, 165]. Like the research in this dissertation, this work clearly separated implementation (or design) specifications from high-level requirements specifications. However, run time assurance was constrained to what they described as a Monitoring and Checking (MaC) framework, to assure correctness of execution at run time. The MaC framework monitored system performance but did not react to requirements violations with an automated response. Formal verification of linear temporal logic specifications has been applied to PID attitude control of spacecraft [166], a simplex run time assurance system for spacecraft attitude [167, 168, 169], switching logic for automatic maneuvering [170], and an LTL specification monitor automaton [171].

2.5 Requirements Elicitation from Hazards Analysis using Systems Theoretic Accident Model and Processes and Systems Theoretic Process Analysis

This section provides background information on the STAMP and STPA methods used to conduct the hazard analysis. A hazard assessment using STAMP modeling and STPA was selected because it can be applied early in a design process, incorporates human interaction, and allows for analysis of hazards in the presence of feedback between systems, while many hazard analysis methods are linear and unidirectional. While STAMP and STPA can provide indication of unsafe actions that could be translated to safety requirements, they are qualitative, not quantitative and cannot provide

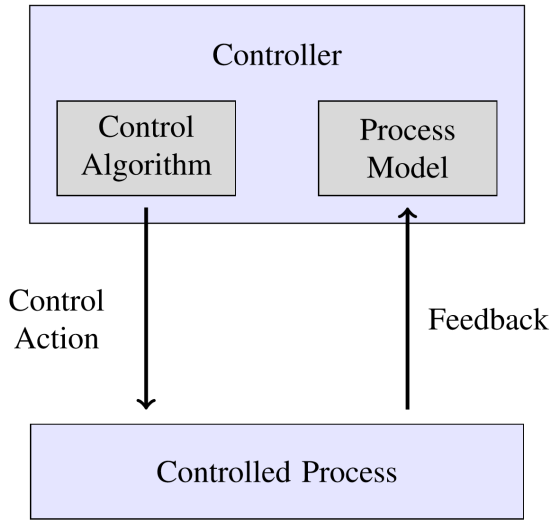


Figure 2.15: Components of a STAMP functional control block diagram.

a risk level or percent reliability. STAMP and STPA can be done iteratively from the earliest stage of system design until completion, with refinement as design decisions are made.

2.5.1 Hazards, Faults and Failures

This section provides context for the hazard assessment with definitions of failures, faults, and fault tolerance. A *failure* is the inability of a system, subsystem, component, or part to perform its required function within specified limits [43, 8]. A *fault* is the physical or logical cause that explains a failure, [43] or a manifestation of an error [8]. Fault tolerance is the ability of the system to sustain acceptable performance and safety in the event of one or more failures [43, 8]. These definitions compliment the STPA and STAMP definitions provided in the next couple sections.

2.5.2 Systems Theoretic Accident Models and Processes

STAMP is a functional control block diagram model of complex systems described in [12], and briefly summarized here. The basic structure of the diagram is shown in Fig. 2.15. Each of the blocks in the system are functional blocks rather than physical component blocks which enables multiple physical instantiations to be applied. The diagram is also hierarchical where blocks above have control over blocks below. Each functional block is either a controller or a controlled process but can be both depending on its location in the diagram and relationship to other blocks. Every

controller has a control algorithm that describes how it decides its output and a process model that describes how it uses inputs to construct a model of the world. Down arrows in the diagram are control actions while up arrows are feedback. While not explicitly stated in previous work, STAMP also facilities nesting of subsystems within larger systems, and can be expanded to explore multiple levels of a system design.

2.5.3 Systems Theoretic Process Analysis

STPA is a hazard analysis technique described in [12], and briefly summarized here. The first step is to identify *accidents*, which are undesired or unplanned events that result in some type of loss, including but not limited to loss of human life, human injury, property damage, environmental pollution, or mission loss. The second step is to identify *hazards*, defined a system state or set of conditions that in a worst-case environment would lead to an accident. Each hazard should tie directly to an accident. The third step is to identify *safety constraints*, which are constraints than can be implemented to reduce the risk level of the hazards.

Having completed the first three steps, the next step is to identify *unsafe control actions* by analyzing the STAMP diagram. Unsafe control actions are comprised of four components: a controller, control action, unsafe control action type, and context. Each control action (down arrow) in the STAMP model is evaluated for four possible unsafe control action types:

- provided (in an inappropriate context),
- not provided (in a context where it should be),
- duration (a continuous control action is provided for too long or too short a duration), and
- timing (a control action is provided to early or too late).

Requirements can be generated that constraining the design or highlighting the need for additional feedback to prevent unsafe control actions.

2.5.4 Previous Applications of STAMP and STPA in Aerospace

A number of factors including ineffective safety engineering of spacecraft software have been shown to cause spacecraft accidents [172]. While software is becoming more prevalent in aerospace, haz-

ard analysis processes outside of STAMP and STPA often fall short of identifying software hazards [173, 174]. STAMP and STPA have been used to used to analyze spacecraft [175], an aircraft rapid decompression event [176], space launch vehicles [177], safety and cyber security for integrating unmanned aircraft in the national airspace system [178], the NextGen air traffic management system [179, 180] manned-unmanned teaming of aircraft [181, 182], the takeoff phase of a complex UAV [183], and others. A combination of intent specifications [67], STAMP and STPA were applied to a spacecraft in low Earth orbit [184]. Previous applications demonstrate the ability of However, previous research has not specifically considered hazards for automated maneuvering, and a complete set of safety constraints, and requirements from the process have not been presented.

2.6 Design Elements Specific to Aerospace Systems

This dissertation is scoped to run time assurance systems, and specifically collision avoidance systems, although the concepts are anticipated to apply in other domains. Aerospace engineering is relatively unique because of several features. Aerospace systems are in a class of safety or mission critical systems like nuclear and medical domains where high assurance is expected. Aerospace systems also often feature long lifecycles and legacy computing requirements when they are in service over the course of several decades or require rugged or radiation hardened electronics that are decades behind. Aerospace systems are also often produced in smaller quantities, and sometimes only one implementation of a specific system is built. Aerospace Systems feature unique sensing needs that are common to their class of system but not necessarily common to other domains such as pitot tubes, angle of attack sensors, and star sensors. Finally, aerospace often divides the systems engineering process into a common set of subsystems: structures, propulsion, power, thermal, controls, aerodynamics (mostly aircraft but also a consideration for low orbiting spacecraft), and communication, and command and data handling. A common set of subsystems facilitates similar high-level software architectures and RTA approaches.

CHAPTER 3

CASE STUDY I: AUTOMATIC GROUND COLLISION AVOIDANCE SYSTEM

Auto GCAS is a very interesting case study for the development and acceptance of automatic control systems in aerospace because of the degree of risk if the system were to fail, a three decade development with a rich set of best practices and lessons learned, and its uniqueness as a deployed system that takes control from the pilot in life or death situations [16]. Several factors of the Auto GCAS system design as well as the development and test process contributed to its success. Applying design principles such as simplicity of the recovery response, nuisance free operations, the ordering of high-level requirements, failure monitoring, pilot-selectable risk tolerance, operator transparency, modular design, and high reliability are instrumental to the success of future autonomous systems. Including operators, design and test engineers, managers, and certification authorities throughout the design process with a deep understanding of the need for the automated system are critical components of the autonomous system design process.

3.1 Key Design Factors Contributing to the Successful Development of the Automatic Ground Collision Avoidance System

3.1.1 Simplicity of the Recovery Response

The first key to the successful implementation and certification of automatic ground and air collision avoidance systems was the simplicity of the automatic recovery maneuver, which facilitated comprehensive evaluation and improved human trust in the system. Simplicity of the automatic recovery response is key because it gives a finite set of responses that are acceptable and can be pre-verified for use. Pilots can evaluate whether specific maneuvers were too aggressive or not responsive enough. In automation trust research, it has been found that simplifying algorithms so that they are understandable to the end user is an important factor in calibrating user trust [185]. For Auto GCAS, familiarity of the maneuver and its consistency with pilot training and behavior resulted in strong positive perceptions of the system [17]. Rather than optimizing a trajectory for each specific collision avoidance scenario, a predefined set of verified maneuvers was created. In F-16

Auto GCAS, a single roll to wings level and 5g pull maneuver is conducted [186]. In the Automatic Air Collision Avoidance System (Auto ACAS) system, nine [187] different maneuvers are considered. In patented [188] and experimental Auto GCAS designs for small UAVs [38], lesser capability aircraft [189], and cargo-class aircraft [190], three or more possible maneuvers may be considered. A finite set of pre-verified maneuvers is critical to the development of autonomous backup control systems.

3.1.2 Nuisance Free Operations

In studies of pilot trust in Auto GCAS, *nuisance avoidance* was the most prominent condition for developing trust in Auto GCAS [16]. One pilot in the study indicated that one false fly-up would likely cause pilots to turn the system off and lose the protection it provided. Interviews with pilots indicated that the high occurrence of nuisance warnings caused pilots to "tune out" or turn off previous ground collision warning systems that required a manual response [191, 192]. This feedback is an example of cases found in previous research that frequent false-alarm rates lead pilots to deactivate critical alarm systems [193].

Shortcomings of Previous Ground Collision Warning Systems

Previous collision warning systems were prone to missed detections and false alarms that informed development of Auto GCAS. Several ground collision detection, warning, and advisory systems were implemented on aircraft. Line in the sky and Digital Terrain Elevation Database (DTED)-based collision warning systems are described in this section, with discussion of their shortcomings.

Early altimeter and radar-based solutions tended towards overly conservative or partial coverage. An early line in the sky system warned the pilots if the barometric altimeter indicated the aircraft was below a pilot-set mean sea level (MSL) altitude [194, 190], and an altitude low (ALOW) system was implemented that warned the pilot when the aircraft radar altitude went below a pilot-set above ground level altitude floor [194, 190]. Line in the sky systems were overly conservative, potentially providing warnings and advisories frequently and far in advance of the need to maneuver. Improving on these line in the sky systems, the Ground Avoidance Advisory Function (GAAF) used radar to predict collisions while considering altitude loss during a maneuver, reaction time, roll response time, weight, drag, and potential energy [194, 195, 190]. A similar commercial Ground Proximity

Warning System (GPWS) warned for excessive descent rate, excessive terrain closure rate, altitude loss after takeoff, unsafe terrain clearance, and excessive deviation below glideslope [196, 197, 198]. While less nuisance-prone than line in the sky solutions, radar-based solutions like GAAF and GPWS primarily considered terrain below the aircraft and lacked the ability to detect collisions with rising terrain ahead of the aircraft, leaving the aircraft vulnerable. A solution that didn't depend on radar or barometric altitude was needed.

The need to reduce nuisance activations while also looking at terrain ahead of the aircraft prompted development of systems that used DTED and GPS to predict collisions. The Enhanced GPWS (EGPSW) and Terrain Awareness Warning Systems (TAWS) used terrain, obstacles (such as towers), and airport runway databases as well as aircraft state to provide collision cautions or warnings about to 60 seconds ahead; however these systems did not recommend recovery maneuver actions [196, 197, 199]. A US Navy derivative of TAWS did recommend two possible recoveries: either a roll to wings level and 5-g pull Vertical Recovery Trajectory (VRT) or a 5-g pull Oblique Recovery Trajectory (ORT) from the current bank angle. The USAF equivalent to TAWS was the Predictive Ground Collision Avoidance System (PGCAS), which directed pilots to conduct a roll to wings level and 4-g pull when the aircraft was expected to dip below a pilot-set minimum terrain clearance (MTC) height [195]. While systems that use DTED and GPS to predict collisions may be able to detect collisions ahead of the aircraft and be less nuisance prone than line in the sky solutions, they still must come on early enough for an aware pilot to engage the maneuver. The primary shortcoming of warning-based systems was that conservatism was built into the design to allow for pilot reaction time. This conservatism made them prone to false alarms, which have been shown to degrade trust and acceptance of a system [200]. In addition, if the pilot is distracted, task fixated, spatially disorientated, or unconscious from gravity-induced loss of consciousness (GLOC), a warning is not enough. Auto GCAS goes one step beyond a warning system by automatically engaging an avoidance maneuver.

Manually Enabled Automatic Maneuver System

The Pilot Activated Recovery System (PARS) [201, 191, 192, 202] is a related, manually-activated, automatic response system that leverages ground collision avoidance research but is not ground aware. Pilots may activate PARS at any time to command the aircraft to return to wings level flight

as flight test aid, to mitigate spatial disorientation, or for any other reason.

Time-Based Metrics

To facilitate nuisance-free collision avoidance, Auto GCAS developed a novel approach that used time rather than distance as a metric [16]. In 1995, a study was conducted to develop a time-based metric where pilots felt an aggressive recovery should be activated to avoid a collision [191, 192]. The *time available* metric was developed where zero time available would result in an automatic maneuver that touched the terrain, and increasingly positive time available would result in maneuvers at greater altitudes above the terrain.

Table 3.1: Pilot Anxiety Rating Scale [191, 192]

Anxiety Rating	Value
Was never more than casually aware of the ground	1
Would have felt more comfortable with a recovery at a lower altitude	2
Recovery went as anticipated	3
Recovery went lower than personal comfort levels allow	4
Sensation of life-threatening conditions	5

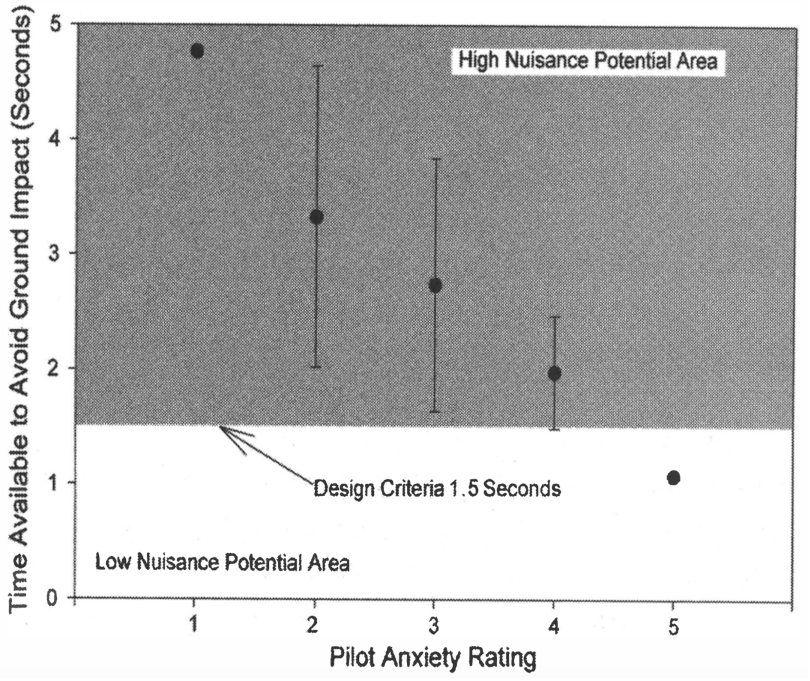


Figure 3.1: Nuisance Boundary Based on Time Available [191, 192]

To determine the line between nuisance and appropriate activation times, pilots flew towards the ground at a variety of dive angles, bank angles, airspeeds, and load factors and activated a recovery maneuver when his or her comfort threshold was met. After each run, the pilots rated the timing of the recovery initiation, their anxiety level during the maneuver, and the precision/aggressiveness of the maneuver for each run. The anxiety level descriptions are provided in Table 3.1. The results of the study are shown in Fig. 3.1. A nearly linear relationship between the average time available for a specific anxiety rating was observed, and a rapidly decreasing lower boundary for time available appeared to asymptotically approach a lower boundary. Based on this information a 1.5 second time available design criteria was identified.

3.1.3 High Level Requirements Precedence

Another key aspect of the Auto GCAS design that has made it successful over previous systems is the order of the high-level requirements. Placing the requirement to not interfere with the primary operations of the aircraft (be nuisance free) ahead of preventing collisions was critical. In order of precedence, the requirements for the automatic ground and air collision avoidance systems were to do no harm, do not interfere, and prevent collisions. These requirements were so important that they were featured prominently on the edges of the program patch, shown in Fig. 3.2. In a way, the order of the high-level requirements is somewhat counter intuitive as one might expect preventing collisions to be the top priority of a collision avoidance system. This order of requirements proved critical in pilot acceptance of the system [17]. To do no harm, the system could not put the aircraft or the pilot in additional danger. To not interfere, the system had to engage later than an aware pilot to avoid nuisance activations and ensure the pilot can complete his or her mission. Finally, when the system doesn't cause harm to the pilot or activate as a nuisance, the system will maneuver to prevent collisions. Nuisance free operation, in accordance with the "do not interfere" high level requirement proved to be one of the most important goals of the system design [191, 192, 16].

3.1.4 System Wide Integrity Management

Recognizing and accommodating failures is a critical element of Auto GCAS [187], which accomplishes this through distributed integrity monitoring methods [191]. For example, a failure of the inertial navigation system should not allow Auto GCAS to roll inverted and pull down into the

Figure 3.2: Program Patch for the Automatic Collision Avoidance Technologies Fighter Risk Reduction Program (ACAT/FRRP)

ground instead of pulling up to avoid a collision [191]. Most of the Auto GCAS algorithm resides in non-redundant mission avionics and a method was needed to mitigate single-point-of-failure issues [203]. Auto GCAS uses System Wide Integrity Management (SWIM), a set of hardware and software tests hosted in the redundant flight control system, to ensure automatic maneuver requests are safe and accurate [203]. Example tests include heartbeat checks where a stale or missing heartbeat signal from a component indicates a failure, and "reasonableness checks" that ensure incoming data are within a reasonable range of values [38]. Extensive simulation, ground testing, and in-flight testing verified that any failure state that could cause unsafe activation of an automatic maneuver was detected and inhibited operation of Auto GCAS [191]. While protection from ground collisions are lost in the presence of a failure, it is ensured that the higher priorities of do no harm, and do not interfere are respected [191].

3.1.5 Variable Risk Tolerance

One of the challenges of developing Auto GCAS is that the level of protection was sometimes competing with the need to be nuisance free. To deal with this, Auto GCAS used two pilot selectable modes: a "norm" mode which provided adequate safety buffers for most cases, and a "min" mode which minimized buffer size for nuisance-free low-level flying at the trade-off of reduced protection [191, 192]. Including an ability to modify the level of acceptable risk for different missions may prove to be an important consideration in the design of other automated flight control systems.

3.1.6 Transparency

Initially there were concerns that adding content to the pilot vehicle interface, and in particular chevrons indicating proximity to Auto GCAS activation to the head-up-display (HUD), would consume valuable display "real estate" and could give pilots a false sense of security that changed how they flew [16]. In the end it was decided that the benefit of providing system transparency to the operator outweighed the risks [17, 16]. However, future automated system designs will need to consider implications of transparency.

3.1.7 Modularity

One of the key concepts in the overall Automatic Collision Avoidance Technologies (ACAT) program, which included Auto GCAS and Auto ACAS, was the development of a modular architecture that could be applied across many platforms [38]. While the modular architecture is applicable across platforms, practical challenges prevent taking the software directly from one platform to another. For example, when the NASA Small Unmanned Aerial Vehicle (SUAV) Auto GCAS program was initiated, the original requirement was to tailor the F-16 algorithm including the actual F-16 Auto GCAS C^{++} code to the small UAV; however, the documentation available was insufficient for a third party to tailor the software for a new platform [38]. Nonetheless, the functional component architecture of Auto GCAS was transitioned across platforms even if the software was not. The modularity was further built upon in NASA's iGCAS, a ground collision avoidance warning system for general aviation [38].

Modular Architecture of Auto GCAS

The modular architecture of the F-16 Auto GCAS is presented in Fig. 3.3. The "Aircraft State" and "Navigation Solution" functions pass required aircraft state information to the Auto GCAS algorithm. The "Digital Terrain Elevation Data" functional block contains terrain elevation data from the National Geospatial-Intelligence agency including updated data from the Shuttle Radar Topography Mission in 2000 that collected data from 60° North latitude to 56° South latitude [191, 192]. The "Terrain Map Scanning" function scans the DTED using a variety of scan patterns based on the aircraft state and creates a two-dimensional profile of the terrain ahead of the aircraft. The "Aircraft

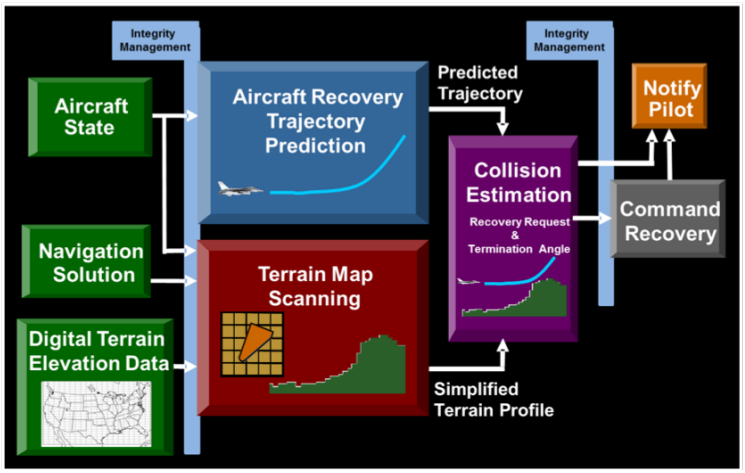


Figure 3.3: Auto GCAS Modular System Architecture [191, 192, 203, 186]

Recovery Trajectory Prediction" function uses aircraft state information to estimate an aircraft trajectory of the single roll to wings level and 5-g pull automatic recovery maneuver. The "Collision Estimation" function compares the trajectory prediction against the two-dimensional terrain profiles with a safety buffer. When the trajectory prediction "touches" the terrain, the "Command Recovery" function executes the automatic recovery maneuver and the "Notify Pilot" function informs the pilot when the maneuver initiates and terminates. The "Integrity Management" function is distributed throughout the Auto GCAS interfaces in the form of System Wide Integrity Management (SWIM) as discussed in Section 3.1.4 [203, 191].

Modular Architecture of NASA SUAV Auto GCAS

To provide insight into how the F-16 Auto GCAS architecture may be adapted to other platforms, the NASA SUAS Auto GCAS is included here as an example. Though slight differences in implementation exist, the basic functions are the same across platforms. The modular architecture of the NASA SUAS Auto GCAS in [38] Fig. 3.4 is visibly and functionally very similar to the F-16 Auto GCAS architecture in 3.3. The "Sense own-state" function provides the required aircraft state information to Auto GCAS. The "Sense terrain" function provides a map of local digital terrain elevation data to Auto GCAS. The "Predict avoidance trajectories" function uses aircraft state information to create trajectory predictions for each of the three pre-defined avoidance trajectories. While F-16 Auto GCAS predicts one trajectory, SUAV predicts three. The "Identify collision threats" function

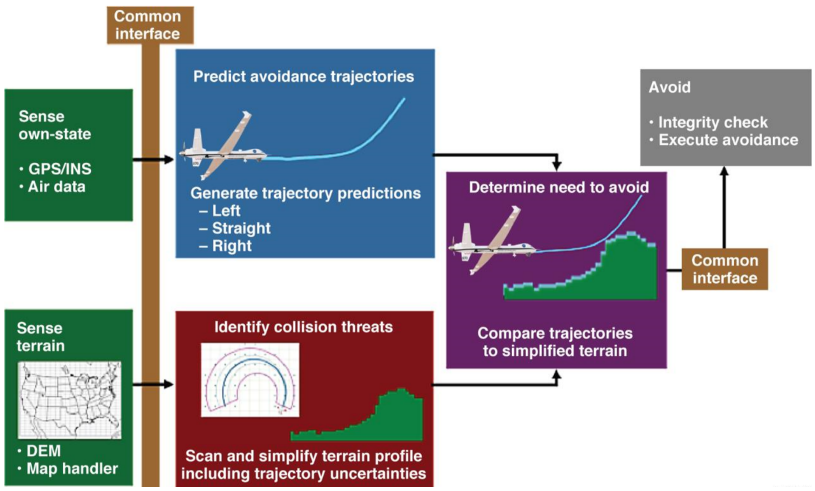


Figure 3.4: NASA SUAV Auto GCAS Modular System Architecture [38]

scans local digital terrain elevation data to generate a two-dimensional terrain profile ahead of the aircraft. The area scanned differs between the SUAS and F-16 solutions. The "Determine need to avoid" function compares the three trajectory predictions against the two-dimensional terrain profiles with a safety buffer (in blue). When the last of the three trajectory predictions intersects the terrain, a collision avoidance maneuver following that trajectory is commanded and executed by the "Avoid" function. The "Common interface" function provides similar functionality to the System Wide "Integrity management" (SWIM) functions of the F-16 Auto GCAS - it checks for failure in the Auto GCAS algorithm or data used by Auto GCAS.

It is worth noting that the modularity of the design was so effective, that later instantiations were adapted from a GCAS function to a GeoFence monitor that ensured a UAV stayed within a bounded area on a map by modifying only 30 lines of Auto GCAS Code [39].

Modular Architecture of Auto ACAS

To provide insight into how the architecture of the F-16 Auto GCAS architecture may be adapted for a different function (midair collision avoidance rather than ground collision avoidance), the F-16 Auto ACAS architecture is included here as an example. Just as with the Auto GCAS solution, the Auto ACAS "Aircraft State" and "Navigation Solution" function blocks pass required aircraft state information to the Auto ACAS algorithm [204]. Instead of DTED, locations of other aircraft are captured and sent to the Auto ACAS Algorithm via radar with the "Radar Target Location" and

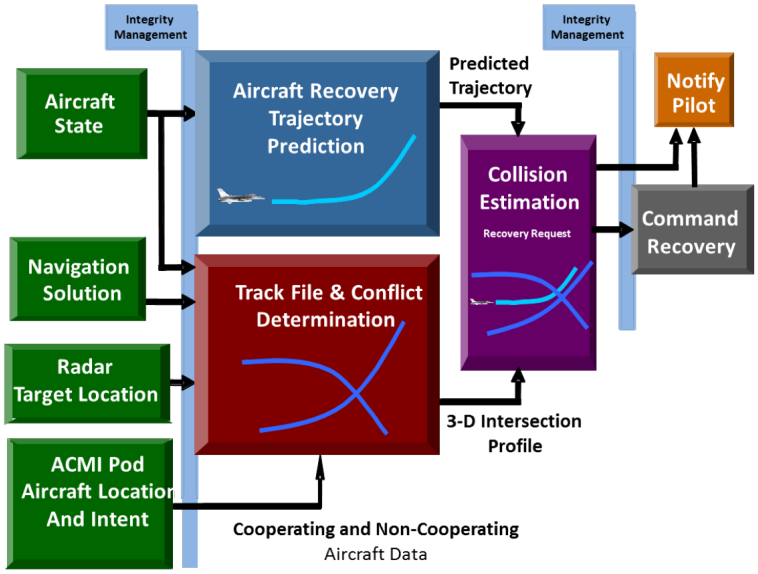


Figure 3.5: Auto ACAS Modular Algorithm Architecture [187]

via datalink with the "[Air Combat Maneuvering Instrumentation] (ACMI) Pod Aircraft Location and Intent" functional blocks [187]. The "Track File & Conflict Determination" functional block includes a "Track Manager" that determines the best estimate of other aircraft locations from multiple input sources and predicts their trajectories with appropriate uncertainty buffers corresponding to the fidelity of their source, a "Threat Isolation" function that prioritizes highest collision threats [187], and a "Formation Logic" function that prevents activation of an automatic maneuver when the aircraft is in formation flight described by a formation deactivation boundary [187]. The "Aircraft Recovery Trajectory Prediction" functional block estimates the future path of the aircraft over 4.5 seconds of a hypothetical automated recovery maneuver and is partitioned into generic and platform specific sections to maximize reuse of the software for integration on other platforms [187]. The "Collision Estimation" function includes a "Maneuver Selection and Coordination" function as well as a "Maneuver Activation and Control" function. The selection function chooses the three best of nine possible recovery maneuver options based on pre-selection logic including "rules of the road," pilot training and preferences, collision geometry, energy considerations, and maneuver effectiveness [187]. For example, if two aircraft are approaching head-on, "rules of the road" will pre-select options where both aircraft turn to the right and not consider options that maneuver to the left [187]. The activation and control block commands an automatic recovery maneuver when a collision is

determined to be imminent and terminates the maneuver when separation criteria or a max duration is met. Like Auto GCAS, Auto ACAS notifies the pilot when maneuvers are commanded and terminated. SWIM plays an important role in Auto ACAS, just as it does in Auto GCAS. In Auto ACAS, the ownship aircraft as well as external aircraft are monitored for faults. In addition to tests in redundant flight control, Auto ACAS performs several self-tests including freshness and validity of inputs, health of supporting subsystems, heartbeat checks, and mathematical operations checks [204]. For other aircraft, Auto ACAS monitor's its confidence in the data provided, especially if it is from a nonredundant source, or subject to frequent dropouts [204].

3.1.8 Reliability

Avoidance of false alarms and clear understanding of reliability were key factors in the trust of the Auto GCAS system. People tend to initially assume machines are perfect and their trust in them rapidly deteriorates following system errors [205]. As human relationships with automated systems mature, dependability and predictably become the primary basis of trust in the system [206]. Important factors to increasing trust in automation include past performance (pedigree/heritage), simplified and understandable performance, system intent, and explaining and demonstrating reliability [185]. Knowledge of Auto GCAS's 98% reliability increased test pilot trust of the system [17].

3.1.9 Inclusion of Pilots, Engineers, Managers, and Certification Authorities Throughout the Design Process

A study of Auto GCAS trust found that mutual understanding and respect was critical between pilots, engineers, and program managers. These personnel featured different accountability, vulnerability, and expectations in the development of Auto GCAS [16]. Pilots as the end users were accountable for successful missions, were vulnerable to loss of life and interference of the system with their mission, and expected Auto GCAS to save their life when needed. Engineers were accountable for designing the system to save pilot lives without interfering with the pilot's ability to perform their mission, were vulnerable to backlash from managers and pilots if the system wasn't properly designed, and expected the system to be designed and executed properly. Managers were accountable for accomplishing larger military goals for the system, were vulnerable to loss of re-

sources to complete system design and evaluation, and expected the system to reduce mishaps. The study [16] found that understanding of these different accountabilities, vulnerabilities, and expectations resulted in human-to-human trust that led to greater trust in Auto GCAS.

In addition to engaging with system stakeholders, early engagement with certification authorities during system design is important to smooth the certification process that allows fielding of systems. Military and civilian certification standards are updated regularly, and new flight certificates may be tested to new standards may require additional design and development costs. The Auto GCAS team engaged the USAF flight safety certification group during system requirements development to design a tailored certification program for the Auto GCAS solution for F-16s with analog flight control computers years before certification would be requested [203].

3.1.10 Personal Connection to the System Need

An important influence of the expectations and acceptance of Auto GCAS was found to be pilot's personal connections through the loss of fellow pilots to ground collisions [17]. A controlled flight into terrain (CFIT) mishap occurs when an airworthy aircraft under the control of a pilot with inadequate awareness flies into terrain, water, or obstacles [207]. Prior to Auto GCAS, CFIT was the top cause of USAF fighter pilot fatalities, and second leading cause of fighter mishaps [208]. Between 2000 and 2013, the DoD lost 23 pilots and 29 F-16, F/A-18, or F-22 aircraft to CFIT [190]. At the time of this writing, Auto GCAS has been credited with saving 8 aircraft and 9 lives on F-16s with digital flight control computers. Personal experience with loss in the absence of an automated system will be an important factor in motivating the development and transition of future automatic backup systems.

3.2 Selected Conceptual Requirements

In this section, selected generalized, conceptual requirements for Auto GCAS are included to provide context for the development of a generalized run time assurance requirements and architecture presented in the next chapter. Additional insights into design considerations can be found in [203, 187, 192, 191, 17, 204, 38].

3.2.1 Do No Harm

- The automatic recovery shall not cause harm to pilot, aircraft, or components.
- The automatic recovery maneuver shall not place the aircraft in an uncontrollable state.
- Auto GCAS shall not activate an automatic recovery maneuver if a failure of a subsystem supporting Auto GCAS exists.
- Auto GCAS shall leave the failed mode state if a failure does not exist.
- Subsystem monitors shall determine if a failure exists and be provided to Auto GCAS.
- Interlock conditions shall prevent Auto GCAS activation.
- The automatic recovery maneuver shall not activate during aerial refueling.
- The automatic recovery maneuver shall not activate when the aircraft has an excessively high angle of attack.
- The automatic recovery maneuver shall not activate when the aircraft velocity is too low.
- The pilot shall be able to interrupt a maneuver.
- The pilot shall be able to manually engage a maneuver.

3.2.2 Do Not Interfere

- The automatic recovery maneuver shall not activate during landings.
- The pilot shall be able to turn the system off.
- The pilot shall be able to select the protection level.
- Auto GCAS shall notify the pilot when the automatic recovery maneuver initiates and terminates.
- The system shall record data about each automatic recovery maneuver activation.
- The DTED used by the system shall have a resolution and accuracy that supports safe and nuisance free operation.

- Auto GCAS shall inform the pilot if the aircraft speed is too low for the automatic recovery maneuver.

3.2.3 Prevent Collisions

- The system shall conduct an automatic recovery maneuver when the projected recovery trajectory intersects the terrain profile contour with buffers.
- The system shall terminate an automatic recovery maneuver as soon as it is determined that the aircraft will clear the terrain threat.
- The recovery maneuver shall consist of a roll to wings level and a pull up.

3.3 Inspiration for Spacecraft Last-Instant Collision Avoidance System Design from Air Domain Systems

In this chapter, Auto GCAS was explored as a real-world example of an RTA system. Other air and space collision avoidance systems are also explored and captured in an air and space collision avoidance taxonomy in Appendix A. The goal of Chapter 4 is to create a generalizable run time assurance approach including a novel generalized architecture and patterns for formal hazard analysis and requirements elicitation. In Chapter 5, this methodology is evaluated on the development of an RTA architecture and formal requirements for a hypothetical last instant collision avoidance system for spacecraft.

This last section of this chapter focuses on the *last instant* collision avoidance category, which is by necessity the most automated, to better understand common requirements across both air and space domains. The name "last instant" was used to differentiate systems like The Traffic Collision Advisory System (TCAS) [209, 210], the Traffic Alert and Collision Avoidance System II (TCAS II) [211, 212], the Aircraft Collision Avoidance System X (ACAS-X) [213, 214, 215], AFRL's Sense and Avoid (SAA) system [210, 216, 217], and NASA's Detect and Avoid Alerting Logic for Unmanned Systems (DAIDALUS) program [218, 219] that operate tens of seconds ahead of a collision from systems like Auto ACAS that operate with seconds until the collision (time scales are longer in space domain). Here, air domain high-level collision avoidance requirements are examined and recommendations of which requirements concepts are shared or different across domains is discussed.

This part of the research investigates the mapping of high-level aircraft collision avoidance system

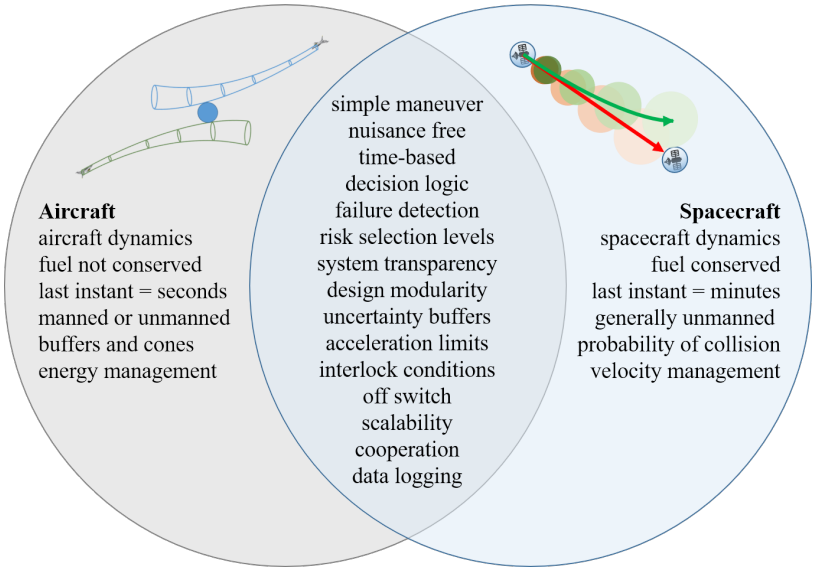


Figure 3.6: Differences and similarities in last instant collision avoidance systems across air and space domains

design considerations to the spacecraft domain.

The high-level requirements that are expected to be shared by both aircraft and spacecraft last instant systems are summarized in Fig. 3.6 and explained as follows:

- *simple maneuver*: A simple collision avoidance maneuver can be thoroughly evaluated offline and provides predictability that increases trust in the system [185], especially when the maneuver is consistent with what a human would have selected [17]. In Auto GCAS a single maneuver was used [186], while other instantiations of ground collision avoidance systems used three or more maneuvers [188, 38, 189, 190], and Auto ACAS used nine [187]. Further research is required to determine an appropriate finite set of maneuvers for spacecraft collision avoidance.

- *nuisance free*: In the aircraft domain, any automatic collision avoidance maneuver should begin as late as practical and return to operations as soon as possible to have a minimal impact on the mission of the aircraft or spacecraft [191, 192, 16]. For spacecraft that provide critical space-based services, a similar criteria will likely limit service downtime.

- *time available metric*: Aircraft systems used "time available" to maneuver as described in

Section A.2.4, (where zero time available would result in a maneuver that intersects the obstacle in a single point) [191, 192, 16], and time will also likely play into the maneuver decision for spacecraft.

- *decision logic*: Spacecraft will likely use a similar set of modes and conditions used to decide when it is safe to plan and engage a collision avoidance maneuver with considerations for safety interlock conditions, failures, human interaction with the system, and whether the condition to maneuver is present as are seen in the aircraft systems [203, 187, 191, 204, 38].

- *failure detection*: A critical element of last instant collision avoidance systems is failure detection [187] through a variety of monitoring methods [191], such as "heartbeat checks" for stale signals and "reasonableness checks" to evaluate whether signals are within a realistic range of values [38]. Special considerations may be made to mitigate single-point-of-failure in non-redundant hardware [203].

- *risk level selection*: Spacecraft collision avoidance systems could both benefit from aircraft-inspired operator tunable risk tolerances [191, 192, 38] that may impact the probability of collision threshold or size of the uncertainty buffers in the maneuver.

- *system transparency*: Spacecraft collision avoidance systems could also take inspiration from how and what information is presented to the human operator (or pilot) so that their trust in the system is calibrated in air collision avoidance systems [17, 16].

- *design modularity*: A key design concept in aircraft last instant collision avoidance system designs was the use of a modular architecture that could be used across many platforms, which enabled the use of a similar architecture on different aircraft classes [38].

- *uncertainty buffers*: Predicted maneuvers should have some buffer around them that describes the possible outcome of the maneuver with bounded disturbances from the environment [191, 192, 187]. Uncertainty can come from measurement of the current state or variations in the execution of a maneuver, both of which should factor into a spacecraft collision avoidance system design.

- *acceleration limits*: Both air and space collision avoidance maneuvers feature bounded translational and rotational acceleration limits, including:
  - *structural*: aircraft [38] or spacecraft baseline structural acceleration limit;
  - "*payload*": while a human pilot and spacecraft payload are not one-to-one equivalent, pilot physiological limits [204] and physical payload limits both represent an onboard system that may have tighter acceleration limit requirements than the baseline aircraft or spacecraft; and
  - *special configurations*: aircraft may have several configurations depending on additional fuel tanks or other mission-specific items that change the moment of inertia of the aircraft and can result in additional loads, while spacecraft have antenna, solar panels, booms, tethers and other deployable components that similarly will change the acceleration limits of the system.
- *interlocks*: Several interlock conditions may cause either system to go into standby temporarily. For example, an aircraft undergoing aerial refueling [38] or a spacecraft undergoing refueling from a service satellite may not want to automatically maneuver to avoid the other object.
- *off switch*: For both aircraft and spacecraft, a human operator (pilot or ground station operator) should have the ability to turn the system off, a feature that builds trust in the system [17].
- *scalability*: Both air and space collision avoidance systems will need to scale to avoid collisions with multiple objects in their neighborhood, in part to ensure that avoiding a collision with one object does not cause a collision with another [204, 187]. This is particularly challenging in the space domain when trying to avoid a cloud of debris rather than a single object [20].
- *cooperation*: Both aircraft [220, 204, 187] and spacecraft need to consider levels of cooperation between themselves and other aircraft or spacecraft. It may be more efficient or safer for two objects to coordinate a collision avoidance maneuver than for one or both to independently plan a maneuver.

- *data logging*: Information about a collision avoidance activation or predicted collision should be logged on either system for analysis after the event [38].

While air and space collision avoidance systems feature many similarities in the design, there are several key differences as follows:

- *dynamics*: The dynamics of an aircraft or spacecraft maneuver are very different. Aircraft last instant collision avoidance maneuvers feature roll and pull, bunt, and maintain options [187], while spacecraft may conduct in-plane or out-of-plane maneuvers, or maneuver using linearized relative motion dynamics or natural motion trajectories [221].

- *fuel conservation*: While fuel conservation is important for aircraft, fuel usage is not a huge design consideration for aircraft collision avoidance. However, fuel is a limited resource that generally cannot be replenished and is used conservatively on spacecraft.

- *last instant definition*: For aircraft, a last instant maneuver may be on the order of seconds ahead of a collision [191], while a conservative spacecraft maneuver may be days or weeks ahead [20].

- *manned versus unmanned*: The aircraft collision avoidance system examined relied on human-centric maneuver limits, depended on the human pilot as a supervisor, expected real time responses from the pilot, and had considerations for when a pilot experienced gravity-induced loss of consciousness [38]. The spacecraft class considered in this research is unmanned and should consider maneuver timing and duration limits for fault tolerance, as well as delays in ground responses and ground communication outages.

- *energy*: The Auto GCAS considered potential and kinetic energy of the aircraft in the decision to maneuver [38]. In the space domain, change in velocity (ΔV) required to conduct the maneuver may be a larger driving factor than spacecraft energy state.

- *uncertainty representation*: While aircraft maneuver uncertainty grows with time, when that prediction is on the order of seconds, the uncertainty growth is relatively small and can be represented as a cone around the maneuver [187]. Spacecraft uncertainty is often represented with ellipses and grows in a more complex shape over time resembling something closer to a

"basket," "banana," or "bean," requiring a different representation than that used for aircraft for longer duration predictions [18, 222, 223, 224, 225, 226, 227].

In summary and depicted in 3.6, roughly 15 conceptual requirements are expected to have equivalences in the air and space domain, while 6 conceptual requirements are expected to differ significantly. The requirements specified here are further refined in Chapter 5. For instance, the decision logic requirement corresponds to a subcomponent of the collision avoidance system that is expected to have at least 14 requirements defining the transition logic between modes, as described in [170].

One of the key differences between the aircraft and spacecraft domains to be explored further is the impact of information content, latency, accuracy, and precision on requirements. In the aircraft domain, shorter time horizons, onboard transponders and sensors (radar), and less time between measurements translates to less uncertainty accumulation compared to the spacecraft domain. These differences could be explored in more depth in future work.

Deeper examination of the requirements presented here inform future development of a metamodel for system-level formal and non-formal analysis of a spacecraft with an automatic collision avoidance system in the context of a space traffic management framework. In Chapter 5, design specification, requirements, and functional interface descriptions for spacecraft last instant automatic maneuvering are formally specified in temporal logic with inspiration from the air domain. These design specifications and requirements are analyzed for realizability, logical consistency, and logical entailment using formal methods [98, 81, 94, 96, 13]. The requirements are refined through hazard analysis to identify additional safety constraint requirements.

3.4 Summary

In this chapter, Auto GCAS was explored as a unique real-world example of an RTA system. Lessons learned from the development of Auto GCAS are integrated into the development of a generalizeable run time assurance approach featuring a novel generalized architecture and patterns for formal hazard analysis and requirements elicitation in Chapter 4. This approach is evaluated on the development of an RTA architecture and formal requirements for a hypothetical last instant collision avoidance system for spacecraft in Chapter 5.

CHAPTER 4
FORMAL RUN TIME ASSURANCE DESIGN APPROACH

Chapter 2 described the current state of the art in run time assurance, formal specification and analysis, and other techniques with a hint at how they are employed or expanded in the rest of this thesis. Chapter 3 investigated the development of a specific RTA system for clues to common requirements and architectural elements across aerospace RTA systems. This chapter describes the overall RTA design approach, including the first formally specified and analyzed generalized RTA architecture that includes a fault monitor, interlock monitor, and human-machine interface. While the presentation of this generalized approach is novel, its development was heavily influenced by the author's experience in the development and evaluation of automatic ground and midair collision avoidance systems [228, 186, 187, 202, 204, 229], as explored in the last chapter.

At the highest level, the requirements of RTA systems are to do no harm, do not interfere, and automatically respond to unsafe conditions. To do no harm, the system must monitor for faults and interlock conditions. A *fault* is a failure due to a design or component malfunction that make it unsafe to calculate or engage an automatic maneuver. An *interlock* condition is a condition where it is safe to compute automatic control, but unsafe to engage it. To not interfere, the system must monitor boundary violations (and the boundary violations must be sufficiently large to allow maximum safe operating region for the nominal controller), and it must be capable of interacting with a human supervisor. Finally, the system must be capable of monitoring and detecting unsafe conditions in order to engage a revert to a safe controller when appropriate.

4.1 General Run Time Assurance Architecture

The general run time assurance architecture is presented in Fig. 4.1 and features these primary functional components (functional components describe specific functions and interactions that may be implemented on one or more physical components):

- *Plant*: This functional component is the system under control and could be a spacecraft, aircraft, or other system.

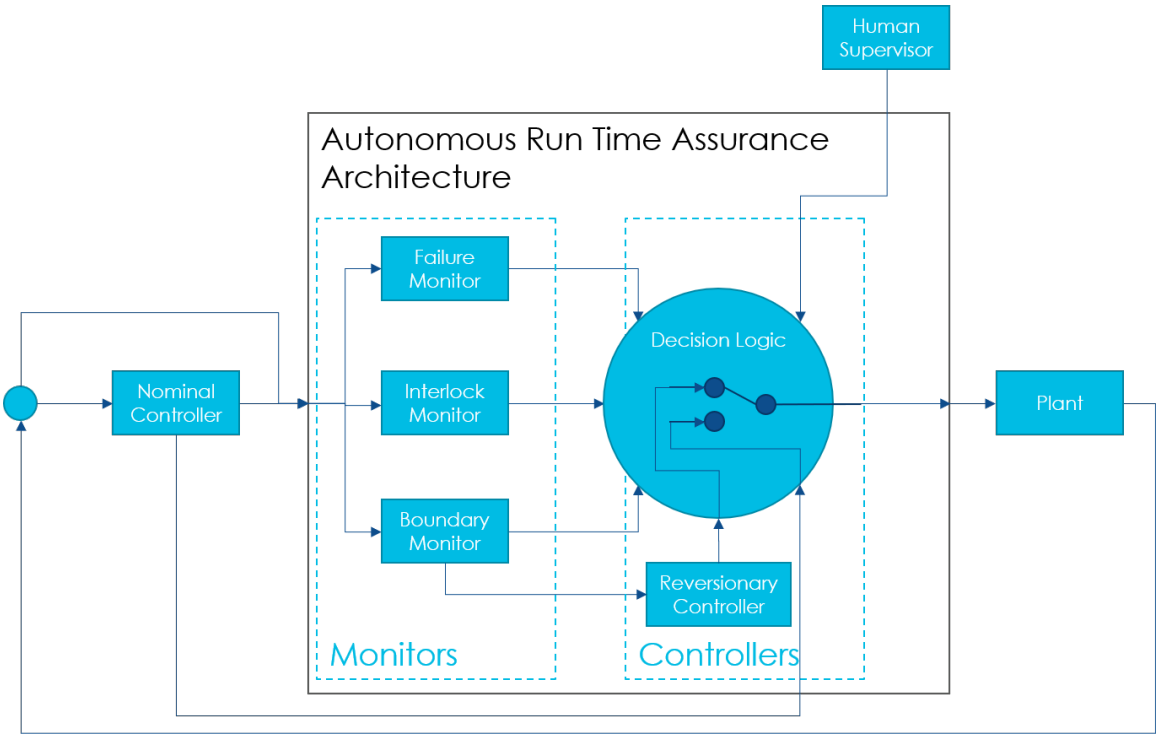


Figure 4.1: General Run Time Assurance Architecture

- *Nominal controller*: This functional component is the primary, high performance controller of the system that isn't verified with traditional offline verification techniques. Depending on the application this could be a human pilot, or other adaptive or complex control software for which offline verification techniques do not exist.

- *Reversionary controller*: This functional component continuously computes a simple, verified, control response to the state of the plant.

- *Human supervisor*: This functional component is a human on the loop that can interact with the nominal and reversionary controller in a variety of ways such as manually engaging the reversionary controller, turning the ability to switch to the reversionary controller off, manually switching from reversionary controller back to nominal controller, or adjusting the sensitivity of the decision to switch between controllers. In the case of a human-piloted vehicle, the human supervisor and nominal controller could be the same entity.

- *Failure monitor*: This functional component monitors the state of the plant, control subsystem, and subsystems and components supporting the control subsystem for failures. Informa-

tion about the failures is used in the decision logic to determine which controller is used.

- *Interlock monitor*: This functional component monitors the state of the plant and its subsystems for interlock conditions where it is unsafe for certain control actions to take place. Interlock conditions are mutually exclusive control actions. For example, door control and motion control on an elevator are interlock conditions where the elevator is prevented from moving unless the doors are closed, and the doors are locked closed while the elevator is in motion.

- *Boundary monitor*: This functional component monitors the state of the plant and the nominal controller for predetermined safety boundary violations. Examples of boundary violations might be a trajectory prediction that intersects the terrain (indicating an imminent ground collision) or a control input that would cause excessive acceleration (risking damage to the structure or components).

- *Decision Logic*: The functional component takes inputs from all the other components pictured and determines which control output, nominal or reversionary, to output to the plant.

While inclusion of a human supervisor, failure monitor, and interlock monitor is unique to this form of run time assurance, it is not wholly unprecedented, as was discussed in the last chapter.

4.1.1 Abstract Output States of RTA Components

In addition to monitoring for unsafe boundary violations, the proposed RTA architecture considers inputs from the human supervisor, the presence of safety interlock conditions, and faults on systems providing critical information. Detecting the presence of a condition to maneuver, failure condition, or interlock condition is an important element of the overall system development. None of these capabilities are trivial to develop. The abstracted output states of each component introduced here will be described further in future sections:

- failure (f): A failure is a condition where it is unsafe to compute or execute a maneuver because a fault is detected on a system that provides critical information used by the automatic maneuver system to decide when and how to maneuver. This is an output of the failure monitor component.

- interlock (i): A safety interlock is a condition in which it is unsafe to conduct a maneuver or conducting a maneuver may result in higher risk to the plant component. This is an output of the interlock monitor component.

- condition to maneuver (c): A condition to maneuver is true when a safety boundary has been crossed and the reversionary controller should take over, and false at all other times. Specification of this component is mission specific. This is an output of the boundary monitor component.

- maneuver completed (m): The maneuver completed flag is false when a system is maneuvering, and true when not actively maneuvering. This flag is important as part of the overall design principles that requires a maneuver be completed before engaging another maneuver (to prevent introduction of instability from rapidly switching between control strategies). This is an output of the reversionary controller component.

- person manually interrupts (p): The condition where a person manually interrupts a maneuver is included because maneuver operations often still include a human in the loop; including the condition in the logic allows for human supervision of autonomous operations. This is an output of the human supervisor component.

- person manually engages maneuver (e): The condition where a person decides to manually engage a maneuver for a variety of reasons; the system activates the current planned maneuver. This is an output of the human supervisor component.

The above abstracted states represent the core states of the RTA architecture, however other variations may be implemented depending on the needs of the specific system. For example, variations and lower abstractions of the failure monitor component, may also include identification of the specific fault or failure. In addition, there may be a variety of other human supervisor inputs such as system settings or opportunities to aid a maneuver.

Four states are proposed for the decision logic component, in order of precedence with failed having the highest priority, and maneuver having the lowest priority. These priorities are founded on a guiding principle that first the system should do no harm, second that it should not interfere with

the primary mission of the system, and third that it should maneuver when an appropriate condition exists. Transition between these four system states is determined by the state of the six conditions.

- *Failed* (F): In the failed state, the system does not predict conditions to maneuver or calculate automatic maneuver trajectories, because a critical system providing information for the prediction has failed. The system stays in the failed state as long as a failure is present (f).

- *Standby* (S): In the standby state, the system predicts conditions (like an imminent collision, or the need to maintain a position in a constellation) and calculates automatic maneuver trajectories, but is prevented from maneuvering. It is intended to account for safety interlock conditions, in which no failure is present, but some other condition exists that prevents the system from maneuvering. The system standby state is the initial state of the system, and each time the system is reset, it should first return to the standby state. The system stays in the standby state as long as there is no failure and an interlock condition is present ($\neg f \wedge i$).

- *Active* (A): In the active state, the system is predicting the need to maneuver, calculating automatic maneuvers, determining whether a maneuver need is imminent, and is prepared to activate a maneuver. While the need to maneuver is calculated in both standby and active states, the primary difference is that the system will not activate a maneuver in standby. The active state is the most vulnerable state to change as the system only remains in the active state if there are no failures, no interlocks, no manual interrupt, there is no maneuver condition present, and the operator has not manually engaged a maneuver ($\neg f \wedge \neg i \wedge \neg p \wedge \neg c \wedge \neg e$).

- *Maneuver* (M): In the maneuver state, the system's automatic maneuvers are prescribed by the need (which may be collision avoidance, station keeping, or rendezvous). The system stays in the maneuver state if the maneuver has not been completed and there is no manual interrupt from a human ($\neg m \wedge \neg p$). The rationale for this staying condition is that the safety of the system depends on completing the maneuver once it has started and may be compromised if the maneuver is stopped in the middle.

4.1.2 Decision Logic Functional Component Formalization

At the core of an automatic collision avoidance system is the responsibility of deciding whether to maneuver. The *decision logic* functional component accepts the state of several system conditions and determines whether the system is allowed to maneuver. This section first describes a set of conditions that impact the decision to maneuver is presented, then a set of system states is defined, and finally, a design specification is developed to constrain transition between states based on the previous state and the conditions of the system. This design specification is described in formal logic and shown as a finite state machine. The design specifications and requirements of the decision logic functional component could be applied to a variety of spacecraft automatic maneuver missions beyond collision avoidance such as proximity operations or station keeping. While introduced in Chapter 2, it is worth noting again for clarity that $\bigcirc^{-1}$ indicates that the formal statement is true for the previous timestep, and $\Box$ indicates that the formal statement is invariant, i.e. from a past time perspective, it is historically (always) true and has been true for all timesteps up to and including the current timestep.

Decision Logic Design Specification

Using the 4 states (F, S, A, M) and 6 conditions (f, i, c, m, p, e) described in Section 4.1.1, 14 design specifications define the transition system behavior in the decision logic functional component. These 14 design specifications are a logically equivalent, but reduced set of conditions corresponding to the safety requirements described later. The finite state transition system described by these design specifications is depicted in Fig. 4.2, which visually depicts the transitions between states. The ptLTL formulation of the design specification is listed in Table 4.1. The natural language expression of these design specifications and the rationale for each is listed below.

[DL01] Once the system enters the standby state (S), the system shall remain in the standby state (S) if there is no failure detected and an interlock condition exists ($\neg f \wedge i$).

Rationale: Initializing the system in the standby state ensures that the system has sufficient time to check for failures or interlocks before deciding to maneuver. The only ways to leave the standby state are to go to the active state if there is not an interlock condition and not a failure or to the failed state if there is a failure.

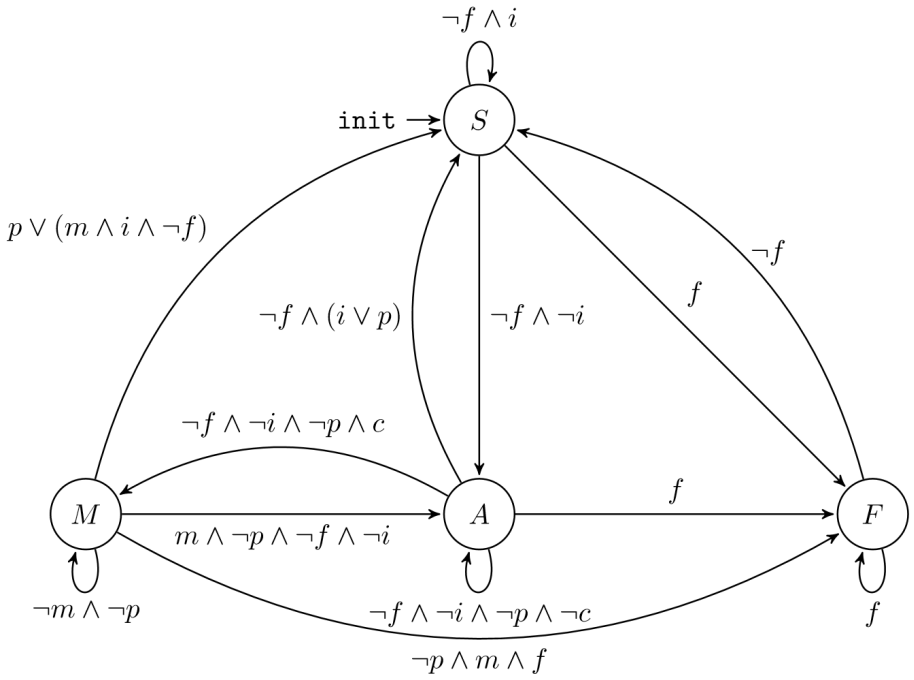


Figure 4.2: Finite state machine describing automatic maneuver decision logic defined by the design specifications.

Table 4.1: Decision logic functional component transition system design specification expressed in ptLTL.

ID				
DL01	$\bigcirc^{-1} S$	$\wedge(\neg f \wedge i)$	$\Longrightarrow$	S
DL02	$\bigcirc^{-1} F$	$\wedge f$	$\Longrightarrow$	F
DL03	$\bigcirc^{-1} M$	$\wedge \neg m \wedge \neg p$	$\Longrightarrow$	M
DL04	$\bigcirc^{-1} A$	$\wedge \neg f \wedge \neg i \wedge \neg p \wedge \neg c \wedge \neg e$	$\Longrightarrow$	A
DL05	$\bigcirc^{-1} F$	$\wedge \neg f$	$\Longrightarrow$	S
DL06	$\bigcirc^{-1} S$	$\wedge f$	$\Longrightarrow$	F
DL07	$\bigcirc^{-1} A$	$\wedge f$	$\Longrightarrow$	F
DL08	$\bigcirc^{-1} A$	$\wedge(\neg f \wedge (i \vee p))$	$\Longrightarrow$	S
DL09	$\bigcirc^{-1} M$	$\wedge(p \vee (m \wedge i \wedge \neg f))$	$\Longrightarrow$	S
DL10	$\bigcirc^{-1} S$	$\wedge(\neg f \wedge \neg i)$	$\Longrightarrow$	A
DL11	$\bigcirc^{-1} M$	$\wedge m \wedge \neg p \wedge \neg f \wedge \neg i$	$\Longrightarrow$	A
DL12	$\bigcirc^{-1} A$	$\wedge \neg f \wedge \neg i \wedge \neg p \wedge (c \vee e)$	$\Longrightarrow$	M
DL13	$\bigcirc^{-1} M$	$\wedge \neg p \wedge m \wedge f$	$\Longrightarrow$	F
DL14		`init`	$\Longrightarrow$	S

[DL02] Once the system enters the failed state (F), the system shall remain in the failed state (F) if a failure condition (f) is still detected.

Rationale: Failed is a worst-case scenario, where it is undesirable to calculate an automatic

maneuver trajectory because the information the system is basing that calculation on could be unreliable, so it has priority over the other states if a failure is detected. It is undesirable for the system to switch out of a failed state unless no failures exist.

[DL03] Once system enters the maneuver state (M), the system shall remain in the maneuver state as long as the maneuver has not been completed and there is no manual interrupt ($\neg m \wedge \neg p$).

Rationale: It is desired that the maneuver be completed before the system switches to another state, so that the maneuver is not stopped prematurely if a failure or interlock condition is detected. This is because while the end state of the calculated maneuver may be considered safe, prematurely stopping could result in an unsafe condition at some time in the future. The only reason for a system to be stopped in the middle of a maneuver is if a human supervising the maneuver sends a manual interrupt.

[DL04] Once the system enters the active state (A), the system shall remain in the active state as long as there is no failure detected, no interlock detected, no manual interrupt, no condition to maneuver exist, and the operator has not manually commanded a maneuver ($\neg f \wedge \neg i \wedge \neg p \wedge \neg c \wedge \neg e$).

Rationale: The active state is the most vulnerable state for transition to other states and will only hold true while there are no failures, no interlocks, and a condition to maneuver is not present. This prioritization prevents maneuvers that might compromise safety of the system, assuming that in the spacecraft environment, the safest action is often to do nothing.

[DL05] The system shall transition from the failed state (F) to the standby state (S) when there is no failure detected ($\neg f$).

Rationale: If a failure is not detected, the system should switch out of the failed state. The system switches into the standby state before switching back to the active state because no maneuver trajectories or conditions to maneuver are calculated in the failed state, but they are calculated in the standby and active states. Switching from failed to standby provides sufficient time to calculate maneuver trajectories, determine whether the conditions for an automatic maneuver exist, and check that no interlock or failure conditions are detected, before switching the system into an active state.

[DL06] The system shall transition from the standby state (S) to the failed state (F) when a failure is detected (f).

Rationale: If a failure is detected, the system will transition to the failed state which has the highest priority and prevents the system algorithm from actively predicting a condition to maneuver based on compromised data integrity.

[DL07] The system shall transition from the active state (A) to the failed state (F) when a failure is detected (f).

Rationale: If a failure is detected, the system will transition to the failed state which has the highest priority, and prevent the system algorithm from actively predicting potential need for an automatic maneuver based on compromised data integrity.

[DL08] The system shall transition from the active state (A) to the standby state (S) when there is no failure and either there is an interlock or a manual interrupt ($\neg f \wedge (i \vee p)$).

Rationale: If the system is in active state and there is no failure, but an interlock condition is detected that makes it unsafe or undesirable to automatically maneuver, the system should switch to the standby state. If there is not a failure and a human monitoring the system wants to prevent a maneuver through a manual interrupt, then the system should not be able to transition to an automatic maneuver state.

[DL09] The system shall transition from the maneuver state (M) to the standby state (S) when either a manual interrupt is selected, or the maneuver is completed and there is an interlock condition and no failure condition detected ($p \vee (m \wedge i \wedge \neg f)$).

Rationale: If the system is actively maneuvering, it should complete its maneuver before transitioning. Once the maneuver is complete, if an interlock condition is detected the system should switch directly to a standby. If, however, during the maneuver, a human supervisor determines the need to abort the maneuver, a manual interrupt will also send the system to the standby state.

[DL10] The system shall transition from the standby state (S) to the active state (A) when the there is no failure and no interlock detected ($\neg f \wedge \neg i$).

Rationale: If the system is in the standby state and the interlock condition that caused the sys-

tem to go into standby no longer exists, and there is no failure, then the system will transition into the active state.

[DL11] The system shall transition from the maneuver state (M) to the active state (A) when the maneuver is completed and there is no manual interrupt, no failure, and not interlock present ($m \wedge \neg p \wedge \neg f \wedge \neg i$).

Rationale: If the system is in the maneuver state, where it is actively maneuvering and there are no interlock, failure, or manual interrupt conditions detected, then on completion of the maneuver, the system will transition back to an active state where it will be prepared to immediately conduct another automatic maneuver should a condition to maneuver exist.

[DL12] The system shall transition from the active state (A) to the maneuver state (M) when there is no failure, no interlock, no manual interrupt, and either the condition to maneuver is present or a command is manually activated($\neg f \wedge \neg i \wedge \neg p \wedge (c \vee e)$).

Rationale: Maneuver is the lowest priority of the states. As such, only when no failure conditions or interlock conditions are detected, and a condition to maneuver is present, will the system conduct an automatic maneuver. When there is a failure, interlock, or manual interrupt condition, it is assumed that it is safer to do nothing than to maneuver.

[DL13] The system shall transition from the maneuver state (M) to the failed state (F) when there is no manual interrupt, the maneuver has been completed, and a failure exists ($\neg p \wedge m \wedge f$).

Rationale: The system shall not transition from the maneuver unless it has been completed or there is a manual interrupt. Once the maneuver is completed, if there is no interrupt, the next highest priority is a failure, which will transition the system directly to the failed state ($\texttt{init} \implies S$).

[DL14] The system shall initialize in the standby state. Initializing the system in the standby state ensures that the system has enough time to check for failures or interlocks before deciding to maneuver.

Rationale: Initializing the system in the standby state ensures that the system has sufficient time to check for failures or interlocks before deciding to maneuver.

In addition to the transition design specifications, a set of seven specifications describe the

interface to the decision logic functional block. These specifications were inspired by requirements stating what the decision logic should consider for aircraft systems and are formally interpreted as inputs and outputs in this work. These specifications are captured in the inputs and outputs section of the SpeAR Decision Logic system specification but are not formalized in temporal logic or analyzed using formal methods.

Table 4.2: Formal description of Decision Logic functional block interface.

Specification ID	Variable	Description	Interface
DL15	f	failure exists	input
DL16	i	interlock exists	input
DL17	p	manual interrupt by a person	input
DL18	c	condition to maneuver	input
DL19	m	maneuver completed	input
DL20	e	manually commanded by a person	input
DL21	sys_{power}	on/off power state	input
DL22	$DLmode$	decision logic mode	output

[DL15] The decision logic shall accept the failure state f of supporting subsystems as an input.

Rationale: A failure monitor is envisioned to be a separate component to promote modularity (monitors should be designed specific to the specific hardware on board). For example, if an automatic maneuver system relies on GPS, but the GPS system is failed, the automatic maneuver system should be aware to prevent acting on failed information.

[DL16] The decision logic shall accept the interlock state i of the spacecraft as an input.

Rationale: An interlock monitor is envisioned to be a separate component to promote modularity by monitoring interlock conditions specific to the spacecraft and mission.

[DL17] The decision logic shall accept manual interrupts from the operator p as an input.

Rationale: The operator's ability to interrupt maneuvers should be accepted as an input to stop the automatic maneuver.

[DL18] The decision logic shall accept the condition to maneuver c as an input.

Rationale: A prediction function separate to the decision logic that monitors for a specific condition to maneuver c (such as collision avoidance, station keeping, etc.) promotes modularity of the system and should be known by the decision logic.

[DL19] The decision logic shall accept maneuver completed m as an input.

Rationale: The decision logic should know if a maneuver has been completed to be able to decide whether it is appropriate to activate another maneuver.

[DL20] The decision logic shall accept manual activation commands from the operator e as an input.

Rationale: The operator's ability to manually initiate maneuvers is accepted as an input to activate a maneuver.

[DL21] The decision logic shall accept the on/off state sys_{power} from the operator as an input.

Rationale: The operator's ability to turn the system off should be accepted as an input to stop the automatic maneuver to allow full operator control, especially in off-nominal states.

[DL22] The decision logic shall output the current decision logic mode $DLMode$.

Rationale: The decision logic mode may need to be used by other systems, and will be provided to the maneuver controller.

Decision Logic Requirements

Requirements may be used in a formal analysis technique like model checking [13] to validate that the a design specification meets desirable the system behavior. In this section, requirements are specified for all the conditions that can affect transition from each Decision Logic transition system state, while marking conditions that do not impact the state as "don't cares" ($\times$). By systematically setting each condition that does impact the state to true (1) or false (0) and specifying the next state, it can be verified that the logic in the requirements account for all possible conditions. The requirements relating to each phase are described in Tables 4.3-4.6.

Documented in Table 4.3, the only condition which impacts transition from the failed state (F) is the detection of a failure (f). Once in the failed state (F), the system will stay in the failed state if a failure is detected (f). When a failure is no longer detected ($\neg f$), the system will transition to the standby state (S).

As seen in Table 4.4, the only conditions which impact transition from the standby state (S), are the detection of a failure (f) or detection of an interlock condition (i). In the standby state (S), if a failure is detected (f), the system will transition to the failed state (F). The system will stay in the standby state (S) as long a there is no failure detected but there is an interlock detected ($\neg f \wedge i$).

Table 4.3: Transition and Staying Condition Requirements for Failed State

Requirement	Previous State	f	i	p	c	m	e	Condition Summary	Operating State
DLRF01	F	0	$\times$	$\times$	$\times$	$\times$	$\times$	$\neg f$	S
DLRF02	F	1	$\times$	$\times$	$\times$	$\times$	$\times$	f	F

The system will transition from standby (S) to active (A) when there is no interlock and no failure condition ($\neg i \wedge \neg f$).

Table 4.4: Transition and Staying Condition Requirements for Standby State

Requirement	Previous State	f	i	p	c	m	e	Condition Summary	Operating State
DLRS01	S	0	0	$\times$	$\times$	$\times$	$\times$	$\neg f \wedge \neg i$	A
DLRS02	S	0	1	$\times$	$\times$	$\times$	$\times$	$\neg f \wedge i$	S
DLRS03	S	1	0	$\times$	$\times$	$\times$	$\times$	$f \wedge \neg i$	F
DLRS04	S	1	1	$\times$	$\times$	$\times$	$\times$	$f \wedge i$	F

Evident from Table 4.5, all but the maneuver completed flag (m) impact the system transition from active state (A). Once in the active state, if a failure is detected (f), the system will transition to the failed state (F). If there is no failure and either an interlock condition exists or the manual interrupt is engaged by a person ($f \wedge (i \vee p)$), the system will transition to the standby state (S). If there is no failure detected, no interlock detected, no manual interrupt and a condition to maneuver exists or a maneuver is manually commanded by the operator ($\neg f \wedge \neg i \wedge \neg p \wedge (c \vee e)$), the system will transition to the maneuver state (M). If there is no failure detected, no interlock condition detected, no manual interrupt, no condition to maneuver present, and no manually commanded maneuver ($\neg f \wedge \neg i \wedge \neg p \wedge \neg c \wedge \neg e$), then the system will remain in active (A).

As seen in Table 4.6, the maneuver state (M) is impacted by all conditions except the condition to maneuver (c). Once in the maneuver state, if a manual interrupt is engaged or the maneuver is completed and an interlock is detected ($p \vee (m \wedge i)$), the system will switch to the standby state (S). If the maneuver is completed and a failure is detected ($m \wedge f$), the system will switch to the failed state (F). If the maneuver is completed and there is not a manual interrupt, not failure detected, and not an interlock detected ($m \wedge \neg p \wedge \neg f \wedge \neg i$), then the system will transition to the active state (A). If there is no manual interrupt and the maneuver is not completed ($\neg p \wedge \neg m$), then the system will stay in the maneuver state.

Table 4.5: Transition and Staying Condition Requirements for Active State

Requirement	Previous State	f	i	p	c	m	e	Condition Summary	Operating State
DLRA01	A	0	0	0	0	$\times$	0	$\neg f \wedge \neg i \wedge \neg p \wedge \neg c \wedge \neg e$	A
DLRA02	A	0	0	0	0	$\times$	1	$\neg f \wedge \neg i \wedge \neg p \wedge \neg c \wedge e$	M
DLRA03	A	0	0	0	1	$\times$	0	$\neg f \wedge \neg i \wedge \neg p \wedge \neg c \wedge e$	M
DLRA04	A	0	0	0	1	$\times$	1	$\neg f \wedge \neg i \wedge \neg p \wedge c \wedge e$	M
DLRA05	A	0	0	1	0	$\times$	$\times$	$\neg f \wedge \neg i \wedge p \wedge \neg c$	S
DLRA06	A	0	0	1	1	$\times$	$\times$	$\neg f \wedge \neg i \wedge p \wedge c$	S
DLRA07	A	0	1	0	0	$\times$	$\times$	$\neg f \wedge i \wedge \neg p \wedge \neg c$	S
DLRA08	A	0	1	0	1	$\times$	$\times$	$\neg f \wedge i \wedge \neg p \wedge c$	S
DLRA09	A	0	1	1	0	$\times$	$\times$	$\neg f \wedge i \wedge p \wedge \neg c$	S
DLRA10	A	0	1	1	1	$\times$	$\times$	$\neg f \wedge i \wedge p \wedge c$	S
DLRA11	A	1	0	0	0	$\times$	$\times$	$f \wedge \neg i \wedge \neg p \wedge \neg c$	F
DLRA12	A	1	0	0	1	$\times$	$\times$	$f \wedge \neg i \wedge \neg p \wedge c$	F
DLRA13	A	1	0	1	0	$\times$	$\times$	$f \wedge \neg i \wedge p \wedge \neg c$	F
DLRA14	A	1	0	1	1	$\times$	$\times$	$f \wedge \neg i \wedge p \wedge c$	F
DLRA15	A	1	1	0	0	$\times$	$\times$	$f \wedge i \wedge \neg p \wedge \neg c$	F
DLRA16	A	1	1	0	1	$\times$	$\times$	$f \wedge i \wedge \neg p \wedge c$	F
DLRA17	A	1	1	1	0	$\times$	$\times$	$f \wedge i \wedge p \wedge \neg c$	F
DLRA18	A	1	1	1	1	$\times$	$\times$	$f \wedge i \wedge p \wedge c$	F

Table 4.6: Transition and Staying Condition Requirements for Maneuver State

Requirement	Previous State	f	i	p	c	m	e	Condition Summary	Operating State
DLRM01	M	0	0	0	$\times$	0	$\times$	$\neg f \wedge \neg i \wedge \neg p \wedge \neg m$	M
DLRM02	M	0	0	0	$\times$	1	$\times$	$\neg f \wedge \neg i \wedge \neg p \wedge m$	A
DLRM03	M	0	0	1	$\times$	0	$\times$	$\neg f \wedge \neg i \wedge p \wedge \neg m$	S
DLRM04	M	0	0	1	$\times$	1	$\times$	$\neg f \wedge \neg i \wedge p \wedge m$	S
DLRM05	M	0	1	0	$\times$	0	$\times$	$\neg f \wedge i \wedge \neg p \wedge \neg m$	M
DLRM06	M	0	1	0	$\times$	1	$\times$	$\neg f \wedge i \wedge \neg p \wedge m$	S
DLRM07	M	0	1	1	$\times$	0	$\times$	$\neg f \wedge i \wedge p \wedge \neg m$	S
DLRM08	M	0	1	1	$\times$	1	$\times$	$\neg f \wedge i \wedge p \wedge m$	S
DLRM09	M	1	0	0	$\times$	0	$\times$	$f \wedge \neg i \wedge \neg p \wedge \neg m$	M
DLRM10	M	1	0	0	$\times$	1	$\times$	$f \wedge \neg i \wedge \neg p \wedge m$	F
DLRM11	M	1	0	1	$\times$	0	$\times$	$f \wedge \neg i \wedge p \wedge \neg m$	S
DLRM12	M	1	0	1	$\times$	1	$\times$	$f \wedge \neg i \wedge p \wedge m$	S
DLRM13	M	1	1	0	$\times$	0	$\times$	$f \wedge i \wedge \neg p \wedge \neg m$	M
DLRM14	M	1	1	0	$\times$	1	$\times$	$f \wedge i \wedge \neg p \wedge m$	F
DLRM15	M	1	1	1	$\times$	0	$\times$	$f \wedge i \wedge p \wedge \neg m$	S
DLRM16	M	1	1	1	$\times$	1	$\times$	$f \wedge i \wedge p \wedge m$	S

Decision Logic Discussion

While the content of this section was previously published in [170], it is included here for completeness, with a few slight modifications. In the initial logic, the operator could not manually engage a maneuver. However, after evaluating the Auto GCAS design specifications and requirements and its pilot activated recovery system [201, 191, 192, 202], which enabled pilots to manually engage Auto GCAS recovery maneuvers, it was determined that such a functionality would also apply to spacecraft. In the case study in this work, a ground operator may wish to manually abort a proximity operation such as docking by activating a collision avoidance maneuver. To accommodate this in the logic, only a few changes were made to the design and safety requirements. First, a 6th Boolean state, e was added to the list of states to indicate that a ground station operator has manually engaged a collision avoidance maneuver. Next, design specification DL04 was updated to include $\wedge\neg e$ in the conditions to stay in the active state, and DL12 was updated from c to $(c \vee e)$ as a condition to transition from the active state to the maneuver state. A few safety requirements were also changed. Requirement DLAR01 was updated to include $\wedge\neg e$ as a condition to stay in the active state, DLAR02 was split into 3 conditions to account for possible combinations of c and e that could transition the system from the active state to the maneuver state, and e was set to a "don't care" for the other requirements.

4.2 Design Specification and Requirements Patterns

In this section, the general requirements patterns used in the specification and verification of the run time assurance architecture are described.

4.2.1 Initialization Pattern

The initialization pattern is common across all components. To completely describe component dynamics, the initial state and reaction to a given input in each state must be specified. For an *initial state* q, the value of $q(x)$ for every state variable x is consistent with the initialization of x [78]. The initialization design specification pattern is described by Eqn. 4.1.

$$\texttt{init} \implies state \tag{4.1}$$

The specific initial state of each component is determined as part of the design process. This is one of many design specifications for each component.

4.2.2 Decision Logic Patterns

The decision logic design specifications and requirements both follow a simple specification pattern that the previous decision logic mode ($\bigcirc^{-1} DLMode$) and the state of a subset of the RTA abstract system states f, i, p, c, m, e implies the current decision logic mode $DLMode$, as described below.

$$\begin{aligned} (\bigcirc^{-1} DLMode) \wedge (f \wedge i \wedge p \wedge c \wedge m \wedge e) \implies DLMode, \\ DLMode \in \{F, S, M, A\} \\ f, i, p, c, m, e \in \{true, false, \times\} \end{aligned} \tag{4.2}$$

4.2.3 Interlock Monitor Patterns

Each of the interlock condition design specifications follow the pattern that always a condition implies the system is in interlock ($\Box condition \implies i$), except for the initial state where it is assumed an interlock condition does not exist $\neg i$. The state transition of the interlock condition follows a distinct pattern: the disjunction ("or") of all of the individual interlock requirements φ_{IL_i} describes the interlock present state i, while the conjunction ("and") of all the requirements describes the absence of an interlock $\neg i$, as summarized by Eqn. 4.3.

$$\begin{aligned} i &:= \bigvee_{i=1}^{k} \varphi_{IL_i} \\ \neg i &:= \bigwedge_{i=1}^{k} \neg \varphi_{IL_i} \end{aligned} \tag{4.3}$$

Each individual interlock requirement follows the same pattern, shown in Eqn. 4.4. However, the states used to define the $condition$ that implies interlock may vary and are application specific.

$$\Box condition \implies i. \tag{4.4}$$

Two categories of interlock conditions are *temporary* and *permanent*. Temporary interlocks occur during finite activities likely to occur frequently throughout a mission, while permanent interlocks occur during a very small set of conditions likely only encountered once in the life on the platform. Examples of temporary and permanent interlocks are described in the case study in the net chapter.

4.2.4 Ground Computer Pattern

At this early stage of development, the ground computer's responsibility is to update the state model with information from the operator or the spacecraft and the only pattern used is to check equivalence of the model with the inputs. The design specification is a record update and the requirements check that each variable was updated correctly. This pattern is used in Section 5.3.2.

$$\varphi_{RGC_i} : x_{model} = x_{source} \tag{4.5}$$

4.2.5 Other Specification Patterns

The other components primarily feature record updates, or conditions described in the safety constraints in the next section.

4.3 Safety Specification Patterns

Requirements can be sourced from both the safety constraints and unsafe control actions from the STAMP and STPA analysis. The safety constraints are sourced top-down directly from high level accidents and the hazards that could case those accidents. Unsafe control actions are sourced from a bottom-up analysis of signals and information exchanged between different components of the system.

Safety constraints corresponding directly to the limited set of hazards fell into three general categories: acceleration or velocity limits, pointing and time-bounded pointing constraints, and interlock conditions. Each of these patterns are described in more detail in this section.

4.3.1 Acceleration and Velocity Constraints

The simplest safety constraint pattern is constraints on the upper limits of translational and rotational accelerations for all time, i.e. the acceleration in any given axis is always smaller than some maximum and larger than some minimum value. These pattern limits are simple enough to be expressed in propositional, first order, or linear temporal logic and are summarized by Eqs. 4.6-4.7, where a represents any translational or angular position variable. Acceleration constraints prevent damage to the spacecraft and its components such as a structural failure resulting from excessive forces during acceleration, or excessive wear on an actuator that frequently operates near its limits. Velocity limits on the spacecraft help ensure the spacecraft is controllable within a finite time horizon. For instance, it may be difficult or impossible to slow a vehicle with high angular velocity so that it is able to achieve communication requirements [230].

$$\varphi_{acceleration_{limit}} = \square(\ddot{a} \leq \ddot{a}_{upper_{limit}}) \wedge (\ddot{a} \geq \ddot{a}_{lower_{limit}}) \tag{4.6}$$

$$\varphi_{velocity_{limit}} = \square(\dot{a} \leq \dot{a}_{upper_{limit}}) \wedge (\dot{a} \geq \dot{a}_{lower_{limit}}) \tag{4.7}$$

4.3.2 Pointing and Time-Bounded Pointing Constraints

Safety constraint patterns also include time-bounded requirements on spacecraft orientation. For communication, data transfer, solar panel charging, sensor pointing keep out zones, and duration limited attitudes, a variation of a pointing constraint or time-bounded pointing constraint is used. First, a centerline unit vector $\hat{n}$ of the desired or undesired attitude is defined. This unit vector may be aligned with the boresight of the sensor or antenna or orthogonal to a solar panel or spacecraft surface, depending on the requirement. From this pointing unit vector, all other angles are measured. Second, an angle around this centerline unit vector $\theta_{\hat{n}}$ is defined by variables such as an antenna or sensor's solid angle field of view α, a maximum sun incidence angle for charging, a generalized unsafe angle from the unit vector θ_{US}, and/or a safety buffer angle β.

In this simplest cases of this requirement, such as attitude keep out zones or solar panel charging, the angle between $\hat{n}$ and the object of interest (such as the sun) should be less than the desired angle

$\theta_{desired}$ or greater than the undesired angle $\theta_{undesired}$:

$$\varphi_{attitude_{exclusion}} = \Box\theta_{\hat{n}} \geq \theta_{undesired}, \tag{4.8}$$

$$\varphi_{attitude_{desired}} = \Box\theta_{\hat{n}} \leq \theta_{desired}. \tag{4.9}$$

When the pointing requirement is only to be met during some scheduled time, or under some pointing condition $X_{pointing}$ the scope of these constraints may be reduced from always ($\Box$) to during some scheduled time $t_{scheduled}$ in the form:

$$\varphi_{attitude_{desired}} = \Box(X_{pointing} \vee t_{scheduled}) \implies \theta_{\hat{n}} \leq \theta_{desired}. \tag{4.10}$$

For more complex instances, labeling functions are used to define a line of sight (LOS) and field of view (FOV) property. The LOS property is true when the angle between object of interest and $\hat{n}$, sometimes called the zenith angle θ_s is less than some max zenith angle $\theta_{s_{max}}$. The FOV property is true when the angle between the centerline unit vector and some desired object (such as a receiving ground station) or undesired object (such as the sun), the fixation angle θ_R, is within the angle defined by the second step.

$$L(\theta_s, \theta_R) = \begin{cases} \emptyset, & \text{if } \theta_s > \theta_{s_{max}} \text{ and } \theta_R > \theta_{desired} \\ FOV & \text{if } \theta_s > \theta_{s_{max}} \text{ and } \theta_R \leq \theta_{desired} \\ LOS & \text{if } \theta_s \leq \theta_{s_{max}} \text{ and } \theta_R > \theta_{desired} \\ LOS \wedge FOV & \text{if } \theta_s \leq \theta_{s_{max}} \text{ and } \theta_R \leq \theta_{desired}. \end{cases} \tag{4.11}$$

Then the safety requirement becomes:

$$\varphi_{attitude_{labeled}} = \Box t_{scheduled} \implies (LOS \wedge FOV). \tag{4.12}$$

4.3.3 Interlock Conditions

The third type of safety constraint described interlock conditions, or conditions where it is unsafe to maneuver even when a fault is not present. These conditions were identified as ways to allow a human operator to intervene to prevent excessive fuel or actuator use, by limiting duration of maneuvers or preventing maneuvers when the fuel level f_l below a fuel level threshold f_{l_t} or end of life reserve $f_{l_{EOL}}$ as follows:

$$\varphi_{duration_{limit}} = \text{(condition on time)} \implies i, \text{ or} \tag{4.13}$$

$$\varphi_{fuel_{limit}} = (f_l \leq f_{l_t}) \vee (f_l \leq f_{l_{EOL}}) \implies i. \tag{4.14}$$

4.4 Additional Considerations

Three other important considerations for RTA design are independence of the RTA system from the system under observation, redundant monitoring, and detection of RTA failures beyond physics to include computational failures. Some systems separate the RTA in software only, while other systems such as the safe Testing of Autonomy in Complex Environments (TACE) also separate the RTA system in hardware [231]. Hardware separation helps ensure that failures in autonomy components do not propagate to the lower level control and backup systems and enable unique testing features. Another consideration in RTA design is whether to develop redundant monitors. In Auto GCAS development, SWIM conducted hardware and software tests in four redundant flight control computer processors [203]. To check for computational failures in developmental Auto GCAS, SWIM used heartbeat checks and reasonableness checks to monitor for computation failures [38].

4.5 Summary

In this chapter, the generalized RTA architecture featuring fault monitoring, interlock monitoring, and a human supervisor and the abstract states of the system were described. Then patterns for design specifications, requirements and safety constraints (requirements specifically from hazard

analysis) were presented. This RTA architecture and set of patterns are used in the next chapter to formally specify and analyze the requirements and architecture of a hypothetical spacecraft automatic collision avoidance system.

CHAPTER 5

CASE STUDY II: ILLUSTRATION OF IMPROVED RTA AND REQUIREMENTS ELICITATION APPROACH IN HYPOTHETICAL LAST INSTANT SPACECRAFT COLLISION AVOIDANCE SYSTEM

This chapter explores the application of the RTA and requirements approach in the design of a novel system for last instant spacecraft collision avoidance and includes the development of a system model, design specifications, and requirements for spacecraft last instant collision avoidance systems. First, the scope of the collision avoidance problem and associated reference frames and dynamics are explained. Second, requirements are identified from standards and regulations. Third, requirements are identified from constraints in spacecraft collision avoidance literature. Fourth, additional requirements are identified based on hazard analysis. Fifth, the requirements from literature and hazard analysis are combined with requirements identified in the aircraft collision avoidance domain and further refined into design specifications and requirements. These design specifications and requirements are then captured in formal logic, which can be analyzed manually for common patterns and automatically with formal methods for realizability, logical entailment, logical consistency, and traceability. While the focus of this chapter is on the design of an last instant automatic collision avoidance system for spacecraft, many design specifications and requirements are applicable to longer time horizons and other automatic maneuver missions as well.

5.1 Problem Scoping

To provide scoping to the collision avoidance requirements, a proximity operations scenario was selected [232, 233]. Since multiple spacecraft collisions have occurred during rendezvous [234, 235, 236, 237], this research is scoped to consider last instant collision avoidance for spacecraft attempting rendezvous. In order to keep the requirements general, no specific set of sensors or actuators are considered. This has some specific implications such as assumed ability to maneuver in any direction. In reality a collision avoidance maneuver would likely include an attitude maneuver to align the thrusters in a desired direction before maneuvering [233, 238, 239]. In addition, this

research stops short of completing design of a collision avoidance system.

A conceptual depiction of the collision avoidance scenario is presented in Fig. 5.1, where two spacecraft (SC1 and SC2) with connections to independent ground stations (GS1 and GS2) are conducting a rendezvous and docking operation and violate safety boundaries triggering a collision avoidance maneuver. The collision avoidance maneuver is assumed to select one of multiple nearby natural motion trajectories to maneuver to avoid the collision. Once the spacecraft transfers to a natural motion trajectory such as an ellipse, periodic line segment, or stationary point relative to the other spacecraft, it will remain in the trajectory. While the spacecraft will drift from the trajectory over time with the accumulation of disturbances, it remains there for enough time for ground operators to review the operation and determine appropriate next steps.

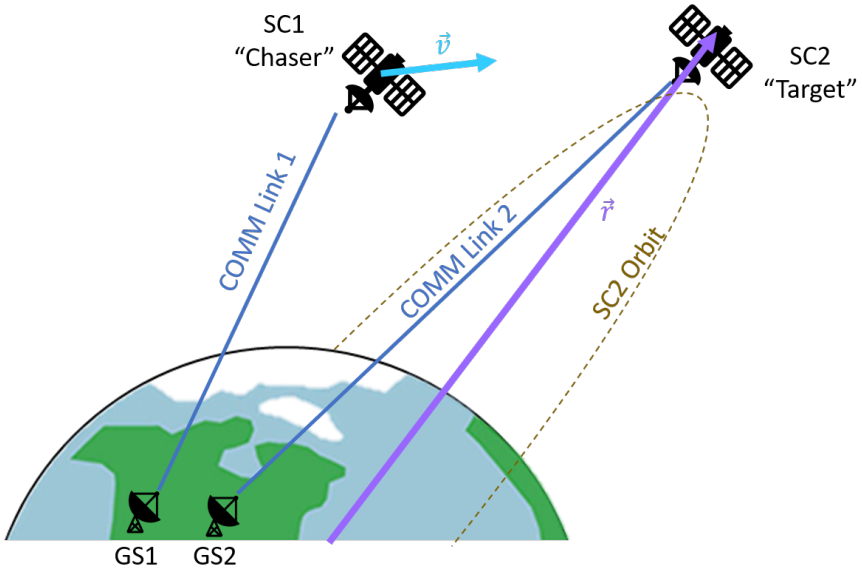


Figure 5.1: Conceptual depiction of two spacecraft operating in close proximity with communication back down to separate ground stations on earth.

Examples of multiple candidate elliptical trajectories is depicted in Fig. 5.2. The top candidate elliptical collision avoidance trajectories are highlighted in a darker green. In Fig. 5.3, a chaser spacecraft is depicted as rendezvousing with the target at the center using online in-plane motion. Several in-plane ellipses are considered for collision avoidance trajectories. The red ellipse behind the spacecraft is excluded because of the excess fuel required to reverse course, while the next nearest ellipse in green is the top choice and other blue ellipses remain candidate trajectories.

Assuming the spacecraft are in a near circular orbit and within a few kilometers of one an-

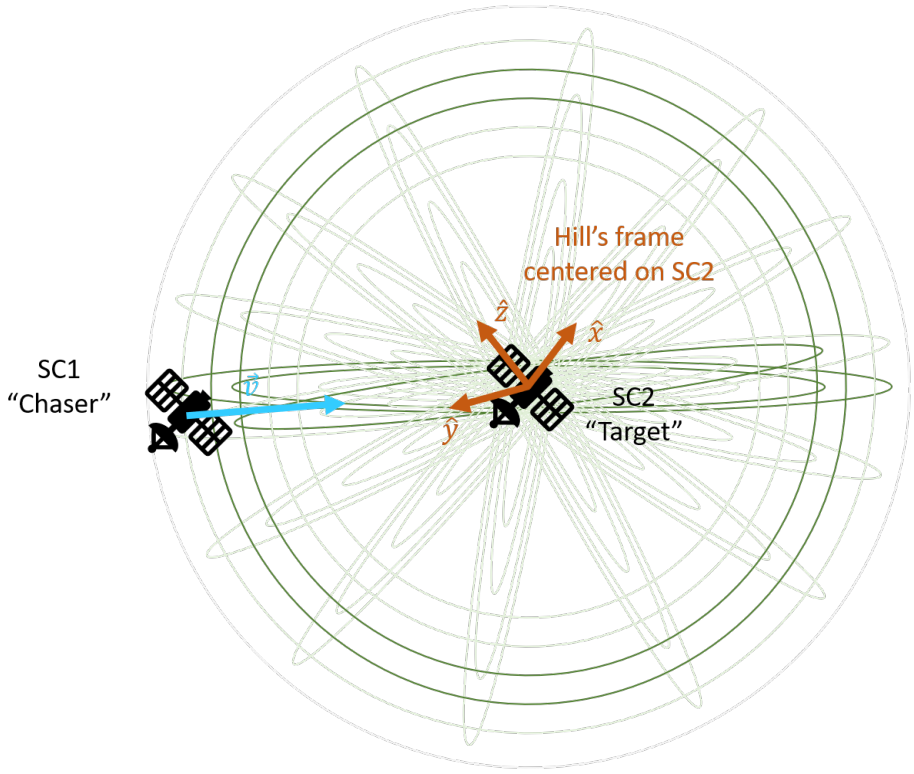


Figure 5.2: Conceptual depiction of a "chaser" spacecraft operating around a "target" spacecraft surrounded by natural motion trajectory ellipses that serve as candidate escape trajectories.

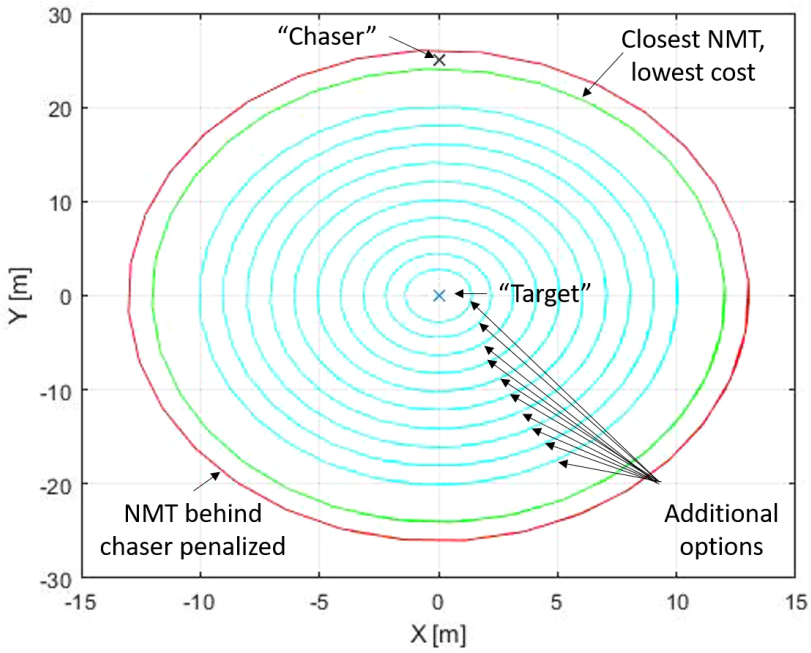


Figure 5.3: Hill's reference frame with chaser and target satellites and relative positions.

other, Hill's reference frame and linearized Clohessy-Wiltshire may be used to describe the system dynamics.

5.1.1 Hill's Frame

For relative motion dynamics, a satellite-centered reference frame is used. Developed in the 1870s to describe the orbit of one body about each other [240], the reference frame is also sometimes referred to as the local-vertical local-horizontal (LVLH) or radial-tangential normal (RTN) frame when centered on the earth. The frame, depicted in Fig. 5.4 is centered on the "target" (also called "chief") satellite, with:

- x-axis $\hat{x}$ (or $\hat{e}_R$ for "radial direction") that points outwards from Earth's center ($\hat{x} = \hat{e}_R = \frac{\vec{r}}{r}$),
- y-axis $\hat{y}$ (or $\hat{e}_T$ for "tangential direction") points in the direction of the velocity vector ($\hat{y} = \hat{e}_T = \hat{e}_N \times \vec{\hat{e}}_R$) of the target satellite, and
- z-axis $\hat{z}$ (or $\hat{e}_N$ for "normal direction") points orthogonally out of the orbital plane ($\hat{z} = \hat{e}_N = \frac{\vec{r}\times\vec{v}}{||\vec{r}\times\vec{v}||}$, where $\vec{r}$ and $\vec{v}$ are the position and velocity vectors of the target satellite).

The x and y axes define the orbital plane of the satellite at the center of Hill's frame, while z is referred to as out of plane.

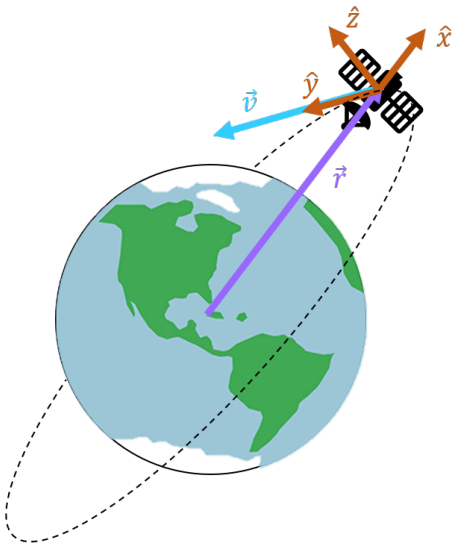


Figure 5.4: Hill's reference frame.

5.1.2 Clohessy-Wiltshire Equations

The relative motion of a "chaser" (or "deputy") spacecraft to a "target" (or chief) spacecraft, is described by Eqn. 5.1:

$$\ddot{\vec{R}} = -\mu \frac{\vec{R}}{|\vec{R}|^3} + \frac{1}{m_c}\vec{F} \tag{5.1}$$

where μ is Earth's gravitational parameter ($(3.986004418 \pm 0.000000008) \times 10^{14} \frac{\mathrm{m}^3}{\mathrm{s}^2}$), $\vec{r}_H$ is the relative position of the chaser satellite in Hill's frame, $\vec{r}$ is the position of the target satellite with respect to the center of the Earth, $\vec{R}$ is the relative position of the chaser satellite with respect to the center of Earth ($\vec{R} = \vec{r} + \vec{r}_H$), m_c is the mass of the chaser satellite, and $\vec{F}$ is a vector of external forces.

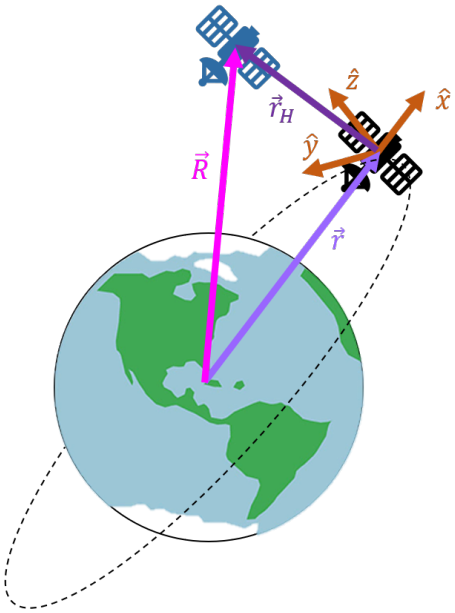


Figure 5.5: Hill's reference frame with chaser and target satellites and relative positions.

These dynamics were linearized by Clohessy and Wiltshire in 1960 in [241] by assuming the spacecraft are in a circular orbit and that the distance between the spacecraft is much smaller than the distance from either spacecraft to the center of the orbit ($\vec{r}_H << \vec{R}$). The linearized Clohessy-Wiltshire dynamics in Hill's frame are:

$$\begin{aligned} \ddot{x} &= 2n\dot{y} + 3n^2 x + \frac{F_x}{m_c} \\ \ddot{y} &= -2n\dot{x} + \frac{F_y}{m_c} \\ \ddot{z} &= -n^2 z + \frac{F_z}{m_c} \end{aligned} \tag{5.2}$$

where x, y, and z are Cartesian positions (in this notation the dotted variables are derivatives with respect to time, i.e. $\dot{x} = \frac{dx}{dt}$ and $\ddot{x} = \frac{d(dx)}{d(dt)}$); F_x, F_y, and F_z are thrust force applied by chaser spacecraft; a is length of the semi-major axis of the target's orbit; and n is satellite mean motion ($n = \sqrt{\mu/a^3}$).

In state space form, the equations may be described as:

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1/m_c & 0 & 0 \\ 0 & 1/m_c & 0 \\ 0 & 0 & 1/m_c \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \tag{5.3}$$

As can be seen from the equations, the in plane dynamics (x and y) are decoupled from the out of plane dynamics (z). In addition, the in plane dynamics are unstable with two eigenvalues at the origin and two at $\pm n_j$. The out of plane dynamics are stable with two eigenvalues at $\pm nj$. The in plane dynamics are completely controllable from F_y but not controllable from F_x, and the out of plane dynamics are controllable from F_z.

5.1.3 Natural Motion Trajectories

Within the linearized Clohessy-Wiltshire equations $\dot{X} = AX + BU$, natural motion trajectories (NMTs) describe the motion of the satellite with no control input ($U = 0$ and $\dot{X} = AX$). Open NMTs, denoted $\bar{\mathcal{N}}$, may be a non-periodic line segment, helix (traveling ellipse), or spiral. Closed NMTs, denoted $\mathcal{N}$ may be ellipses, periodic line segments or stationary points. Examples of closed NMTs with randomly selected initial conditions are shown in Fig. 5.6

To create any closed NMT, one must choose an initial state $X_0 = X(0)$ such that $\dot{y}(0) = -2nx(0)$ where n is the mean motion of the circular orbital trajectory. In addition to this criteria, each closed NMT has specific additional criteria. For the closed NMT to be a stationary point, $x(0) = z(0) = \dot{x}(0) = \dot{z}(0) = 0$, and $y(0) = y_0$ may be any arbitrary value reasonably close (5-10 kilometers or less) to the target spacecraft. To create a periodic line segment $x(0) = \dot{x}(0) = \dot{y}(0) =$

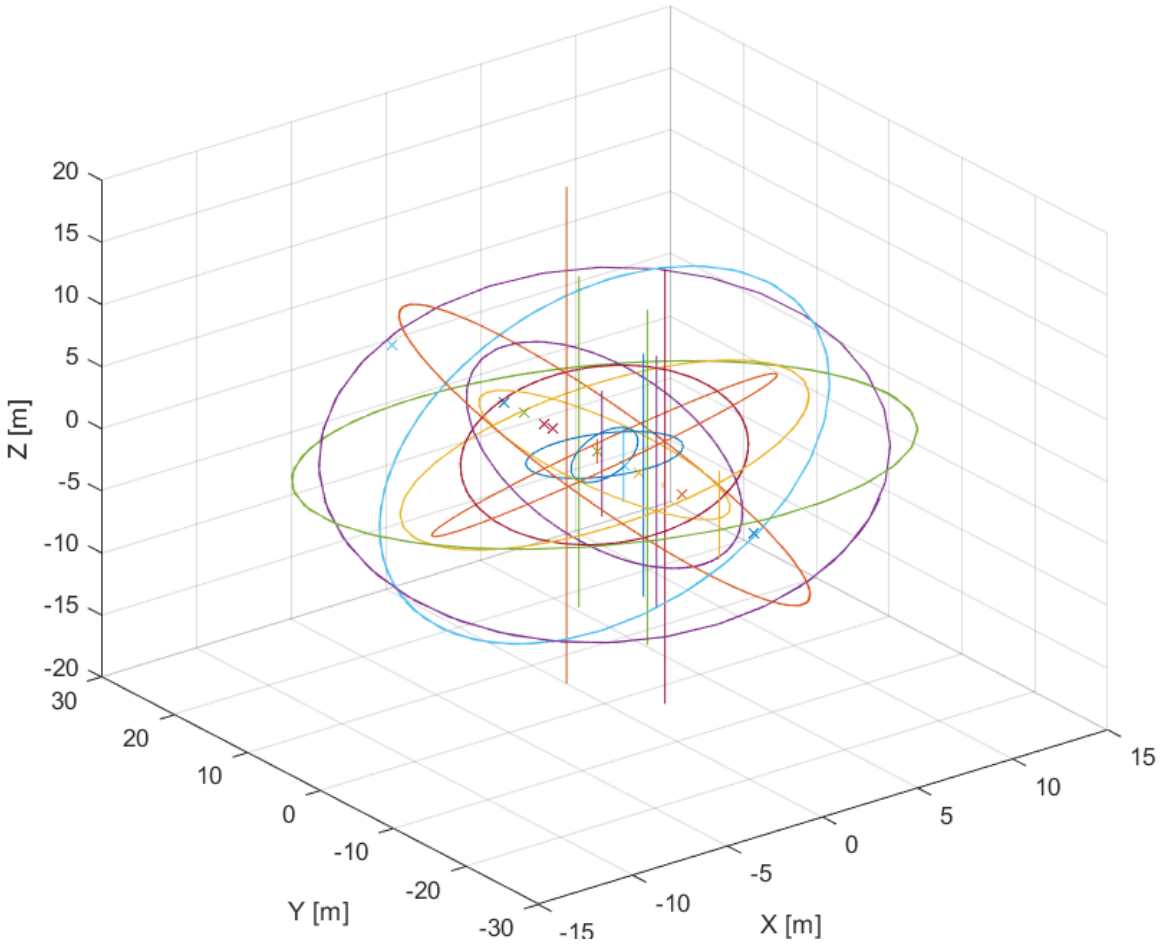


Figure 5.6: Examples of closed natural motion trajectories in Hill's relative motion reference frame including ellipses, periodic line segments, and stationary points (marked with x's).

0, $z(0) = c\sin\psi$, $\dot{z}(0) = nc\cos\psi$, and $y(0) = y_0$ may be any arbitrary value reasonably close to the target, where c is the magnitude of the oscillation (1/2 the length of the line segment) and n is the mean motion as previously defined. To create an ellipse NMT $y(0) = \frac{2}{n}\dot{x}(0)$, for arbitrary but close values of $x(0)$, $z(0)$, $\dot{x}(0)$ and $\dot{z}(0)$.

5.2 Requirements Elicitation

In this section, formal design specifications are elicited where applicable to describe the behavior of the components in the form of transitions systems and formal requirements are elicited that describe constraints on the system design. The design specification and requirements elicitation process was iterative, starting with an initial set of requirements inspired by aircraft collision avoidance requirements that were manually converted to appropriate spacecraft collision avoidance requirements,

and supplemented with additional requirements informed by a literature search and hazard analysis. While introduced in Chapter 2, and discussed in Chapter 4 it is worth noting again for clarity that all formal statements are in ptLTL where $\bigcirc^{-1}$ indicates that the formal statement is true for the previous timestep and $\Box$ indicates that the formal statement is invariant, i.e. from a past time perspective, it is historically (always) true and has been true for all timesteps up to and including the current timestep.

5.2.1 Requirements from Standards and Guidance

No requirements or safety properties for automatic spacecraft maneuvering currently exist in standards or regulations. This is in part due to the often proprietary or classified nature of spacecraft programs. There is also not currently a space traffic management organization that governs maneuvering of vehicles in space analogous to air traffic management systems that govern commercial aircraft operations. However, several entities are involved in the development and approval of a satellite. The largest regulation source is arguably the Federal Communications Commission (FCC), which governs the radio frequencies satellites operate in and reviews applications from satellite owners and operators in the United States for US market access licenses. The Code of Federal Regulations (CFR), Title 47, Part 25 Satellite Communications [45], has specific regulations governing debris mitigation that generally address geostationary orbits where the majority of the valuable spacecraft reside; however, applications for large numbers of non-geostationary systems has prompted debate over regulations and guidance for satellites and constellations of satellites operating in other than geostationary orbit [242]. The Federal Aviation Administration (FAA) is primarily concerned with launch safety. The development of a satellite is managed by the owner or operator of that satellite. The NASA Systems Engineering Handbook [43] and USAF Space and Missile Systems Center (SMC) Systems Engineering Primer & Handbook [42] provide complementary guidance on the development of spacecraft in a rigorous system engineering process, but these are also not a source for on-orbit maneuver requirements.

5.2.2 Requirements from Spacecraft Collision Avoidance Literature

To supplement general collision avoidance requirements from the aircraft domain, a literature search on spacecraft collision avoidance strategies was conducted. The collision avoidance literature is

largely a variety of optimization approaches and the constraints provide an excellent source of requirements. A variety of long term and short-term approaches were examined. Two-point boundary value problems that constrain initial and final positions of the spacecraft are common, but excluded here in favor of more general constraints. While some approaches to collision avoidance were nonlinear [243, 244, 245], many involved linear relative motion dynamics [246, 221, 247]. The following constraints were identified from the literature:

- *Bounded control input* $\mathbf{u}$*, or thrust, or change in velocity* $\Delta\mathbf{v}$: Some form of this constraint is the most common in the literature and is used to represent the limits of how much the spacecraft state can be changed by the onboard propulsion system. In some cases this is presented as a generalized constraint, such as

$$|U_k|_\infty \leq u_{max} \tag{5.4}$$

  where U_k is a velocity impulse in a discrete-time spaceraft model and u_{max} is a maximum value such as 10 m/s [233]. In some cases a nonlinear control constraint is linearized. For example, in [245] a nonlinear constraint on control $u_x^2 + u_y^2 \leq u_{max}^2$ is linearized as

$$-\bar{\gamma}u_{max} \leq u_x(k) \leq \bar{\gamma}u_{max}, -\bar{\gamma}u_{max} \leq u_y(k) \leq \bar{\gamma}u_{max}, \bar{\gamma} \in \left[\frac{1}{\sqrt{2}}, 1\right] \tag{5.5}$$

  where $\bar{\gamma}$ is chosen to reduce conservatism ($\bar{\gamma} = 1$ implies all thrust acts in on direction, while $\bar{\gamma} = \frac{1}{\sqrt{2}}$ implies thrust is evenly distributed between x and y directions, and the actual distribution is somewhere in between). Similar control magnitude limits are presented in [248, 247]. Maximum thrust is also a common constraint, with values such as 5 N [221] or 10 N in each axis [249]. In other cases, the change in velocity $\Delta\mathbf{v}$ is limited [246, 250, 251], in some cases to just a single value in a positive or negative direction [252].

- *Minimum separation distance*: This constraint describes some minimum distance between the space objects that must be maintained at all times. Variations consider intersections of ellipses described by standard deviation in covariance matrices [252]. In [253], a minimum

distance constraint is described as:

$$||\mathbf{r}(t) - \mathbf{r}_O(t)|| \geq d_{min}(t) \tag{5.6}$$

where $\mathbf{r}(t)$ and $\mathbf{r}_O(t)$ are the positions of the spacecraft and obstacle and $d_{min}(t)$ is the desired minimum distance between these objects. A similar constraint is presented in [246]. This minimum distance constraint can alternatively be represented as a penalty function [253] in the optimization of a trajectory.

- *Thrust direction limits*: Constraints on thrust direction may be imposed to avoid firing a thruster into the object the spacecraft is trying to avoid. This constraint is highly dependent on geometry, though an example of avoiding an in-orbital track target is given in [233]:

$$\Delta\dot{y} \leq \mu e^{-\beta k}, \mu > 0, \beta > 0 \tag{5.7}$$

where $\Delta\dot{y}$ is the change in velocity in the y-direction.

- *Station keeping constraints*: Depending on the type of orbit, it may be required that a satellite maintain station keeping constraints, i.e. stay within an acceptable range of a desired state $\mathbf{x_j}$ at time $t(j)$. In [246] these constraints us a curvilinear coordinate frame as six inequality constraints for N timesteps:

$$-x_B \leq x_j \leq x_B, \ -y_B \leq y_j \leq y_B, \ -z_B \leq z_j \leq z_B, \ j \in [1, N]. \tag{5.8}$$

- *Avoiding obstacles with a tangent line or hyperplane*: An alternative to a minimum distance requirement is defining obstacle avoidance with a tangent line or hyperplane. In [246] a linear constraint is identified as:

$$\alpha_j y_j + \beta_j z_j \geq r \forall j \in [1, N] \tag{5.9}$$

where $\alpha_j = cos\theta_j$, $\beta_j = \sin\theta_j$, and $\theta_j = \theta_0 + nt_j$, as pictured in Fig. 5.7.

In [233], a hyperplane is defined that separates the obstacle from the spacecraft and used in a

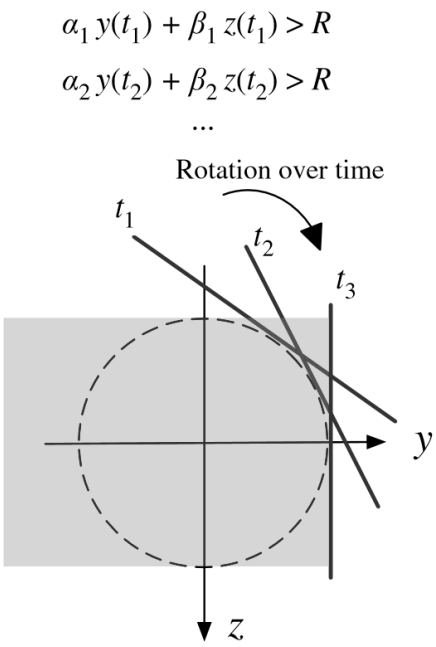


Figure 5.7: Separation constraint method based on tangent line [246].

constraint that

$$n_k^T X_{p,k} \geq n_k^T r_{c,k} \tag{5.10}$$

where n_k is a vector normal to the hyperplane, $r_{c,k}$ is a point on the boundary of the uncertainty ellipsoid of the obstacle, and $X_{p,k}$ is the position of the spacecraft at time k.

- *Minimum drift rate*: Along-track drift of a satellite in a relative motion orbit results in elliptical motion around an object if the along-track drift is zero, or corkscrew motion for non-zero drift [246]. Keeping the drift below a minimal value ensures the spacecraft will not move in a corkscrew motion into the obstacle in a finite period of time.

- *Line of sight constraints*: While a more important requirement for proximity operations and docking, constraints on line of sight [245, 233] between a sensor on the spacecraft and the obstacle being avoided enable higher accuracy, closed-loop control during the avoidance maneuver.

- *Spacecraft size*: Because spacecraft can differ considerably in size and shape, and even change size by deploying antennas, booms, solar panels, and other appendages, it is im-

portant to incorporate the size of the spacecraft in the collision avoidance maneuver. Some approaches to this include using the longest axis of a spacecraft at a spacecraft radius [243] and creating a buffer ellipse around the obstacle spacecraft [221].

- *Minimum probability of collision* P_c: Probability of collision remains a standard metric for determining when it is appropriate to maneuver. When the probability of collision is over a threshold such as 10^{-4}, it may be grounds to maneuver to ensure safe separation [247, 252].

- *Time of closest approach (TCA)*: The time of closest approach between the two space objects and the available thrust onboard provide a lower limit to how close a maneuver can occur to the predicted collision time. In [252], a limit of 10^{-3} to 10^{-2} meters per second in Δv restricts collision avoidance maneuvers to occurring far from TCA.

- *Collision Geometry*: The geometry of the collision scenario can play an important role in collision avoidance maneuvers. Determining whether the collision is frontal or lateral is important in [252], as their interrelation could cause a maneuver that increases separation in the frontal direction decrease separation in the lateral direction and vice versa.

- *Post maneuver trajectory prediction*: Considering the trajectory of the spacecraft after the maneuver and whether the maneuver will result in increased collision risk with other objects is an important consideration in maneuver selection [252].

Additional practical considerations on maneuver time and communication may factor into the automatic collision avoidance maneuver controller design. In practice, a spacecraft maneuver involves first reorienting the spacecraft to align the primary thruster in the desired direction, and then waiting for an optimal time to command a burn [238, 239]. In some cases with a small notification time before the maneuver, it may be assumed that there is not time to reorient the spacecraft, and a burn is conducted from the current orientation [238, 254]. In addition, there is a cost benefit analysis between the amount of propellant consumed and the estimated probability of collision [255]. Finally, communication and computation concerns such as signal routing, real time feedback, and real time communication can determine success of a maneuver [256].

5.2.3 Requirements from Hazard Analysis

Following the approaches listed in Section 2.5, accidents, hazards, and safety constraints are identified for spacecraft automatic maneuvers. While the intention is to use these as a guideline for developing safety requirements for a spacecraft automatic collision avoidance system, these accidents, hazards, and constraints are generic and may be applied to the development of any spacecraft control system that maneuvers automatically. Accident and hazard identification was primarily accomplished by interviewing stakeholders.

Accidents

As described earlier, an accident is defined as an undesired or unplanned event that results in a loss. The loss is application specific and could be a variety of challenges such as the loss of human life or health, damage to property, environmental pollution, or mission loss. This definition of a loss is not consistent across sources and may lead to confusion. For instance, Air Force Instruction 51-503: Aerospace and Ground Accident Investigations [257] defines an accident as an unplanned occurrence, mishap or series of occurrences, that results in damage or injury and meets Class A, B, C, or D mishap reporting criteria.

Two primary *accidents* related to automatic or autonomous spacecraft maneuvering were identified, as follows:

[A1] Spacecraft is damaged or destroyed.

[A2] Spacecraft is unable to complete its mission.

These accidents are obviously not mutually exclusive.

Hazards

With the accidents in mind, *hazards* were identified. MIL-STD-882E, Department of Defense Standard Practice System Safety [72], defines a hazard as "A real or potential condition that could lead to an unplanned event or series of events (i.e. mishap) resulting in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment." In STAMP and STPA a hazard is defined as a system state or set of conditions that in a worst-case environment, will

lead to an accident. In both definitions, a hazard is a specific condition that could lead to a mishap or accident. Each hazard is traced to the accident(s) they may cause by indicating those accident numbers in parentheses at the end of the hazard. Rationale is provided for each hazard to provide additional context and examples. The hazards are written in priority order, but this priority may be very subjective depending on the spacecraft and mission.

[H1] Spacecraft maneuver causes ground communication loss (A1, A2).

Rationale: A "loss of spacecraft signal" is addressed in a standard fault protection type called "Command Loss Response" at JPL [258]. While this communication loss can be caused by a variety of factors, erroneous spacecraft attitude (pointing error) is focused on here, as it can result from incorrect automatic or autonomous control operations. This hazard is listed first because a ground communication loss prevents operator intervention that may be critical to prevent damage or mission loss. This hazard is further scoped to only referring to maneuvers that prevent ground communication that would otherwise be available and scheduled. Many spacecraft missions do not have constant ground communication and may be intentionally out of contact multiple times a day. This could occur because the spacecraft does not have line of sight to a ground station in the operator's network for a portion of its orbit, or communication with that spacecraft is a lower priority than spacecraft that share the same ground station. This hazard only applies in situations where the spacecraft has line of sight to a ground station in the operator's network and has priority to use it over another spacecraft. Communication requires that the ground station is in the field of view of the spacecraft's communication antenna.

[H2] Spacecraft is on a collision course with another spacecraft or debris (A1).

Rationale: Spacecraft collisions could not only lead to loss of the spacecraft and mission, they could lead to loss of other spacecraft as well as the creation of debris that threatens other space operations. Several examples of collision events include collisions during rendezvous and proximity operations, as well as confirmed random, accidental collisions:

1. 23 December 1991: (but not recognized until 2005) collision of a defunct Cosmos navigation satellite with a piece of debris from another Cosmos satellite [259];

2. 1994: collision of the Soyuz TM-17 ferry spacecraft with the MIR space station, resulting in only minor damage [234];
3. 24 July 1996: collision of the French Ceris Satellite with a fragment of Ariane-1 H-10 upper rocket stage [260];
4. 2001: an 800kg, 2-meter diameter cylindrical Russian satellite launched in 1998 was struck by Cosmos 926 debris [261];
5. 1997: collision of the Progress M-34 spacecraft with Mir, causing damage to MIR solar panels, radiators, and a hull puncture [235, 236];
6. 17 January 2005: collision of a U.S. rocket body with a fragment of the third stage of a Chinese launch vehicle [259, 261];
7. 2005: collision of DART with MUBLCOM sending MUBLCOM into a higher orbit with no significant damages [237];
8. February 2009: Iridium 33 and Cosmos-225 Collision [262]; and
9. 22 January 2013: BLITS retroreflector satellite was impacted by a piece of orbital debris [263].

In addition to the collision events listed here, several suspected but unconfirmed orbital debris collisions have occurred, and are excluded here for brevity; however more details may be found in [264, 261].

[H3] Spacecraft maneuver is aggressive enough to cause damage (A1).

Rationale: An aggressive maneuver is one with high translational or rotational acceleration and may cause damage to the structure, payload, or appendages. These accelerations could potentially exceed the safe limits of the spacecraft structure, payload (which may be a sensitive instrument), or one of many possible appendages and deployables like solar panels, antennas, booms, and tethers. An example of this occurred in April 2016 when a combination of a design flaw in reaction wheel rotation direction and bad settings for rocket firings caused the Japanese Hitomi X-ray observatory to spin out of control, shedding portions of its solar panels or deployable telescope as a result [230].

[H4] Spacecraft maneuver leads to uncontrollable state. (A1, A2).

Rationale: It is possible for a maneuver or successive maneuvers to lead to a loss of the ability to control a spacecraft. The Japanese Hitomi attitude failures present a convenient example for this hazard because eventually the spacecraft was spinning too fast to control [230].

[H5] Spacecraft generates insufficient power to maintain operations (A1, A2).

Rationale: In order to generate power, the spacecraft's solar panels must be pointed towards the sun for a duration of time. If a spacecraft is unable to point the solar panels at the sun, or is unable to maintain that attitude, the solar panels may not provide sufficient power to keep batteries charged. This hazard is listed fifth because while it may not directly cause damage or mission loss, many other critical subsystems like communications and controls rely on power. In 1998, a series of attitude anomalies on the Near Earth Asteroid Rendezvous (NEAR) spacecraft nearly caused the loss of the spacecraft, which recovered in a safe mode designed to minimize power usage and maximize solar array output [265]. The Japanese Hitomi attitude failures also present an example for this hazard because eventually the spacecraft was spinning too fast for the solar panels to sustain the satellite's battery [230]. There is also some overlap with H5 and H3 in cases where excessive acceleration leads to a loss of the solar panels.

[H6] Spacecraft loses data transfer with the ground (A2).

Rationale: Data transfer is similar to general communication requirements but features higher bandwidth requirements and often tighter antenna field of view pointing requirements. A loss of data transfer ability could lead to a loss of mission data, this is especially true when onboard memory is often very limited. If the onboard memory fills before the data can be transferred down, new data may not be saved, or older data will have to be deleted before it can be downlinked to the ground for more permanent storage.

[H7] Spacecraft damaged or destroyed by unsafe attitude (A1, A2).

Rationale: Depending on the spacecraft mission or payload, there may be attitudes where it is unsafe to point a spacecraft. The most common example is the solar exclusion angle; where

sensitive instruments must not be pointed too close to the sun to avoid damage. This is the motivation behind exclusion zone guidance methods for spacecraft such as that in [266].

[H8] Spacecraft exceeds unsafe attitude duration (A1, A2).

Rationale: This hazard was documented with three specific conditions in mind: the spacecraft attitude causes excessive heating from sun exposure on a particular component (thermal management), the spacecraft's solar panels are pointed away from the sun for long enough to threaten spacecraft power loss (power management), or spacecraft components are left within a solar exclusion angle long enough to cause damage. Development of a safe mode for the Cassini spacecraft included reaching an attitude that enabled the spacecraft to communicate with the ground, was thermally safe for several days, could be maintained without being overwhelmed by aerodynamic forces curing low altitude flybys, and that gave the star tracker a clear field of view [267]. Many spacecraft have a "safe mode" that may do one or more of the following [258]: minimize power by stopping the current operations, powering down all nonessential functions, configuring hardware in safe states, establishing uplink and downlink communications, reconfiguring antenna, and commanding an attitude that achieves thermal safety and solar panel charging.

[H9] Spacecraft expends excess fuel (A1, A2).

Rationale: Fuel is a limited resource on spacecraft and could lead to mission or spacecraft loss when depleted. For example, during the 1998 NEAR spacecraft anomaly, nearly 29 kg of fuel (corresponding to 96 m/s of delta-v) were burned in thousands of thruster fires [265]. After recovering, the spacecraft had barely enough fuel to conduct the original mission, with little to no margin for additional error.

[H10] Spacecraft actuation strategy causes excessive wear or damage to actuators (A1, A2).

Rationale: This hazard was documented with reaction wheel actuators in mind. Reaction wheel actuators control spacecraft attitude by spinning to exchange momentum with the spacecraft. Reaction wheels can become saturated when commanded to their limits and can undergo wear if kept near their limits for extended periods of time. Loss of two reaction wheels occurred in the Kepler Space Telescope mission. After the first wheel was lost, the

wheel vendor recommended keeping the wheel speeds below 300 revolutions per minute to preserve functionality of the remaining wheels for as long as possible [268].

Safety Constraints

MIL-STD-882E [208.2.1(h)] [72] requires a list of constraints that can be implemented to reduce the risk level of hazards. In the STPA process, a safety constraint is a capability that will prevent a hazard from occurring. The safety constraints corresponding to each of the hazards are listed here. These safety constraints are traced to their corresponding hazards and used to generate requirements formal requirements. Safety constraints here are scoped to only those relating to satellite control.

Each of the formalized safety constraints in this section describe the requirements of the system; they define what property that should be true. Design specifications that describe how the system behaves to achieve those requirements may use a variety of approaches and are left open ended.

[C1] Spacecraft shall maintain attitude requirements for communication with ground station (H1). In order to formally specify this constraint as a requirement in ptLTL, first a few definitions are needed. As depicted in Fig. 5.8, the solid angle field of view (FOV) α is the total angle observable to the sensor/transmitter, the boresight is the centerline of the sensor/transmitter FOV, the target zenith angle θ_s is the angle between the sensor/transmitter and surface normal, and the fixation angle θ_R is the angle between the sensor/transmitter boresight and the receiver.

For communication, a reasonable assumption is that the antenna's solid angle field of view (FOV) α is approximately 70° [269]. Let the atomic proposition $AP = \{LOS_{COMM}, FOV_{COMM}\}$ be the set of events where LOS_{COMM} indicates the ground station is in light of sight of the satellite when the zenith angle θ_s is less than or equal to 90° ($\theta_s \leq 90°$), as depicted in Fig. 5.9a and FOV_{COMM} indicates that the fixation angle θ_R is smaller than half the antenna solid view FOV α ($\theta_R \leq \alpha$) as depicted in 5.9b. The trace of the system trajectory

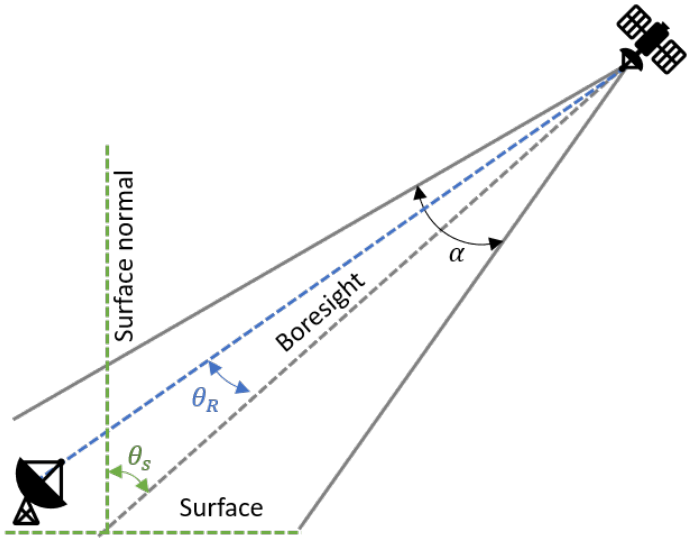


Figure 5.8: Diagram of solid angles, boresight, zenith angle, and fixation angle for a satellite communication antenna.

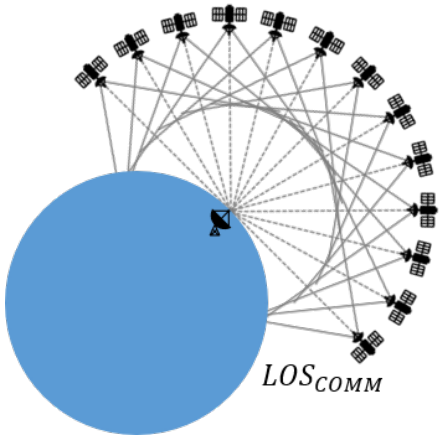
is then given by the labeling function $L(\theta_s, \theta_R) : \mathbb{R}^2 \to 2^{AP}$:

$$L(\theta_s, \theta_R) = \begin{cases} \emptyset, & \text{if } \theta_s > \frac{\pi}{2} \text{ and } \theta_R > \frac{\alpha}{2} \\ FOV_{COMM} & \text{if } \theta_s > \frac{\pi}{2} \text{ and } \theta_R \leq \frac{\alpha}{2} \\ LOS_{COMM} & \text{if } \theta_s \leq \frac{\pi}{2} \text{ and } \theta_R > \frac{\alpha}{2} \\ LOS_{COMM} \wedge FOV_{COMM} & \text{if } \theta_s \leq \frac{\pi}{2} \text{ and } \theta_R \leq \frac{\alpha}{2}. \end{cases} \tag{5.11}$$

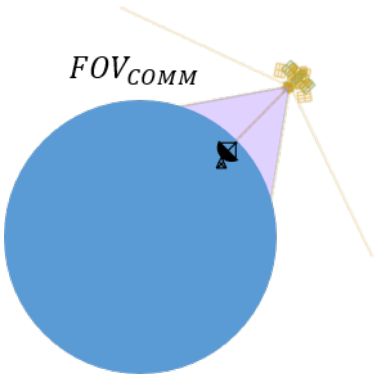
Then the safety requirement becomes:

$$\varphi_{C1} = \Box scheduled_{COMM} \implies (LOS_{COMM} \wedge FOV_{COMM}). \tag{5.12}$$

Further refinement on this requirement might include temporal considerations, such as ensuring proper orientation for some time before the scheduled communication window to sometime after. A design specification should be created to ensure the spacecraft attitude controller maintains the communication safety constraint requirement φ_{C1}. There is an assumption with this requirement that scheduled communication only takes place when the satellite is within the line of sight of the ground station LOS_{GS}, and in the case of a ground station with a limited field of view receiver, that the satellite is within the ground station's field of view



(a) Examples of satellites with line of sight to a ground station, which occurs when the angle between the communication transmitter on the satellite and the surface normal to the Earth's surface where the ground station is located (the zenith angle θ_s) is less than or equal to 90°.



(b) Examples of satellite orientations where the angle between the communication transmitter boresight and the ground station (the fixation angle θ_R) is less than half the antenna solid angle field of view (FOV) α. The two yellow satellites represent extremes of the possible orientations, where the boresight is within the purple shaded region.

Figure 5.9: Depictions of satellite communication with line of site or field of view to a ground station.

FOV_{GS}. This is a constraint that should be checked on the scheduler.

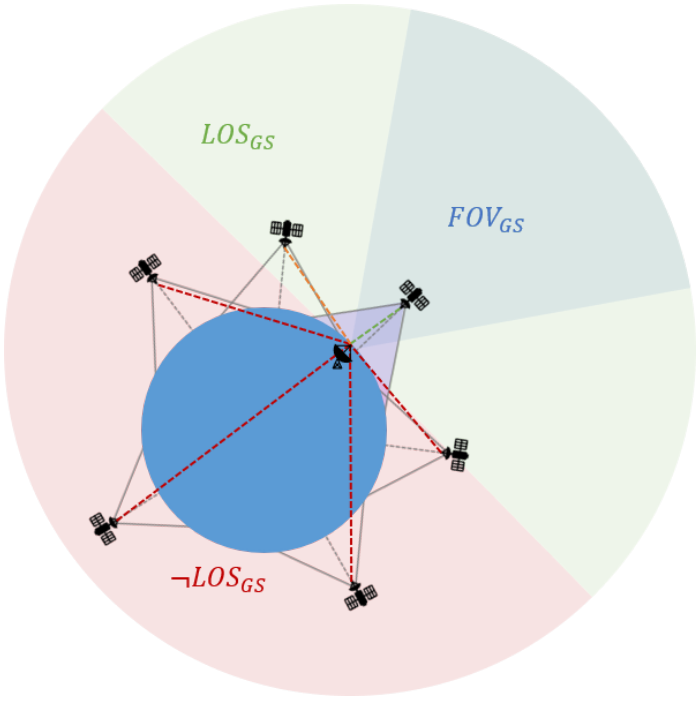


Figure 5.10: Notional depiction of regions where the Ground Station's receiver antenna has line of sight to the ground station (LOS_{GS}) , does not have line of sight ($\neg LOS_{COMM}$), and when the satellite's antenna is in field of view of the ground station (FOV).

[C2] Spacecraft maneuvers shall maintain safe separation with another spacecraft or debris (H2).

There are several ways to represent safe separation, including but not limited to the following:

- The spacecraft are safely separated if the probability of collision P_c is less than some maximum $P_{c_{max}}$, such as 10^{-4} used in modern spacecraft collision detection and avoidance approaches [20, 21, 22, 23].

$$\varphi_{C2_{P_c}} = \Box(P_c \leq P_{c_{max}}). \tag{5.13}$$

- The spacecraft are safely separated if the distance in the relative motion Hill's frame $||\vec{r}_H||$ is greater than some separation distance r_s. This is representative of collision detection systems that warn when the closest simulated miss distance is less than 200 meters, 300 meters, or 1 kilometer (depending on organization) away.

$$\varphi_{C2_{r_s}} = \Box||\vec{r}_H|| > r_s. \tag{5.14}$$

- The spacecraft are safely separated if the distance in the relative motion Hill's frame $||\vec{r}_H||$ is greater than some separation distance r_s and the relative velocity $||\vec{v}_H||$ is below a threshold v_s, or the spacecraft are moving away from one another. Recall that the inner product (or dot product) can indicate relative motion of the of the spacecraft, as depicted in 5.11:

$$\vec{v}_H^T\vec{r}_H = \vec{v}_H \cdot \vec{r}_H = ||\vec{v}_H||||\vec{r}_H||\cos\theta = \begin{cases} > 0 \implies & \text{moving away} \\ < 0 \implies & \text{moving toward} \\ = 0 \implies & \text{moving orthogonally.} \end{cases} \tag{5.15}$$

This constraint then becomes:

$$\varphi_{C2_{rv}} = \Box(||\vec{r}_H|| > r_s) \wedge \Big((||\vec{v}_H|| < v_s) \vee (\vec{v}_H^T\vec{r}_H \geq 0)\Big). \tag{5.16}$$

[C2alt] Spacecraft maneuvers shall safely approach spacecraft during autonomous rendezvous,

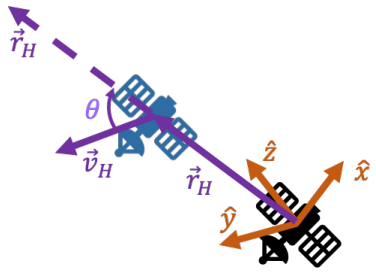


Figure 5.11: Relative position and velocity vectors between two objects in Hill's reference frame.

proximity operations and docking (H2). In cases where spacecraft formation flying or a controlled collision (docking) is intended, alternative criteria may be developed. A variation on this constraint changes the magnitude of the acceptable velocity moving towards the spacecraft based on the distance between the spacecraft, where acceptable relative velocity decreases as the spacecraft distance decreases, as depicted in Fig. 5.12.

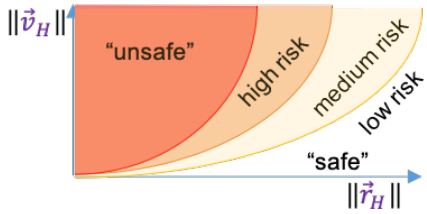


Figure 5.12: Notional depiction of a variable safe relative velocity v_s that decreases as the distance between two spacecraft decreases, where a variable risk level setting adjusts the curves to allow higher velocity closer to.

This concept is similar detecting the difference between aircraft flying in formation and aircraft on a collision course [187]. In the Automatic Air Collision Avoidance System (Auto ACAS) development, a set of formation logic based on range and closure rate as seen in Fig. was used to define whether to inhibit an automatic collision avoidance maneuver. A similar study could be conducted to determine a formation and docking deactivation region for spacecraft.

This concept is notionally borrowed from the idea of using a temporal rather than distance requirement to avoid collisions in the aircraft domain where the time-to-collision

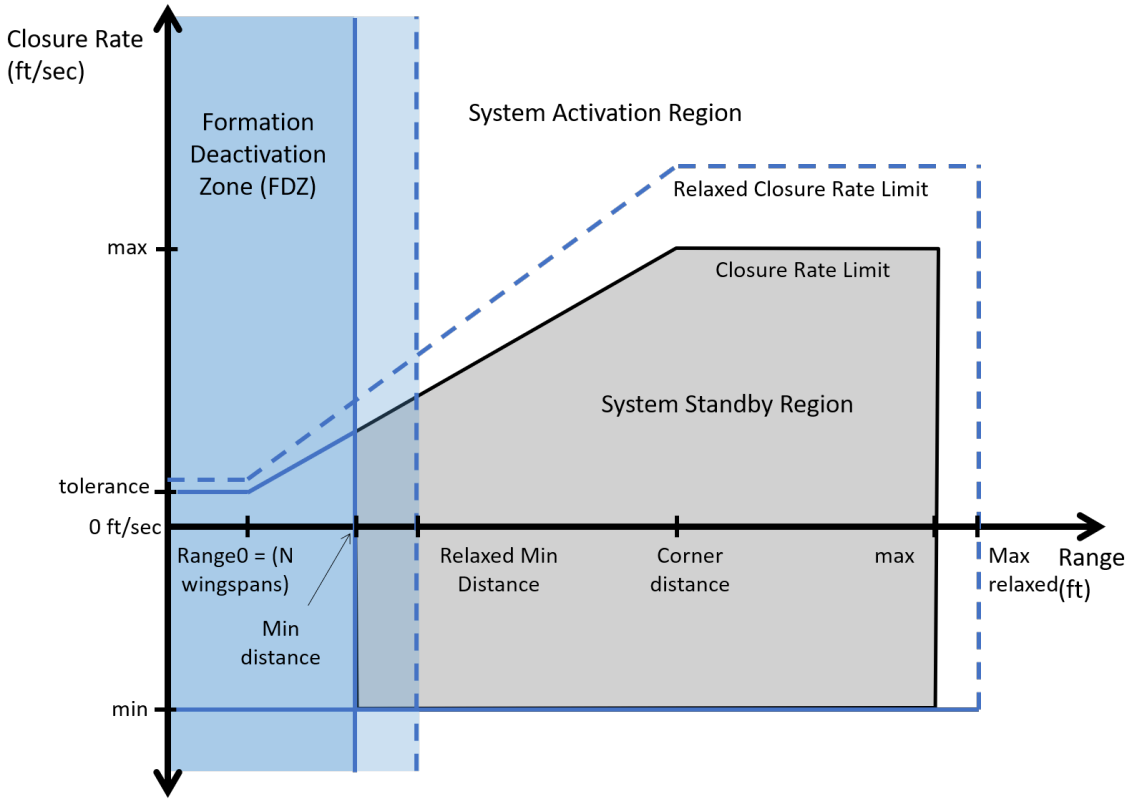


Figure 5.13: Formation flight boundary diagram for the Automatic Air Collision Avoidance (Auto ACAS) program [187].

T_c is the separation distance $||\vec{r}_H||$ over closure velocity $||\vec{v}_H||$ [270]:

$$T_c = \frac{||\vec{r}_H||}{||\vec{v}_H||}. \tag{5.17}$$

However, this function assumes a linear path and in aircraft avoidance maneuver, or in spacecraft relative motion, the path is not linear. A better estimate may be found by the time to collision point T_{cp}, which is the distance from the present position along the trajectory to the collision point d_{cp}, divided by the speed along the trajectory s_{cp} [270]:

$$T_{cp} = \frac{d_{cp}}{s_{cp}}. \tag{5.18}$$

Given the development of a finite set of collision avoidance maneuvers, the T_{cp} could be computed for each maneuver option and used in the decision to engage an automatic collision

avoidance maneuver.

[C3] Spacecraft shall maneuver below acceleration threshold to cause damage (H3).

Spacecraft translational and rotational acceleration limits depend on the limits of the structure of the spacecraft (ex. $\ddot{x}_{str}$), the payload that may vary on satellites that otherwise feature the same components (ex. $\ddot{x}_{pay}$), and variations of limits for special states like the use of deployable antennas or booms (ex. $\ddot{x}_{sp}$). For each axis of translational and rotational acceleration, the minimum and maximum accelerations follow the form $\ddot{x}_{min} = -\min(|\ddot{x}_{str}|, |\ddot{x}_{pay}|, |\ddot{x}_{sp}|)$ and $\ddot{x}_{max} = \min(|\ddot{x}_{str}|, |\ddot{x}_{pay}|, |\ddot{x}_{sp}|)$. This is assuming that the acceleration limits are the same in the positive and negative directions. In the case that the limits are not equal and opposite in each direction, the smallest acceleration in either direction (positive or negative) along the axis is the limit for that direction. Then the constraint becomes:

$$\begin{aligned}\varphi_{C3} =& \Box(\ddot{x}_{min} \leq \ddot{x}) \wedge (\ddot{x} \leq \ddot{x}_{max}) \wedge (\ddot{y}_{min} \leq \ddot{y}) \wedge (\ddot{y} \leq \ddot{y}_{max}) \\ & \wedge (\ddot{z}_{min} \leq \ddot{z}) \wedge (\ddot{z} \leq \ddot{z}_{max}) \wedge (\ddot{\theta}_{1_{min}} \leq \ddot{\theta}_1) \wedge (\ddot{\theta}_1 \leq \ddot{\theta}_{1_{max}}) \\ & \wedge (\ddot{\theta}_{2_{min}} \leq \ddot{\theta}_2) \wedge (\ddot{\theta}_2 \leq \ddot{\theta}_{2_{max}}) \wedge (\ddot{\theta}_{3_{min}} \leq \ddot{\theta}_3) \wedge (\ddot{\theta}_3 \leq \ddot{\theta}_{3_{max}}).\end{aligned} \tag{5.19}$$

Note that all of these constraints are on acceleration of the spacecraft in the spacecraft body frame.

[C4] Spacecraft maneuver shall maintain controllability (H4).

In its simplest form, this constraint states that the state of the spacecraft X_{sc} always remains within the set of controllable states:

$$\varphi_{C4} = \Box X_{sc} \in \mathcal{X}_C. \tag{5.20}$$

Determining the set of controllable states is non-trivial though. A linear time invariant (LTI) system is controllable if for all initial and final states in the set of real numbers ($\forall x_o, x_f \in \mathbb{R}^n$) there exists a control input for a time between 0 and the final time ($\exists u(t), t \in [0, -t_f]$ such that the final state is reached at the final time ($s.t.\ x(t_f) = x_f$). One test for controllability is the controllability matrix: (A, B) is controllable if $\mathcal{C}(A, B) = [B, AB, A^2B, ..., A^{n-1}B]$ is full rank (also $\mathcal{C}(A, b)$ is invertable).

The relative translational motion of a "chaser" (or "deputy") spacecraft to a target (or chief) spacecraft, in linearized Clohessy-Wiltshire dynamics in Hill's frame are:

$$\begin{aligned} \ddot{x} &= 2n\dot{y} + 3n^2 x + \frac{F_x}{m_c} \\ \ddot{y} &= -2n\dot{x} + \frac{F_y}{m_c} \\ \ddot{z} &= -n^2 z + \frac{F_z}{m_c} \end{aligned} \tag{5.21}$$

where x, y, and z are Cartesian positions (in this notation the dotted variables are derivatives with respect to time, i.e. $\dot{x} = \frac{dx}{dt}$ and $\ddot{x} = \frac{d(dx)}{d(dt)}$); F_x, F_y, and F_z are thrust force applied by chaser spacecraft; a is length of the semi-major axis of the target's orbit; and n is satellite mean motion ($n = \sqrt{\mu/a^3}$).

In state space form ($\dot{X} = AX + BU$), the equations may be described as:

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1/m_c & 0 & 0 \\ 0 & 1/m_c & 0 \\ 0 & 0 & 1/m_c \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \tag{5.22}$$

As can be found from the equations, the controllability matrix $\mathcal{C}(A, B) = [B, AB, A^2B, ..., A^{n-1}B]$ is full rank, indicating that it is controllable. In addition, the in-plane dynamics (x and y) are decoupled from the out-of-plane dynamics (z). While the dynamics are controllable, the in-plane dynamics are unstable with two eigenvalues at the origin and two at $\pm nj$. The out-of-plane dynamics are stable with two eigenvalues at $\pm nj$. The in-plane dynamics are completely controllable from F_y but not controllable from F_x, and the out-of-plane dynamics are controllable from F_z.

The spacecraft attitude dynamics are nonlinear and cannot be evaluated using a traditional controllability matrix test. More details on the spacecraft attitude dynamics were described

previously by the authors in [169, 167].

While the system is completely controllable, it is possible for the translational or angular velocity to be so high that it exceeds capabilities of the actuators. One way to deal with this is to place limits on the maximum velocity, much like the acceleration limits in constraint 3. Then the constraint becomes:

$$\varphi_{C4} = \Box(\dot{x} \leq \dot{x}_{max}) \wedge (\dot{y} \leq \dot{y}_{max}) \wedge (\dot{z} \leq \dot{z}_{max}) \wedge (\dot{\theta}_1 \leq \dot{\theta}_{1_{max}}) \wedge (\dot{\theta}_2 \leq \dot{\theta}_{2_{max}}) \wedge (\dot{\theta}_3 \leq \dot{\theta}_{3_{max}}). \tag{5.23}$$

[C5] Spacecraft shall maintain attitude requirements for sufficient power generation (H5).

In the spacecraft domain, solar panels are often used to recharge on-board batteries which provide power to all spacecraft subsystems. Solar panel charging may be triggered in multiple ways. One way is by scheduling charging attitudes as part of the mission plan based on projections of orbital locations and power usage. Let $scheduled_{CHRG}$ be an atomic proposition indicating that charging is scheduled at the present time. Another way to trigger charging is that the depth of discharge DOD of the batteries is above some threshold value DOD_{max} (could also be thought of as the percentage of remaining power below a safety threshold) and a sun safe mode is activated that minimizes power usage and maximizes charging [265] until some charge level is achieved.

For power generation, the solar panels must have a sun incidence angle that is within a range of angles for charging. The power P generated by the solar panels is described by

$$P = P_I I_d \cos\theta_{SI}, \tag{5.24}$$

where P_I is the ideal performance (power generated in watts per square meter), I_d is inherent degradation (nominally 0.677, generally $\in [0.49, 0.88]$), and θ_{SI} is the sun incidence angle (angle between the surface normal $\hat{n}_{SP}$ and sun incident $\hat{r}_{SI}$ unit vectors [269]. The cosine of the sun incidence angle is the cosine loss of power generated compared to a sun pointed normal to the solar panels. The geometry for charging is depicted in Fig. 5.14.

The constraint then becomes that when the depth of discharge is above the max threshold, or

when charging is scheduled, then the spacecraft should be have a sun incidence angle smaller than the angle required for charging:

$$\varphi_{C5} = \square((DOD > DOD_{max}) \vee scheduled_{CHRG}) \implies (\theta_{SI} \leq \theta_{CHRG}). \tag{5.25}$$

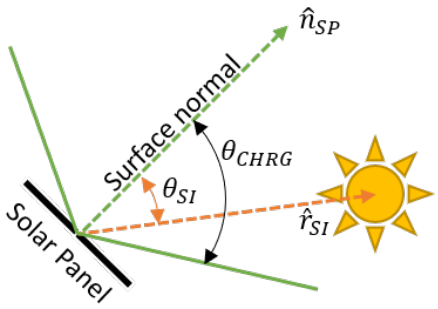


Figure 5.14: Relative position and velocity vectors between two objects in Hill's reference frame.

[C6] Spacecraft shall maintain attitude requirements for data transfer with ground station (H6).

This constraint is very similar [C1], except that the antenna's solid angle FOV for data transfer α_{DATA} is tighter than that required for communications. For data transfer, a reasonable assumption is that α_{DATA} is approximately 40° [269]. Let the atomic proposition AP=$\{LOS_{DATA}, FOV_{DATA}\}$ be the set of events where LOS_{DATA} indicates the ground station is in line of sight of the satellite when the zenith angle θ_s is less than or equal to 90° ($\theta_s \leq 90°$), and FOV_{DATA} indicates that the fixation angle θ_R is smaller than half α_{DATA} ($\theta_R \leq \frac{\alpha_{DATA}}{2}$). The trace of the system trajectory is then given by the labeling function $L(\theta_s, \theta_R) : \mathbb{R}^2 \to 2^{AP}$:

$$L(\theta_s, \theta_R) = \begin{cases} \emptyset, & \text{if } \theta_s > \frac{\pi}{2} \text{ and } \theta_R > \frac{\alpha_{DATA}}{2} \\ FOV_{DATA} & \text{if } \theta_s > \frac{\pi}{2} \text{ and } \theta_R \leq \frac{\alpha_{DATA}}{2} \\ LOS_{DATA} & \text{if } \theta_s \leq \frac{\pi}{2} \text{ and } \theta_R > \frac{\alpha_{DATA}}{2} \\ LOS_{DATA} \wedge FOV_{DATA} & \text{if } \theta_s \leq \frac{\pi}{2} \text{ and } \theta_R \leq \frac{\alpha_{DATA}}{2}. \end{cases} \tag{5.26}$$

Then the safety requirement becomes:

$$\varphi_{C6} = \Box scheduled_{DATA} \implies (LOS_{DATA} \wedge FOV_{DATA}). \tag{5.27}$$

Like [C1], several possible refinements may be made on this constraint.

[C7] Spacecraft shall adhere to attitude keep out zone geometries (H7).

The spacecraft adheres to attitude exclusion zone geometries when the angle between the sensor boresight and the direction of exclusion $\hat{r}$, denoted θ_{EZ} is less than half the sensor field of view α plus a safety buffer angle β, as depicted in Fig.5.15. Then the safety requirement becomes:

$$\varphi_{C7} = \Box \theta_{EZ} > \frac{\alpha}{2} + \beta \tag{5.28}$$

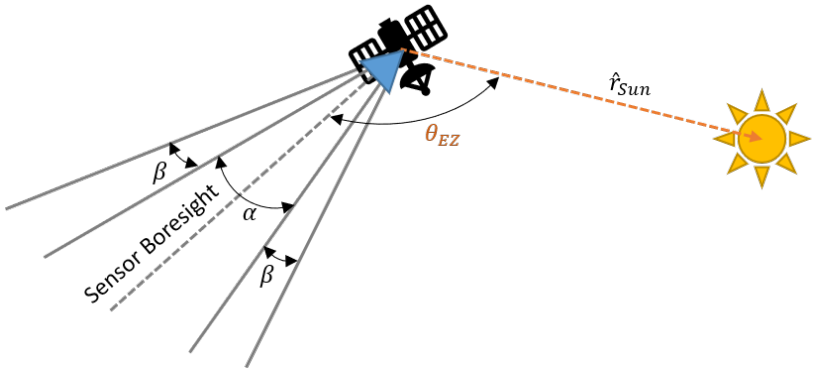


Figure 5.15: Notional depiction of attitude keep out (or exclusion) zone geometry.

[C8] Spacecraft shall limit duration of unfavorable attitudes (H8).

The primary difference between [C7] and [C8] is that in [C7] it is never safe to point in an exclusion zone direction, while in [C8] it is acceptable to point in an exclusion zone direction as long as it is for a limited duration. Let $AP = \{LOS_S, T_S\}$ be the set of events where LOS_S indicates that the spacecraft is in a favorable and safe attitude, and T_S indicates that if the system has been pointing in an unfavorable duration, it has been for a safe duration. One way to define the safe line of sight is using the method in [C7]. Another way to define the safe line of sight is with a vector X_{US} aligned with the sensitive spacecraft directions and

the vector to the unfavorable attitude $\hat{r}$. In this case, θ_R is the angle between the sensitive spacecraft component and the unsafe attitude direction, and θ_{US} is some buffer angle around the sensitive spacecraft component that shouldn't be exposed to the unsafe attitude for a long duration of time. In the case where one face of the spacecraft is sensitive to an exclusion zone, θ_{US} might be 90°. The line of sight is safe (LOS_S is true) when $\theta_R > \theta_{US}$, as pictured in Fig. 5.16. The pointing duration is safe (T_S is true) when the amount of time that the system has had an unfavorable attitude t_{US} is less than the duration safety limit T_{SL}. These conditions are captured by the labeling function $L(LOS_S, T_S) : \mathbb{R}^2 \to 2^{AP}$:

$$L(LOS_S, T_S) = f(\theta_R, t_{US}) = \begin{cases} \neg LOS_S \wedge \neg T_S & \text{if } \theta_R \leq \theta_{US} \text{ and } t_{US} \geq T_{SL} \\ \neg LOS_S \wedge T_S & \text{if } \theta_R \leq \theta_{US} \text{ and } t_{US} < T_{SL} \\ LOS_S \wedge \neg T_S & \text{if } \theta_R > \theta_{US} \text{ and } t_{US} \geq T_{SL} \\ LOS_S \wedge T_S & \text{if } \theta_R > \theta_{US} \text{ and } t_{US} < T_{SL}. \end{cases} \tag{5.29}$$

Then the safety requirement becomes:

$$\varphi_{C8} = (LOS_S \wedge T_S) \vee (LOS_S \wedge \neg T_S) \vee (\neg LOS_S \wedge T_S) \tag{5.30}$$

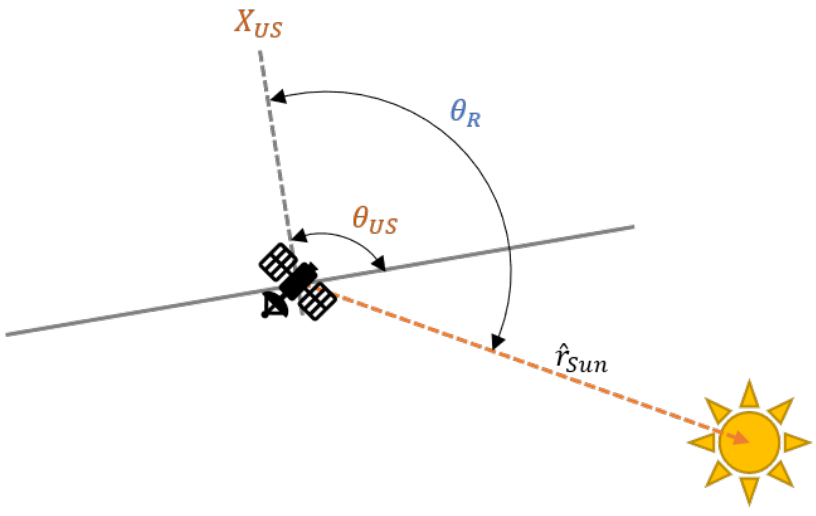


Figure 5.16: Notional depiction of attitude keep out (or exclusion) zone geometry.

[C9] Hazard 9 is divided into 4 different safety constraints, each one of which corresponds to an interlock condition. An interlock condition occurs when two events are mutually exclusive (for example elevator doors do not open when the elevator is in motion, and the elevator will not move when the doors are open). In the case of [C9], each constraint represents a condition that would prevent an autonomous system from maneuvering as a safety measure to prevent excessive fuel use. As constraints are created, care should be taken to minimize the number of resulting interlock conditions. Interlock conditions should also be analyzed for conflicts with other requirements. For example, the Mars Polar Lander included a requirement to go into a sleep mode to conserve batteries after 24 hours without receiving a command, which conflicted with a requirement to test alternative communication methods after 24 hours without a command [271].

[C9a] Spacecraft shall not maneuver if an insufficient amount of time has passed since the last maneuver (H9).

Restricting the minimum time between maneuvers provides an opportunity for a human or computer monitor to detect a fault between firings and prevents a fault from triggering multiple successive firings that deplete fuel reserves. This translates to an interlock condition i being true when the time since the last maneuver T_{s_m} is less than or equal to some minimum time between maneuvers $T_{s_{m_{min}}}$, written as:

$$\varphi_{C9a} = T_{s_m} \leq T_{s_{m_{min}}} \implies i \tag{5.31}$$

[C9b] Spacecraft shall not maneuver if the cumulative maneuver time within a past timeframe exceeds a threshold total time (H9).

Restricting the maximum amount of time that a system can maneuver (i.e. expend fuel) within a time window is a second fault tolerance approach. In this case, if the cumulative maneuver time T_{c_m} exceeds a maximum cumulative maneuver time $T_{c_{m_{max}}}$, an interlock condition i is present.

$$\varphi_{C9b} = T_{c_m} > T_{c_{m_{max}}} \implies i \tag{5.32}$$

[C9c] Spacecraft shall not maneuver if the fuel level is below an operator-specified threshold (H9).

This constraint allows a human operator to specify a fuel level that an automated maneuver cannot operate under as an additional fault tolerant approach. For this constraint, if the fuel level f_l goes below the operator specified threshold, f_{lt}, then an interlock condition i is present.

$$\varphi_{C9c} = f_l \leq f_{lt} \implies i \tag{5.33}$$

[C9d] Spacecraft shall not maneuver when total fuel reaches the end of life threshold with buffer (H9).

Anticipated to be far below the operator's safety threshold, another fault tolerant measure is to ensure faulty automatic maneuvers do not deplete the fuel required to deorbit or reorbit a satellite at the end of its life. Similarly to [C9c], this is written formally as:

$$\varphi_{C9d} = f_l \leq f_{lEOL} \implies i. \tag{5.34}$$

[C10] Spacecraft actuation strategy should conserve actuator use to prevent wear when possible (H10).

As discussed in [167], this constraint is like the acceleration and velocity constraints of [C3] and [C4]. Accelerating uses fuel or can cause excessive wear to actuators like reaction wheels and excessive velocity on internal actuators over an extended period f time can have the same effect. Acceleration and velocity limits are used here, assuming the limits are the same in the positive and negative direction, which may not be the case for all situations.

$$\begin{aligned}\varphi_{C10} =& \Box(||\ddot{x}|| \leq \ddot{x}_{wear}) \wedge (||\ddot{y}|| \leq \ddot{y}_{wear}) \wedge (||\ddot{z}|| \leq \ddot{z}_{wear}) \\ & \wedge (||\ddot{\theta}_1|| \leq \ddot{\theta}_{1_{wear}}) \wedge (\ddot{\theta}_2|| \leq \ddot{\theta}_{2_{wear}}) \wedge (||\ddot{\theta}_3|| \leq \ddot{\theta}_{3_{wear}}) \\ & \wedge (||\dot{x}|| \leq \dot{x}_{wear}) \wedge (||\dot{y}|| \leq \dot{y}_{wear}) \wedge (||\dot{z}|| \leq \dot{z}_{wear}) \\ & \wedge (||\dot{\theta}_1|| \leq \dot{\theta}_{1_{wear}}) \wedge (||\dot{\theta}_2|| \leq \dot{\theta}_{2_{wear}}) \wedge (||\dot{\theta}_3|| \leq \dot{\theta}_{3_{wear}}).\end{aligned} \tag{5.35}$$

A summary of the safety constraint formalization is presented in Table 5.1.

Table 5.1: Summary of formalized safety constraints in ptLTL.

Requirement	ptLTL
φ_{C1}	$\Box scheduled_{COMM} \implies (LOS_{COMM} \wedge FOV_{COMM})$
$\varphi_{C2_{P_c}}$	$\Box(P_c \leq P_{c_{max}})$.
$\varphi_{C2_{r_s}}$	$\Box\|\|\vec{r}_H\|\| > r_s$.
$\varphi_{C2_{rv}}$	$\Box(\|\|\vec{r}_H\|\| > r_s) \wedge \Big((\|\|\vec{v}_H\|\| < v_s) \vee (\vec{v}_H^T \vec{r}_H \geq 0)\Big)$
φ_{C3}	$\Box(\ddot{x}_{min} \leq \ddot{x}) \wedge (\ddot{x} \leq \ddot{x}_{max}) \wedge (\ddot{y}_{min} \leq \ddot{y}) \wedge (\ddot{y} \leq \ddot{y}_{max})$ $\wedge(\ddot{z}_{min} \leq \ddot{z}) \wedge (\ddot{z} \leq \ddot{z}_{max}) \wedge (\ddot{\theta}_{1_{min}} \leq \ddot{\theta}_1)$ $\wedge(\ddot{\theta}_1 \leq \ddot{\theta}_{1_{max}}) \wedge (\ddot{\theta}_{2_{min}} \leq \ddot{\theta}_2) \wedge (\ddot{\theta}_2 \leq \ddot{\theta}_{2_{max}})$ $\wedge(\ddot{\theta}_{3_{min}} \leq \ddot{\theta}_3) \wedge (\ddot{\theta}_3 \leq \ddot{\theta}_{3_{max}})$.
$\varphi_{C4_{set}}$	$\Box X_{sc} \in \mathcal{X}_C$
φ_{C4}	$\Box(\dot{x} \leq \dot{x}_{max}) \wedge (\dot{y} \leq \dot{y}_{max}) \wedge (\dot{z} \leq \dot{z}_{max})$ $\wedge(\dot{\theta}_1 \leq \dot{\theta}_{1_{max}}) \wedge (\dot{\theta}_2 \leq \dot{\theta}_{2_{max}}) \wedge (\dot{\theta}_3 \leq \dot{\theta}_{3_{max}})$
φ_{C5}	$\Box((DOD > DOD_{max}) \vee scheduled_{CHRG})$ $\implies (\theta_{SI} \leq \theta_{CHRG})$.
φ_{C6}	$\Box scheduled_{DATA} \implies (LOS_{DATA} \wedge FOV_{DATA})$
φ_{C7}	$\Box\theta_{EZ} > \frac{\alpha}{2} + \beta$
φ_{C8}	$(LOS_S \wedge T_S) \vee (LOS_S \wedge \neg T_S) \vee (\neg LOS_S \wedge T_S)$
φ_{C9a}	$T_{s_m} \leq T_{s_{m_{min}}} \implies i$
φ_{C9b}	$T_{c_m} > T_{c_{m_{max}}} \implies i$
φ_{C9c}	$f_l \leq f_{lt} \implies i$
φ_{C9d}	$f_l \leq f_{lEOL} \implies i$
φ_{C10}	$\Box(\|\|\ddot{x}\|\| \leq \ddot{x}_{wear}) \wedge (\|\|\ddot{y}\|\| \leq \ddot{y}_{wear}) \wedge (\|\|\ddot{z}\|\| \leq \ddot{z}_{wear})$ $\wedge(\|\|\ddot{\theta}_1\|\| \leq \ddot{\theta}_{1_{wear}}) \wedge (\ddot{\theta}_2\|\| \leq \ddot{\theta}_{2_{wear}}) \wedge(\|\|\ddot{\theta}_3\|\| \leq \ddot{\theta}_{3_{wear}})$ $\wedge(\|\|\dot{x}\|\| \leq \dot{x}_{wear}) \wedge (\|\|\dot{y}\|\| \leq \dot{y}_{wear}) \wedge (\|\|\dot{z}\|\| \leq \dot{z}_{wear})$ $\wedge(\|\|\dot{\theta}_1\|\| \leq \dot{\theta}_{1_{wear}}) \wedge (\|\|\dot{\theta}_2\|\| \leq \dot{\theta}_{2_{wear}}) \wedge (\|\|\dot{\theta}_3\|\| \leq \dot{\theta}_{3_{wear}})$.

5.2.4 Functional Space Traffic Management System Metamodel

A functional metamodel of the space traffic management system was developed concurrently with spacecraft last instant collision avoidance design specifications and requirements. The sparsely populated elements of the metamodel include the expected functional components that will interact with an automatic collision avoidance system from the spacecraft up through informing and controlling agencies, and down to the automatic collision avoidance system inside the controller subsystem. This metamodel was designed to facilitate Systems Theoretic Process Analysis, so controlling function blocks are divided into a control algorithm and process model. Functional blocks that do not control other blocks are modeled as controlled processes. Arrows pointing into each functional block (generally also pointing down) are generally control inputs or other data, while out arrows (generally pointing up) are feedback. Specific functional signals between components are labeled

with a variable name and data type where applicable and reasonable. Recognizing that control inputs occur at multiple levels of a system, multi-level metamodels were created at various levels of abstraction, where the highest level (level 1) depicts a system-wide metamodel, and successive metamodels dive deeper into subsystems and lower level components.

Level 1 Metamodel

To provide context to the hypothetical automatic collision avoidance system, the level 1 metamodel depicted in Fig. 5.17 includes a future hypothetical space surveillance network (SSN), a ground station (GS) block, and a spacecraft (SC) block, which will each be described further in the next several sections.

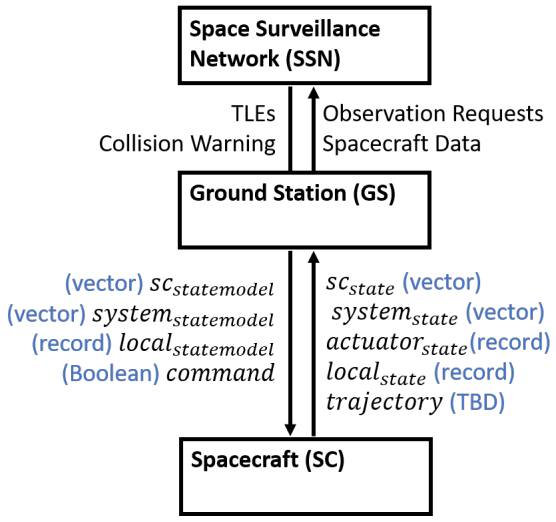


Figure 5.17: Level 1 Metamodel depicting functional relationships between a space surveillance network, a ground station, and a spacecraft in the context of a hypothetical Space Traffic Management system.

Level 2 Space Surveillance Network Metamodel

A level 2 metamodel centered on the SSN block is depicted in Fig. 5.18 A future hypothetical space surveillance network (SSN) functional control block could include a current space situational awareness agency like the Combined Space Operations Center (CSpoC), or a future space traffic control agency. The SSN functional block maintains a catalog of spacecraft, updates two-line element (TLE) sets (or some other future standard data format) describing spacecraft properties,

independently predicts conjunctions, and issues collision warnings. The SSN may also request ground-based observations of satellites from radars, telescopes, and other resources to maintain a current catalog. The control output of the SSN includes updates to individual spacecraft states based on ground observations as well as event-triggered collision warnings. The SSN can accept observation requests from spacecraft owner/operators in the ground station (GS) block, as well as data such as onboard GPS-based position and velocity which may be higher accuracy than ground-based measurements. While the current state of information exchange between the SSN functional block and GS block is accomplished via emails and phone calls [47, 20], a future space traffic control system could include alternative, less human-centric, automatic communications.

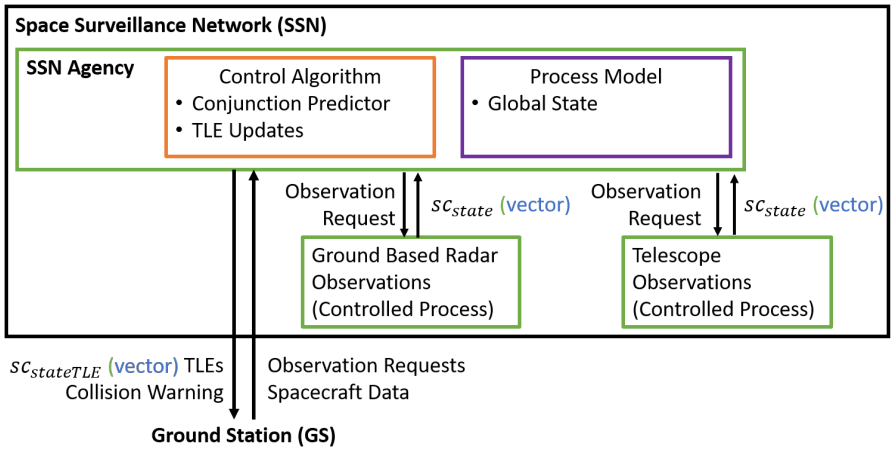


Figure 5.18: Level 2 Metamodel of a hypothetical Space Surveillance Network in the context of a Space Traffic Management system.

Level 2 Ground Station Metamodel

The GS function block, depicted in Fig 5.19, includes a ground operator and computer where the satellite owner/operator interacts with the SSN and spacecraft (SC). The GS may request observations of their satellite or objects at risk of colliding with their satellite and send current spacecraft data such as GPS position and velocity to the SSN. The GS human operator provides mission definitions, operations, maneuvers, and commands that are communicated through the ground computer to the spacecraft. The GS has a process model of the global state of space traffic management (database and picture of all the catalog of objects), local state of N satellites in the spacecraft "neighborhood" (which may have multiple definitions such as the N objects with the closest ap-

proach distance in a finite time horizon), the state of the spacecraft (including position, velocity, orientation, angular rates, and states subsystems as well as other data like health monitoring), and a model of the spacecraft. For the human ground operator, this is a mental model, while the ground computer may maintain a simulation model of the system. Modeled data and commands may be sent from the GS to the SC. The SC provides feedback on the sensed spacecraft, actuator and system states, as well as local state (when capable of sensing or communicating with other objects in the spacecraft "neighborhood," or an echo back of the local state model for verification purposes), and a selected trajectory when a collision avoidance maneuver is activated.

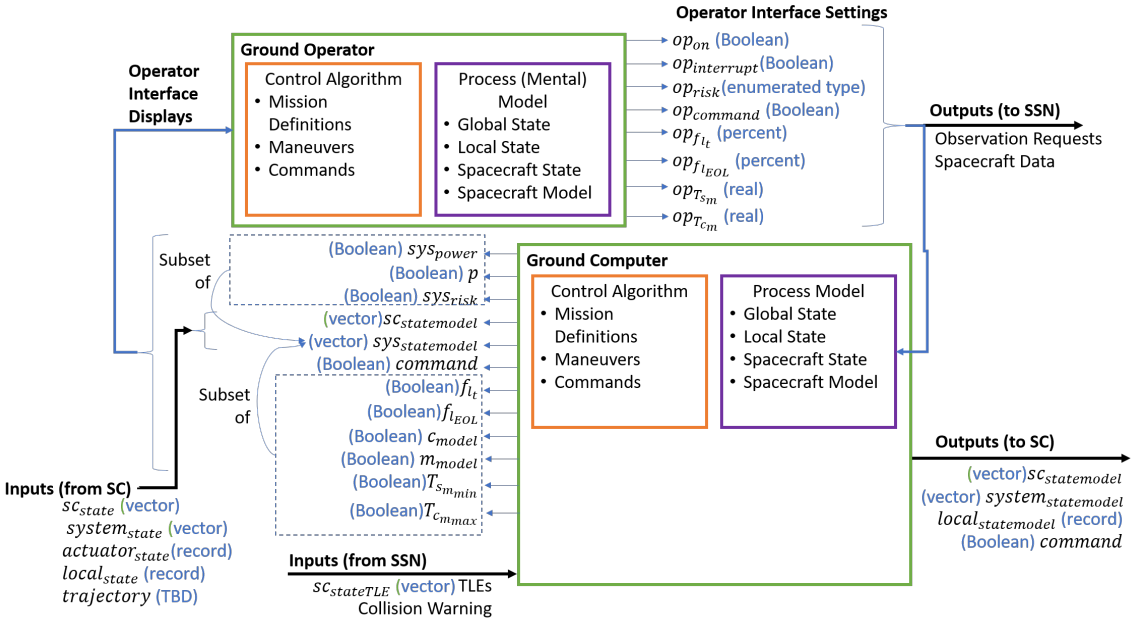


Figure 5.19: Metamodel of a hypothetical ground station in the context of a Space Traffic Management system.

Level 2 Spacecraft Metamodel

Inside the SC are several subsystems. All data in and out pass through the communication (COMM) subsystem and then into a central bus called the command and data handling (CDH) subsystem that keeps a model of the mission, schedule, and tasking and then routes data and commands to other subsystems, including the controller (CTRL).

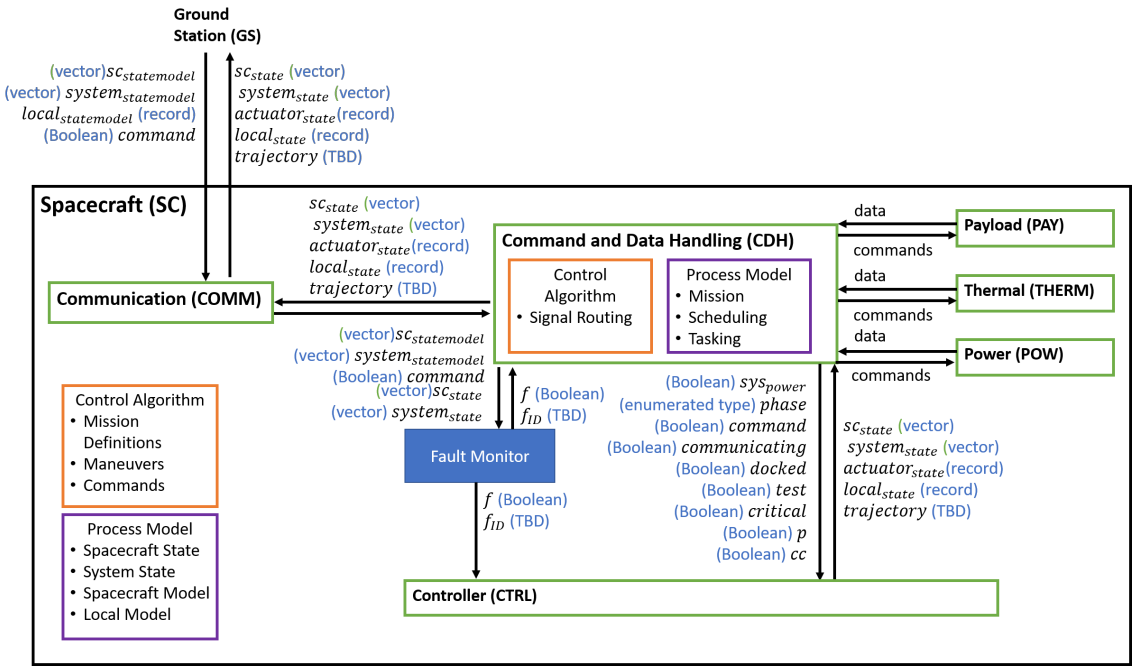


Figure 5.20: Metamodel of a spacecraft in the context of a Space Traffic Management system

Level 3 Spacecraft Controller Metamodel

The CTRL is anticipated to have a monitor function that evaluates the state of the spacecraft, states of different controllers (which could include translational or rotational control for a variety of tasks, special modes, and degraded operations), incoming commands from operators, and other system information like whether the spacecraft is in communication with the ground, whether it is docked to another spacecraft, whether it is in a special test mode, or if there is a critical operation ongoing that cannot be interrupted. One of the control options for the control monitor and selector is the automatic collision avoidance system which is envisioned to contain a model of the spacecraft states, subsystem states, and models of the spacecraft and local object dynamics. The automatic collision avoidance system uses this process model in a set of four proposed functional blocks: an *interlock monitor* block which decides if an interlock condition exists it would be unsafe to maneuver; a *decision logic* block that takes in failure states, interlock conditions, manual interrupts from the ground, whether maneuvers have been completed, and whether a collision is imminent and outputs the control system mode; a maneuver selection block that accepts the operator specified risk levels and state of the spacecraft and local objects to predict when a collision may occur as well as what

maneuver should be selected when to avoid; and a maneuver controller block which accepts the mode of the system and the selected collision avoidance trajectory to maneuver the spacecraft for collision avoidance. The design specifications, requirements, and interfaces in this research focus on this automatic collision avoidance system, including the inputs and outputs to the system and how the operator interacts with it.

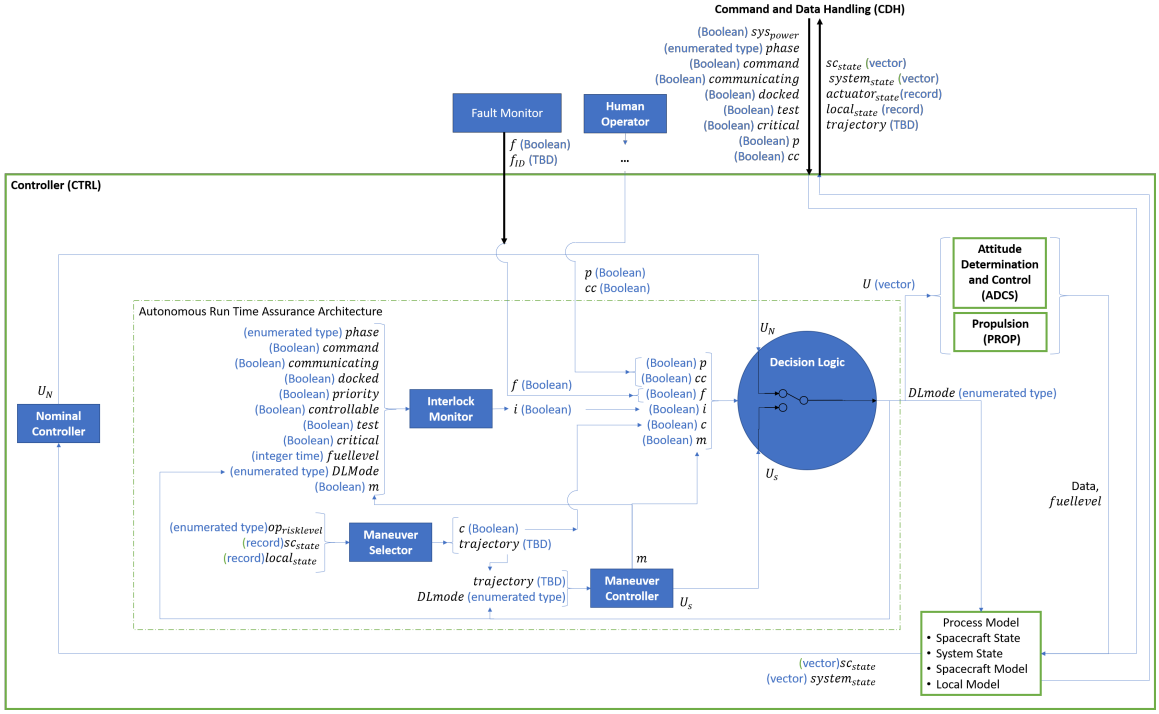


Figure 5.21: Metamodel of a spacecraft controller with automatic collision avoidance function in the context of a Space Traffic Management system

5.3 Formal Design Specifications and Requirements

In this section, formal design specifications are elicited to describe the behavior of the functional components in the form of transitions systems and formal requirements are elicited that describe constraints on the system design. The design specification and requirements elicitation process was iterative, starting with an initial set of requirements inspired by aircraft collision avoidance requirements that were manually converted to appropriate spacecraft collision avoidance requirements, and supplemented with additional requirements informed by a literature search and hazard analysis. In the process of documenting and analyzing the formal requirements and design specifications in

SpeAR additional refinements to the model were identified and implemented. An iterative design process was used to continuously refine the handwritten formal design specifications and requirements, the SpeAR specifications, and the model description.

An abstraction of the STAMP model was used implemented in SpeAR to focus on the elements that provide information to the automatic collision avoidance system. This model is intended to be refined with additional details for any given automatic collision avoidance design and placeholders are included where needed. The SpeAR model has three levels as depicted in Fig.

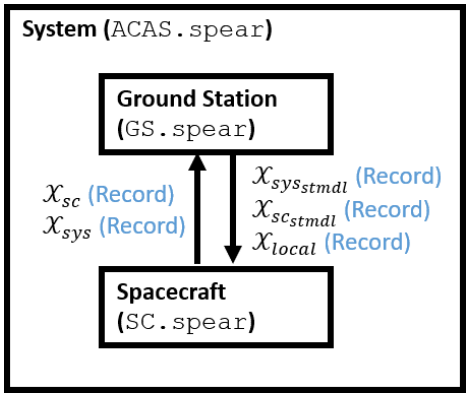


Figure 5.22: Level 1 and Level 2 SpeAR implementation with signal routing

5.3.1 Interlock Monitor Functional Component Formalization

As discussed earlier, a safety interlock is a condition in which it is unsafe to conduct a maneuver or conducting a maneuver may result in higher risk to the vehicle. In this section, design specifications and requirements are described for a safety interlock monitor function that determines when the interlock condition i is true. Like the last section, the design specifications and requirements here may apply to spacecraft automatic maneuvers beyond collision avoidance such as proximity operations or station keeping. While it may be impossible to determine if the list of conditions is ever truly complete, identifying all the conditions explicitly is an important step to understanding conflicts between design specifications. These requirements are inspired by the need for *interlocks* in aircraft automatic collision avoidance systems.

The signals routed through the interlock monitor functional block are depicted in Fig. 5.24.

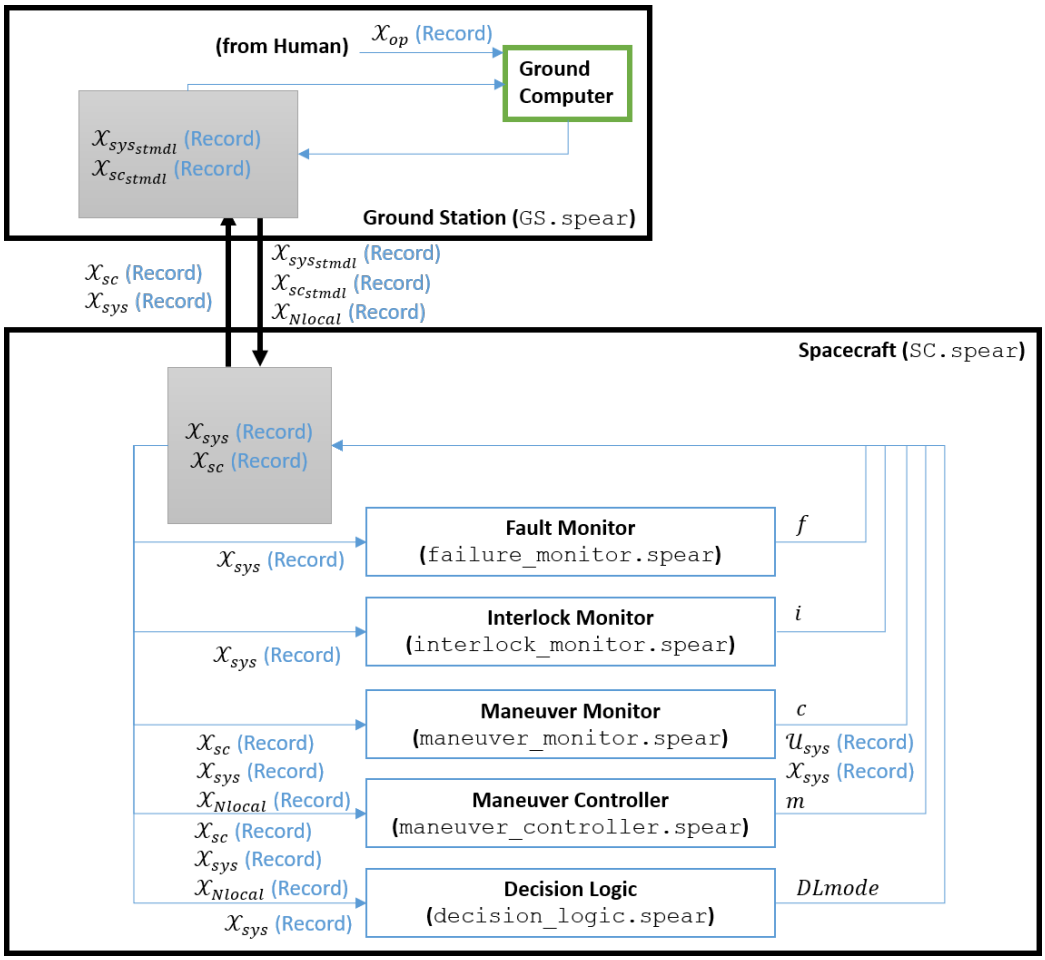


Figure 5.23: Level 2 and Level 3 SpeAR implementation with signal routing

Interlock Monitor Design Specification

Due to the complex nature of the interlock monitor's transition system, a labeling function (rather than the table used to describe the decision logic design specifications) describes the transitions between the states of interlock present i and no interlock present $\neg i$. Let $AP = \{i, \neg i\}$ describe whether or not an interlock exists. This intermediate function, described by Eqn. 5.36, can then serve as a simplification. Each of the variables used in the interlock monitor are described further

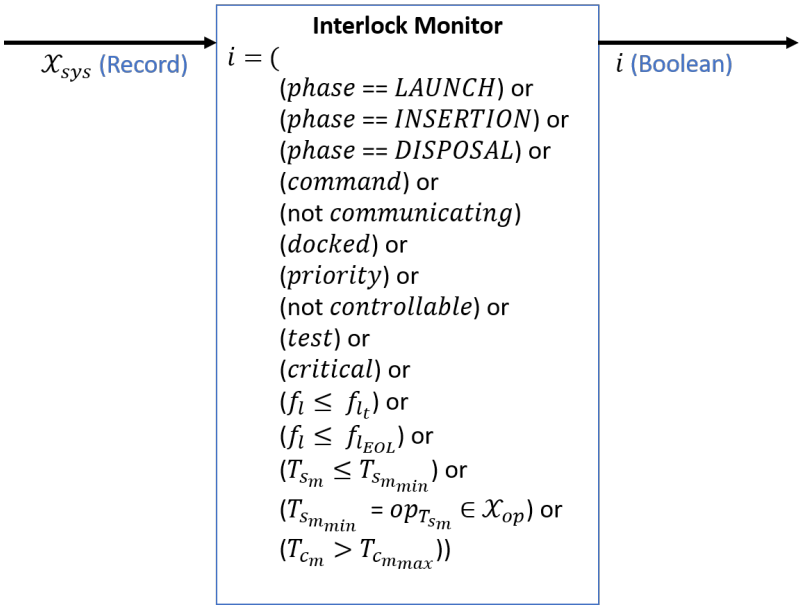


Figure 5.24: Description of the signals routed through the interlock monitor functional block.

in the requirements.

$$L(Phase, command, communicating, docked, priority, controllable, test, critical, T_{s_m}, T_{c_m}, f_l) = \begin{cases} i & (Phase == launch) \vee (Phase == insertion) \vee (Phase == disposal) \\ & \vee command \vee \neg communicating \vee docked \vee \neg controllable \vee test \\ & \vee critical \vee (T_{s_m}(m) \leq T_{s_{m_{min}}}) \vee (T_{c_m}(DLmode) \geq T_{c_{m_{max}}}) \\ & \vee (f_l \leq f_{l_t}) \vee (f_l \leq f_{l_{EOL}}) \\ \neg i & \text{otherwise} \end{cases} \tag{5.36}$$

Interlock Monitor Requirements

The natural language expression of each with corresponding rationale are listed below and the formal expression of each is summarized in Table 5.2. Each interlock condition (IL) requirement implies that the interlock condition i is true.

Table 5.2: Formal expression of system interlock condition safety requirements in ptLTL.

Requirement		
IL01a	$\varphi_{IL01a} = \Box(Phase == launch)$	$\Longrightarrow i$
IL01b	$\varphi_{IL01b} = \Box(Phase == insertion)$	$\Longrightarrow i$
IL01c	$\varphi_{IL01c} = \Box(Phase == disposal)$	$\Longrightarrow i$
IL02	$\varphi_{IL02} = \Box command$	$\Longrightarrow i$
IL03	$\varphi_{IL03} = \Box\neg communicating$	$\Longrightarrow i$
IL04	$\varphi_{IL04} = \Box docked$	$\Longrightarrow i$
IL05	$\varphi_{IL04} = \Box priority$	$\Longrightarrow i$
IL06	$\varphi_{IL06} = \Box\neg controllable$	$\Longrightarrow i$
IL07	$\varphi_{IL07} = \Box test$	$\Longrightarrow i$
IL08	$\varphi_{IL08} = \Box critical$	$\Longrightarrow i$
IL09	$\varphi_{IL09} = \Box T_{s_m}(m) \leq T_{s_{m_{min}}}$	$\Longrightarrow i$
IL10	$\varphi_{IL10} = \Box T_{c_m}(DLmode) \geq T_{c_{m_{max}}}$	$\Longrightarrow i$
IL11	$\varphi_{IL11} = \Box f_l \leq f_{l_t}$	$\Longrightarrow i$
IL12	$\varphi_{IL12} = \Box f_l \leq f_{l_{EOL}}$	$\Longrightarrow i$
IL13	`init`	$\Longrightarrow \neg i$

[IL01a] The system shall set the interlock condition (i) to true when the spacecraft is in launch phase ($Phase == launch$).

[IL01b] The system shall set the interlock condition (i) to true when the spacecraft is in insertion phase ($Phase == insertion$).

[IL01c] The system shall set the interlock condition (i) to true when the spacecraft is in disposal phase ($Phase == disposal$).

Rationale: First, it is assumed that in the lifecycle of the space vehicle, it will go through several mission phases including: launch (from Earth, controlled by launch vehicle), insertion (orbital insertion, controlled by launch vehicle), detumble (stop of tumbling motion after exiting the launch vehicle), deployment (deploy appendages like solar panels), acquisition (initial spacecraft state determination), nominal operations (including various mission-oriented maneuvers), safe (possible operational capability or power reduction during emergencies and failures), other (including special modes), and disposal (procedures to "de-orbit" or "re-orbit" satellite). An interlock during launch and insertion ensures the spacecraft will not maneuver before it has successfully exited the launch vehicle and any other shrouding, while an interlock during disposal prevents maneuvers that would use critical fuel needed for the satellite

disposal process.

[IL02] The system shall set the interlock condition (i) to true when following manual maneuver commands ($command$).

Rationale: It is assumed that the operator has better situational awareness than the spacecraft. It is also assumed that the manual maneuver is of higher priority of than the automatic maneuver.

[IL03] The system shall set the interlock condition (i) to true when outside communication window of a ground station ($\neg communicating$).

Rationale: This requirement is intended for incremental introduction of automatic capabilities, and may be removed when the automatic maneuver system heritage provides confidence in unsupervised operation. Until that point, it is assumed that the operator should be on the loop (i.e. supervising) and able to stop the maneuver if it is faulty; this can only be accomplished when the spacecraft has communication established with the ground station. It is assumed that latency between the spacecraft and ground station is small enough that interrupting a maneuver is feasible.

[IL04] The system shall set the interlock condition (i) to true when the spacecraft is docked with another spacecraft ($docked$).

Rationale: It is assumed that automatic maneuvers like station keeping or collision avoidance are not designed for a docked spacecraft configuration, and the system should go into standby.

[IL05] The system shall set the interlock condition (i) to true if a control system at a higher priority level ($priority$) is actively maneuvering.

Rationale: It is assumed it is possible to assign priority to automatic maneuvers. For example, if an automatic station keeping and automatic collision avoidance system are both on board, it might be of higher priority to avoid collisions than to maintain station. This is to be decided by the owner/operator.

[IL06] The system shall set the interlock condition (i) to true when the spacecraft is in uncontrollable ($\neg controllable$) state.

Rationale: The system should be able to detect that the spacecraft is in an uncontrollable state, and not attempt a maneuver that could exacerbate the system state.

[IL07] The system shall set the interlock condition (i) to true when in a test mode ($test$).

Rationale: It is assumed that there are on board tests or experiments that might need to be completed and that the system would need to be locked out when in test mode.

[IL08] The system shall set the interlock condition (i) to true during operations critical processes ($critical$).

Rationale: It is assumed that some "critical" level operations activities should not be interrupted for an automatic maneuver. Operations should be marked "critical" sparingly. This differs from priority in that it could be an operation that doesn't involve maneuvering while priority implies maneuvering.

[IL09] The system shall set the interlock condition (i) to true if an insufficient amount of time since the last maneuver ($T_{s_m}(m) < T_{s_{m_{min}}}$).

Rationale: This safety requirement provides is a fault tolerance measure designed to prevent the system from maneuvering many times in quick succession. It is designed to allow the operator time to monitor system and detect faults. The minimum time between maneuvers is to be determined by the operators. The time since maneuver T_{s_m} is a function of the maneuver completed flag m.

[IL10] The system shall set the interlock condition (i) to true if the cumulative maneuver time within a past timeframe exceeds a threshold total time ($T_{c_m}(DLmode) > T_{c_{m_{max}}}$).

Rationale: This safety requirement also provides a fault tolerance measure that limits maneuver duration time to prevent the system from expending excessive resources. It allows the operator time to monitor system and detect faults. The cumulative time is a function of the decision logic mode ($DLmode \in \{S, F, M, A\}$) and the maximum value is expected to be set by the operator.

[IL11] The system shall set the interlock condition (i) to true if the fuel level (f_l) is below a specified threshold (f_{l_t}).

Rationale: This safety requirement also provides fault tolerance, in this case to limit cumu-

lative fuel usage over many successive maneuvers in a short time frame and allows operator time to monitor system and detect faults. The fuel level threshold is expected to be set by the operator. The operator may initially be very conservative with this setting and slowly reduce it throughout the mission. The operator should be notified if this threshold is exceeded by other operations outside of the automatic maneuver, so that they can decide to lower it.

[IL12] The system shall set the interlock condition (i) to true when fuel level (f_l) reaches end of life threshold with buffer ($f_{l_{EOL}}$).

Rationale: This safety requirement ensures that the fuel required for end of life disposal of the spacecraft is still available. This should be set by the operators of the mission.

[IL13] The system shall set the interlock condition (i) to false for the initial state (`init`).

Rationale: In the initial state, the system is not experiencing an interlock until one is detected. In the decision logic, the system state starts in standby (S) to give the system time to assess whether an interlock or failure is present before maneuvering.

5.3.2 Ground Station Computer Functional Component Formalization

This section provides design specifications and requirements for the operator to interact with the spacecraft and maintain situational awareness during operations through the mission computer. The operator is located on the ground and communicates with the spacecraft through a mission computer as depicted in Fig. 5.19. These requirements focus solely on the type of data needed for the system to function, not human factors design requirements. However, a more thorough system specification might include human factors considerations for the design of:

- an on/off switch for the automatic maneuver system,
- a mechanism to interrupt an automatic maneuver,
- a mechanism to manually engage an automatic maneuver,
- a mechanism to select automatic maneuver system risk level,
- how the spacecraft state and system state are displayed,
- how the maneuver initiation and termination are displayed,

- how the need to maneuver is displayed,
- and how the operator selected variables are input and displayed.

The signals routed through the operator functional block are depicted in Fig. 5.25

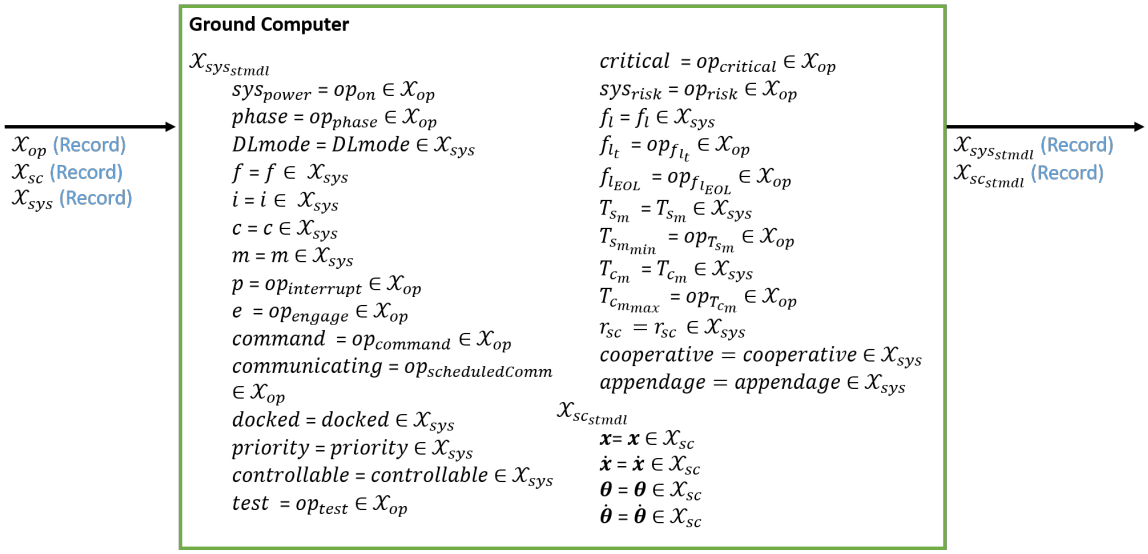


Figure 5.25: Description of the signals routed through the ground station computer.

Ground Computer Design Specification

The ground computer design specification is a record update that updates a model of the spacecraft state and spacecraft system state based on values coming from the spacecraft or operator input states. First, the system state model is updated with data from the spacecraft, then the operator values are updated to reflect operator input as described in the following:

$$\begin{aligned}\mathcal{X}_{sys_{stmdl}} =& \mathcal{X}_{sys}\{sys_{power} = op_{on}\}\{Phase = op_{phase}\}\{p = op_{interrupt}\}\{e = op_{engage}\} \\ & \{command = op_{command}\}\{communicating = op_{scheduledComm}\}\{test = op_{test}\} \\ & \{critical = op_{critical}\}\{sys_{risk} = op_{risklevel}\}\{f_{l_t} = op_{f_{l_t}}\}\{f_{l_{EOL}} = op_{f_{l_{EOL}}}\} \\ & \{T_{s_{m_{min}}} = op_{T_{sm}}\}\{T_{c_{m_{max}}} = op_{T_{cm}}\}\end{aligned} \tag{5.37}$$

Ground Computer Requirements

The ground computer requirements specify how the ground station computer shall respond to operator inputs. The formalization of each requirement is included in Table 5.3.

Table 5.3: Formal expression of ground computer requirements.

Requirement ID	Requirement
RGC01	$\varphi_{RGC01} = \Box(op_{on} \implies sys_{power})$
RGC02	$\varphi_{RGC02} = \Box(op_{phase} = Phase)$
RGC03	$\varphi_{RGC03} = \Box(op_{interrupt} \implies p)$
RGC04	$\varphi_{RGC04} = \Box(op_{engage} \implies e)$
RGC05	$\varphi_{RGC05} = \Box(op_{command} \implies command)$
RGC06	$\varphi_{RGC06} = \Box(op_{scheduledComm} \implies communicating)$
RGC07	$\varphi_{RGC07} = \Box(op_{test} \implies test)$
RGC08	$\varphi_{RGC08} = \Box(op_{critical} \implies critical)$
RGC09	$\varphi_{RGC09} = \Box(sys_{risk} == op_{risklevel})$
RGC10	$\varphi_{RGC10} = \Box(f_{l_t} == op_{f_{l_t}})$
RGC11	$\varphi_{RGC11} = \Box(f_{l_{EOL}} == op_{f_{l_{EOL}}})$
RGC12	$\varphi_{RGC12} = \Box(T_{s_{m_{min}}} == op_{T_{s_m}})$
RGC13	$\varphi_{RGC13} = \Box(T_{c_{m_{max}}} == op_{T_{c_m}})$
RGC14	$\varphi_{RGC14} = \Box(f_{l_{model}} == f_l)$
RGC14UL	$\varphi_{RGC14UL} = \Box(f_{l_{model}} \leq 100.0)$
RGC14LL	$\varphi_{RGC14LL} = \Box(f_{l_{model}} \geq 0.0)$
RGC15	$\varphi_{RGC15} = \Box(DLMode_{model} == DLMode)$
RGC16	$\varphi_{RGC16} = \Box(f_{model} == f)$
RGC17	$\varphi_{RGC17} = \Box(i_{model} == i)$
RGC18	$\varphi_{RGC18} = \Box(m_{model} == m)$
RGC19	$\varphi_{RGC19} = \Box(c_{model} == c)$
RGC20	$\varphi_{RGC20} = \Box(docked_{model} == docked)$
RGC21	$\varphi_{RGC21} = \Box(priority_{model} == priority)$
RGC22	$\varphi_{RGC22} = \Box(controllable_{model} == controllable)$
RGC23	$\varphi_{RGC23} = \Box(\mathcal{X}_{sc_{stmdl}} == \mathcal{X}_{sc})$

[RGC01] The system shall turn off ($sys_{power} == OFF$) when the ground operator directs the system to turn off ($\neg op_{on}$).

Rationale: The operator may want to turn off automatic maneuvers for certain portions of the mission. In addition, the operator may want to turn the system off if they suspect there is an undetected system fault. [This requirement traces to the need for an *off switch* inspired by aircraft collision avoidance systems.]

[RGC01a] The system shall turn on ($sys_{power} == ON$) when the ground operator directs the system to

turn off (op_{on}).

Rationale: Compliments of the requirements such as RGC01a: $\varphi_{RGC01a} = \square(\neg op_{on} \implies \neg sys_{power})$ may also be included in the analysis.

[RGC02] The mission phase on the spacecraft computer $Phase$ should reflect the mission phase set in the ground station computer op_{phase}.

Rationale: It is assumed that the mission phases are updated on the ground and communicated to the spacecraft. During certain mission phases, it is unsafe for the system to automatically maneuver. [This requirement traces to the *interlock* conditions.]

[RGC03] The system shall set the manual interrupt from a person condition (p) to true when the ground operator interrupts an automatic maneuver ($op_{interrupt}$).

Rationale: The operator should be able to stop an undesired automatic maneuver as a human-on-the-loop supervisor. [This requirement traces to the need for the system to be *nuisance free*, and as a fault tolerance function filling the need to *do no harm*, inspired by aircraft collision avoidance systems.]

[RGC04] The system shall set the manually engage a maneuver system setting (e) to true when the ground operator commands an automatic maneuver (op_{engage}).

Rationale: The operator should be able to activate an automatic maneuver. This is a feature that enables the operator to select the currently considered automatic maneuver to ensure safe separation of space objects if they are unhappy with the state of the spacecraft. [This requirement traces inspiration from the pilot activated recovery system in aircraft systems.]

[RGC05] The system shall execute commands ($command$) when the operator commands a preplanned maneuver ($op_{command}$).

Rationale: It is assumed that the operator will still on occasion upload scripted, manually planned maneuvers to the spacecraft and that this planning will incorporate all available information. This setting helps to ensure situational awareness and coordination between spacecraft and operators. [This requirement traces to *interlocks* and system transparency.]

[RGC06] The spacecraft system shall set communication scheduled ($communicating$) to true when the ground station computer has scheduled communication ($op_{scheduledComm}$).

Rationale: Communication between the ground station and the spacecraft may not occur all the time. When the communication is scheduled, this flag should be set to true on both the spacecraft and at the ground station. [This requirement traces to system transparency inspired by aircraft systems and to H1, H6, C1, and C6 in the hazard analysis.]

[RGC07] The system shall be set to the test mode ($test$) when the operator places the system in test mode(op_{test}).

Rationale: It is envisioned that the spacecraft will have a special mode for testing and that some system functions will be disabled during the mode. [This requirement traces to system transparency inspired by aircraft systems.]

[RGC08] The system shall execute commands ($critical$) when the operator commands a preplanned maneuver ($op_{command}$).

Rationale: When a critical operation is occurring it should be identified on the ground and communicated to the spacecraft. It is assumed that some "critical" level operations activities should not be interrupted for an automatic maneuver. Operations should be marked "critical" sparingly. This differs from priority in that it could be an operation that doesn't involve maneuvering while priority implies maneuvering. [This requirement traces to *interlock* conditions]

[RGC09] The system shall set the risk level (sys_{risk}) to the operator selection($op_{risklevel}$).

Rationale: This is analogous to how much safety buffer or margin is limited for the maneuver. The operator should be able to select acceptable level of risk/uncertainty. This risk/uncertainty is expected to impact the size of the reachable space, with high risk/uncertainty used to project a smaller reachable space (such as $1-\sigma$ uncertainty ellipses versus $3-\sigma$ uncertainty ellipses). Higher risk settings are expected to result in less false positive collisions at the risk of false negatives, resulting in fewer collision avoidance maneuvers, but a higher risk of a collision. [This requirement traces to the *risk level selection* need inspired by aircraft collision avoidance systems.]

[RGC10] The system shall set the fuel level safety threshold (f_{l_t}) to the operator specified fuel level safety threshold ($op_{f_{l_t}}$).

Rationale: As a fault tolerance measure, a modifiable minimum safe fuel level will be specified by the operator to prevent the automatic system from overuse of fuel during undetected fault. [This requirement traces to H9 and C9c in the hazard analysis.]

[RGC11] The system shall set the fuel level end of life threshold ($f_{l_{EOL}}$) to the operator specified fuel level end of life threshold ($op_{f_{l_{EOL}}}$).

Rationale: As a fault tolerance measure, a modifiable minimum end of life fuel level will be specified by the operator to prevent the automatic system from overuse of fuel during undetected fault. [This requirement traces to H9 and C9d in the hazard analysis.]

[RGC12] The system shall set the minimum time between maneuvers($T_{s_{m_{min}}}$) to the operator specified minimum time between maneuvers ($op_{T_{s_m}}$).

Rationale: As a fault tolerance measure, a modifiable minimum time between automated maneuvers will be specified by the operator to prevent the automatic system from overusing fuel during undetected fault. [This requirement traces to H9 and C9a in the hazard analysis.]

[RGC13] The system shall set the maximum cumulative maneuver time ($T_{c_{m_{max}}}$) to the operator specified maximum cumulative maneuver time ($op_{T_{c_m}}$).

Rationale: As a fault tolerance measure, a modifiable minimum cumulative maneuver time will be specified by the operator to prevent the automatic system from overusing fuel during undetected fault. [This requirement traces to H9 and C9b in the hazard analysis.]

[RGC14] The system shall set the modeled state of the fuel level ($f_{l_{model}}$) to the spacecraft's fuel level reading (f_l).

Rationale: Keeping an updated model of the amount of fuel left in the spacecraft based on the fuel level facilitates situational awareness for the operator. In addition, it could be used as a fault tolerance measure, where infeasible or excessive changes in fuel levels are used to check for faults. [This requirement traces to H9 and *system transparency* inspired by the aircraft collision avoidance systems.]

[RGC14] : [RGC14UL] and [RGC14LL] The system shall limit the modeled fuel level values to feasible values.

Rationale: Due to the specification of types in SpeAR, RGC14UL (upper level) and RGC14LL

(lower level) are added to ensure that the fuel level settings conform to the predicate on the fuel level type, which states that it should be a value between 0 and 100.0%.

[RGC15] - [RGC23] The system shall update the spacecraft system statemodel variables ($DLMode$, f, i, m, c, $docked$, $priority$, $controllable$, and $\mathcal{X}_{sc}$) set by the spacecraft with the values sent from the spacecraft ($DLMode$, f, i, m, c, $docked$, $priority$, $controllable$, and $\mathcal{X}_{sc}$).

Rationale: This requirement ensures updates to the spacecraft and system states are available to the operator, who requires situational awareness and may need to react to state changes. For instance, if protruding appendages like solar panels or drag devices are retracted prior to a maneuver, the state of those systems should be sent to the operator. Awareness of automatic maneuvers is critical to the operator's understanding of the spacecraft and situational awareness during operations. Also, requirements to update the model of the condition to maneuver c is included with the intent that the automatic system degrades from an automatic maneuver system to an advisory system when an interlock condition is present. Even if the system is not able to maneuver, it should inform the operator that the conditions to maneuver are present. When the system is in Standby S, it becomes an alerting system that should tell an operator when a need to maneuver is imminent even if it is locked out of maneuvering. [These requirements trace to the need for *system transparency* as identified in the aircraft systems].

Ground Computer Discussion

Though outside the scope of this research, the ground station computer could include independent estimation of the spacecraft state that it compares with the inputs from the spacecraft as an additional fault detection capability.

5.3.3 Maneuver Controller Design Specifications and Requirements Formalization

The maneuver controller, also referred to as the reversionary (backup) controller, computes and commands a simple, verified, control response to the state of the plant for functions such as collision avoidance. The computation occurs continuously so that when the maneuver selector, also called maneuver monitor or boundary monitor, detects a potential boundary violation (such as an impending collision), the maneuver controller is ready to engage the preplanned maneuver.

Do No Harm Requirements

This section describes the safety requirements on the planned and actual maneuver to prevent harm to the system and its components. The decision logic, interlock monitor, a subset of the operator requirements, and any failure monitor requirements developed in future work could also be considered do no harm requirements as they are designed to prevent unsafe activations of an automated or autonomous maneuver. The do no harm requirements are further divided into two broad categories: logic and reachability. The logic-based requirements abstract the system state to a set of atomic propositions, e.g. the system either violates acceleration constraints or it doesn't, while the reachability requirements specify that the system stay in a set of safe states. The logic requirements can be verified by imposing limiters in the design while the reachability requirements may only be verified by a complex analysis of the system state or some propagation of the system state in future timesteps. First, the logic-based requirements are introduced and summarized in Table 5.4.

[RMC01] The automatic maneuver translational and angular acceleration shall be within payload limitations.

Rationale: The spacecraft payload may have limited translational and angular accelerations for safe operation. The payload is called out separately because the same spacecraft bus may be used on multiple spacecraft for different payloads. [This requirement traces to the need for *"payload" acceleration limits* inspired by the aircraft collision avoidance systems, as well as to H3 and C3 in the Hazard Analysis.]

[RMC02] The automatic maneuver translational and angular acceleration shall be within structural limits of the spacecraft.

Rationale: Maneuver should be below thresholds that would cause damage to structure, including deployed appendages like solar panels, inflatables booms, and tethers. [This requirement traces to the *structural acceleration limit* inspired by aircraft collision avoidance systems and to H3 and C3 in the hazard analysis.]

[RMC03] The automatic maneuver translational and angular acceleration shall be within limits for special configurations.

Rationale: There may be multiple spacecraft configurations, primarily described by the de-

Table 5.4: Formal expression of logic-based do no harm, maneuver constraint requirements.

Requirement ID	Requirement
RMC01x	$\varphi_{R01x} = \Box(-\ddot{x}_{pay} \leq \ddot{x}) \wedge (\ddot{x} \leq \ddot{x}_{pay})$
RMC01y	$\varphi_{R01y} = \Box(-\ddot{y}_{pay} \leq \ddot{y}) \wedge (\ddot{y} \leq \ddot{y}_{pay})$
RMC01z	$\varphi_{R01z} = \Box(-\ddot{z}_{pay} \leq \ddot{z}) \wedge (\ddot{z} \leq \ddot{z}_{pay})$
RMC01r1	$\varphi_{R01r1} = \Box(-\ddot{r}_{1_{pay}} \leq \ddot{r}_1) \wedge (\ddot{r}_1 \leq \ddot{r}_{1_{pay}})$
RMC01r2	$\varphi_{R01r2} = \Box(-\ddot{r}_{2_{pay}} \leq \ddot{r}_2) \wedge (\ddot{r}_2 \leq \ddot{r}_{2_{pay}})$
RMC01r3	$\varphi_{R01r3} = \Box(-\ddot{r}_{3_{pay}} \leq \ddot{r}_3) \wedge (\ddot{r}_3 \leq \ddot{r}_{3_{pay}})$
RMC02x	$\varphi_{R02x} = \Box(-\ddot{x}_{str} \leq \ddot{x}) \wedge (\ddot{x} \leq \ddot{x}_{str})$
RMC02y	$\varphi_{R02y} = \Box(-\ddot{y}_{str} \leq \ddot{y}) \wedge (\ddot{y} \leq \ddot{y}_{str})$
RMC02z	$\varphi_{R02z} = \Box(-\ddot{z}_{str} \leq \ddot{z}) \wedge (\ddot{z} \leq \ddot{z}_{str})$
RMC02r1	$\varphi_{R02r1} = \Box(-\ddot{r}_{1_{str}} \leq \ddot{r}_1) \wedge (\ddot{r}_1 \leq \ddot{r}_{1_{str}})$
RMC02r2	$\varphi_{R02r2} = \Box(-\ddot{r}_{2_{str}} \leq \ddot{r}_2) \wedge (\ddot{r}_2 \leq \ddot{r}_{2_{str}})$
RMC02r3	$\varphi_{R02r3} = \Box(-\ddot{r}_{3_{str}} \leq \ddot{r}_3) \wedge (\ddot{r}_3 \leq \ddot{r}_{3_{str}})$
RMC03x	$\varphi_{R03x} = \Box(-\ddot{x}_{sp} \leq \ddot{x}) \wedge (\ddot{x} \leq \ddot{x}_{sp})$
RMC03y	$\varphi_{R03y} = \Box(-\ddot{y}_{sp} \leq \ddot{y}) \wedge (\ddot{y} \leq \ddot{y}_{sp})$
RMC03z	$\varphi_{R03z} = \Box(-\ddot{z}_{sp} \leq \ddot{z}) \wedge (\ddot{z} \leq \ddot{z}_{sp})$
RMC03r1	$\varphi_{R03r1} = \Box(-\ddot{r}_{1_{sp}} \leq \ddot{r}_1) \wedge (\ddot{r}_1 \leq \ddot{r}_{1_{sp}})$
RMC03r2	$\varphi_{R03r2} = \Box(-\ddot{r}_{2_{sp}} \leq \ddot{r}_2) \wedge (\ddot{r}_2 \leq \ddot{r}_{2_{sp}})$
RMC03r3	$\varphi_{R03r3} = \Box(-\ddot{r}_{3_{sp}} \leq \ddot{r}_3) \wedge (\ddot{r}_3 \leq \ddot{r}_{3_{sp}})$
RMC04x	$\varphi_{RMC04x} = \Box((x_{sc} - x_{obstacle}) \geq r_{min}) \vee ((x_{sc} - x_{obstacle}) \leq -r_{min})$
RMC04y	$\varphi_{RMC04y} = \Box((y_{sc} - y_{obstacle}) \geq r_{min}) \vee ((y_{sc} - y_{obstacle}) \leq -r_{min})$
RMC04z	$\varphi_{RMC04z} = \Box((z_{sc} - z_{obstacle}) \geq r_{min}) \vee ((z_{sc} - z_{obstacle}) \leq -r_{min})$
RMC05x	$\varphi_{RMC05x} = \Box(\dot{x} \leq \dot{x}_{max}) \vee (\dot{x} \geq -\dot{x}_{max})$
RMC05y	$\varphi_{RMC05y} = \Box(\dot{y} \leq \dot{y}_{max}) \vee (\dot{y} \geq -\dot{y}_{max})$
RMC05z	$\varphi_{RMC05z} = \Box(\dot{z} \leq \dot{z}_{max}) \vee (\dot{z} \geq -\dot{z}_{max})$
RMC05r1	$\varphi_{RMC05r1} = \Box(\dot{\theta}_1 \leq \dot{\theta}_{1_{max}}) \vee (\dot{\theta}_1 \geq \dot{\theta}_{1_{min}})$
RMC05r2	$\varphi_{RMC05r2} = \Box(\dot{\theta}_2 \leq \dot{\theta}_{2_{max}}) \vee (\dot{\theta}_2 \geq \dot{\theta}_{2_{min}})$
RMC05r3	$\varphi_{RMC05r3} = \Box(\dot{\theta}_3 \leq \dot{\theta}_{3_{max}}) \vee (\dot{\theta}_3 \geq \dot{\theta}_{3_{min}})$

ployment state of appendages like solar panels or antennas which may be stowed for launch, and could optionally be stowed for maneuver operations. [This requirement traces to the *special configuration acceleration limit* inspired by aircraft collision avoidance systems and to Hazard 3 and Constraint 3 in the hazard analysis].

[RMC04] The automatic maneuver shall not cause a collision with another object. i.e. The spacecraft shall remain safely separated from the obstacle.

Rationale: As discussed in the hazards analysis, there are many ways to assure that two spacecraft are safely separated from keeping probability of collision below threshold values, to maintaining a minimum distance, to maintaining a combination of minimal position and

velocity constraints. As an initial cut, the simplest of these cases, minimum distance is considered. Note that eventually it would be better to use a combination of position and velocity, or to compute a distance, but due to the linear computation restrictions of the SpeAR tool used in this research [This requirement traces to H2 and C2 in the hazard analysis].

[RMC05] The automatic maneuver shall maintain controllability of the spacecraft. i.e. The velocity of the spacecraft shall be small enough to allow it to be adapted within an acceptable timeframe.

Rationale: As discussed in the hazard analysis section, it is important to keep the velocity of the spacecraft within limits that allow the spacecraft controller to respond as needed within reasonable timeframes. [This requirement traces to H4 and C4 in the hazard analysis.]

For reachability-based analysis, the following do no harm requirements are introduced below, followed by a summary in Table 5.5.

Table 5.5: Formal expression of reachability-based do no harm, maneuver constraint requirements.

Requirement ID	Requirement
RMC04a	$\varphi_{RMC04a} = \Box\mathcal{R}_c \cap \mathcal{R}_t = \forall t \in [0, t_{horizon}]$
RMC04b	$\varphi_{RMC04b} = \Box\mathcal{R}_c \cap \mathcal{R}_t = \forall t \in [0, t_{horizon}], \forall \Delta V \leq \Delta V_{planned}$
RMC05a	$\varphi_{RMC05a} = \Box\mathcal{R}_c \cap \mathcal{R}_{unstable} = \forall t \in [0, t_{horizon}]$
RMC05b	$\varphi_{RMC05b} = \Box\mathcal{R}_c \cap \mathcal{R}_{uncontrollable} = \forall t \in [0, t_{horizon}]$
RMC07a	$\varphi_{RMC07a} = \Box(x_{m_{fin}} \in \mathcal{X}_{fin_{safe}}) \implies m$
RMC07b	$\varphi_{RMC07b} = \Box x_{m_{init}} \in \mathcal{X}_{init_{safe}}$
RMC07c	$\varphi_{R19} = \Box u_{trajectory} \in \mathcal{U}_{trajectory}$

[RMC04a] The automatic maneuver shall not cause a collision with another object: i.e. The projected reachable space of the automatic maneuver shall not intersect the projected reachable space of any object in the spacecraft neighborhood within a finite time horizon.

Rationale: Maneuver shape is assumed to consider dynamics and state of neighboring objects in its maneuver decision. [Traces to inspiration from aircraft system that track several obstacles as well as to the hazard analysis H2 and C2.]

[RMC04b] The automatic maneuver trajectory should be passively safe: i.e. the reachable space of the spacecraft trajectory shall not intersect the reachable space of the object being avoided for all maneuvers (ΔV) smaller than the planned maneuver.

Rationale: If the computer shuts down during the maneuver, it should not result in a collision. This is really more applicable to autonomous rendezvous, proximity operations and docking than it is to collision avoidance but is included here for completeness. [This requirement traces to H2 and C2 in the hazard analysis.]

[RMC05a] The automatic maneuver shall retain stability of spacecraft: i.e. the reachable state of the maneuver shall not intersect the reachable state of unstable maneuvers.

Rationale: The maneuver should not result in an unstable spacecraft state. [This requirement traces to inspiration from aircraft collision avoidance systems that an automatic recovery maneuver shall not place the aircraft in an uncontrollable state. It also traces to H4 and C4 in the hazard analysis.]

[RMC05b] The automatic maneuver shall retain controllability of spacecraft: i.e. the reachable state of the maneuver shall not intersect the reachable state of uncontrollable maneuvers.

Rationale: The maneuver should not result in an uncontrollable spacecraft state. [This requirement traces to inspiration from aircraft collision avoidance systems that an automatic recovery maneuver shall not place the aircraft in an uncontrollable state. It also traces to H4 and C4 in the hazard analysis.]

[RMC07a] The automatic maneuver shall terminate (maneuver completed m) as soon as aor the state is in the safe set of final conditions $x_{m_{fin}} \in \mathcal{X}_{fin_{safe}}$.

Rationale: An automatic maneuver will activate when a threat condition (such as a collision, or loss of constellation position) is present, and this requirement bounds termination time. [This requirement traces to the hazard analysis H9.]

[RMC07b] The spacecraft shall maneuver within the safe set of initial maneuver states $x_{m_{init}} \in \mathcal{X}_{init_{safe}}$.

Rationale: There may be instances where it is only appropriate for a spacecraft to maneuver at a specific location, such as apogee or perigee in an orbit or at a specific location relative to another spacecraft. [This traces to the do not harm high-level requirement in aircraft collision avoidance systems].

[RMC07c] The automatic maneuver $u_{trajectory}$ shall be selected from a set of predefined templates $\mathcal{U}_{trajectory}$.

Rationale: Predefined templates reduce computation time. In addition, a simple maneuver can be more easily verified and makes the system transparent to the operator. [This requirement traces to *transparency* and *simple maneuver* needs inspired by aircraft systems.]

Do Not Interfere

This section includes maneuver selection design specifications and requirements intended to prevent the collision avoidance system from interfering with the primary mission of the spacecraft when possible.

Table 5.6: Formal expression of do no harm, maneuver constraint requirements.

Requirement ID	Requirement
RMC06	$\varphi_{RMC06} = \Box cooperative = cooperative_{obstacle} \leq T_{a_r} \implies c$
RMC07	$\varphi_{RMC07} = \Box(T_m \geq T_{m_{max}}) \implies m$
RMC08	$\varphi_{RMC08} = \Box(T_{p_m} \leq T_{m_{max}})$
RMC09	$\varphi_{RMC09} = \Box f_{m_p} \leq f_{m_{max}})$
RMC10	$\varphi_{RMC10} = \Box(DLMode == M) \implies (x_i \geq \mathcal{X}_{i_c}(t) - x_{i_t}) \wedge (x_i \leq \mathcal{X}_{i_c}(t) + x_{i_t})$
RMC11a	$\varphi_{RMC11a} = \Box appendage \implies r_{modeled} == r_{SC} + r_{appendage}$
RMC11b	$\varphi_{RMC11b} = \Box \neg appendage \implies r_{modeled} == r_{SC}$
RMC12	$\varphi_{RMC12} = \Box m \implies \Diamond^{\leq T_{ror}} \mathcal{X}_{op}$
RMC13	$\varphi_{RMC13} = P_{c_{collateral}} \leq P_{c_{collateral_{max}}}$

Since these requirements do not assume a controller design, to capture requirements while allowing for the design freedom, several assumptions and placeholders were included in SpeAR. During the design process of an automatic maneuver system, these values should be computed and used to replace the constant values in SpeAR. In the SpeAR specification, values for T_m, $T_{m_{max}}$, T_{p_m}, f_{m_p}, and $f_{m_{max}}$ are assumed to be equal to constant values. Desired position and velocity values ($\mathcal{X}_{i_c}(t)$) are chosen to be arbitrarily small constants, while acceptable offsets (x_{i_t}) between desired and actual states are assumed to be excessively large constants. For example, the desired position and velocity are for an ellipse starting at 200 meters in each direction from the origin of the relative motion reference frame, and the allowable offset is larger than what is valid to assume for linear dynamics. Similar assumptions and placeholders are used for operational limits on the spacecraft state.

[RMC06] The automatic maneuver shall update expected cooperation level of the collision threat object based on knowledge of that object.

Rationale: If two spacecraft are trying to automatically maneuver with different rule sets, it could lead to a hazardous situation. For example, if two spacecraft both avoid collision by gaining altitude, both could maneuver to cause a collision. If both spacecraft have the same software on board, they could both maneuver in a manner that is most efficient for both, possibly maneuvering later or conserving fuel. [This requirement traces to inspiration from aircraft systems that considered *cooperation*].

[RMC07] The automatic maneuver shall terminate (maneuver completed m) as soon as a maximum maneuver duration has been met $T_m > T_{m_{max}}$ has been met (or the state is in the safe set of final conditions $x_{m_{fin}} \in \mathcal{X}_{fin_{safe}}$).

Rationale: An automatic maneuver will activate when a threat condition (such as a collision, or loss of constellation position) is present, and this requirement bounds termination time. [This requirement traces to the hazard analysis H9.]

[RMC08] The automatic maneuver shall never exceed threshold duration; i.e. the planned duration of the maneuver T_{p_m} is less than some maximum maneuver time requirement $T_{m_{max}}$

Rationale: Provides hard constraint on maneuver duration as a fault tolerance measure. Allows operator time to monitor system and detect faults. [This requirement traces to the hazard analysis H9.]

[RMC09] The predicted fuel use f_{m_p} for an automatic maneuver shall be less than the maximum fuel use limit for any maneuver $f_{m_{max}}$.

Rationale: Provides fault tolerance to limit amount of fuel used in any one case. Allows operator time to monitor system and detect faults. [This requirement traces to the hazard analysis H9.]

[RMC10] The individual elements of the spacecraft state x_i at time t ($x_i(t)$) during automatic maneuvers $DL_{Mode} == M$ shall be within a +/- threshold x_{i_t} of the commanded value $\mathcal{X}_{i_c}(t)$(timing, position, velocity, acceleration, attitude, attitude rates, and angular accelerations).

Rationale: This design requirement describes a cyber-physical control system tolerance to accounts for some allowable sensor and actuator uncertainty. [This requirement is inspired by accuracy requirements in aircraft collision avoidance systems.]

[RMC11] The automatic maneuver system shall adapt the modeled radius of the spacecraft r_{sc} according to the state of appendage deployment $appendage$.

Rationale: For example, if retracting appendages such as solar panels reduces the drag (small drag is primarily from the upper atmosphere in low orbits or from solar radiation pressure), and any uncertainty associated with that drag, the trajectory prediction should be updated to reflect these changes. This requirement enables more complex automatic maneuver systems in the future to include drag models for various trajectory predictions. [This requirement traces to H3 and C3 in the hazard analysis.]

[RMC12] The automatic maneuver shall allow the spacecraft to stay in operational position and attitude or return to operational position and attitude within interference time threshold; i.e. the spacecraft state $x(t)$ will return to an acceptable operational state $\mathcal{X}_{op}$ within a post maneuver return to operations time requirement T_{ror}.

Rationale: Depending on the system and mission, the automatic maneuver should either have no interruption of operations, or minimal acceptable interruption of operations. [This requirement traces to the nuisance free criteria in the aircraft collision avoidance systems].

[RMC13] The probability of a collateral collision $P_{c_{collateral}}$ resulting from an automatic maneuver shall be less than a maximum collateral probability of collision requirement $P_{c_{collateral_{max}}}$.

Rationale: In order to maneuver, the probability of the maneuver have some collateral collision probability with some other tracked object should be minimized. The maneuver should look ahead to see how maneuvering changes the probability of collision with other objects. [This requirement traces to the nuisance free criteria in the aircraft collision avoidance systems].

5.3.4 Maneuver Selector Design Specifications and Requirements Formalization

The maneuver selector component determines when a collision is imminent. In some cases, it may also select from a predefined set of candidate automatic avoidance maneuvers to recommend to the maneuver controller. The numbering on these requirements is continued to allow for the controller and selector to be more easily combined in some instances.

[RMS14] The system shall maneuver when the perceived time available t_a is less than a time available

Table 5.7: Formal expression of do no harm, maneuver constraint requirements.

Requirement ID	Requirement
RMS14a	$\varphi_{RMS14a} = \Box t_a \leq T_{a_r} \implies c$
RMS14b	$\varphi_{RMS14b} = \Box t_a > T_{a_r} \implies \neg c$
RMS15	$\varphi_{RMS15} = \Box(sys_{risk} == op_{risklevel})$
RMS16	$\varphi_{RMS16} = \Box r_b \leq r_{b_{max}}$
RMS16l	$\varphi_{RMS16l} = \Box sys_{risk} == LOW \implies r_{b_{max}} == r_{b_{LOW}}$
RMS16m	$\varphi_{RMS16m} = \Box sys_{risk} == MEDIUM \implies r_{b_{max}} == r_{b_{MEDIUM}}$
RMS16h	$\varphi_{RMS16h} = \Box sys_{risk} == HIGH \implies r_{b_{max}} == r_{b_{HIGH}}$
RMS17	$\varphi_{RMS17} = \Box P_c \geq P_{c_{min}} \implies c$
RMS18	$\varphi_{RMS18} = \Box r_{obs} == r_{sc_i}$
RMS19	$\varphi_{RMS19} = \Box \mathcal{X}_{obs} == \mathcal{X}_{sc_i}$
RMS20	$\varphi_{RMC20} = \Box r_{modeled} == r_{sc}$

requirement threshold value T_{a_r}; i.e. when the time available t_a is less than or equal to the time available requirement threshold T_{a_r}, the maneuver selector shall set the condition to maneuver flag c to true.

Rationale: The system should only maneuver when absolutely necessary and minimize time of service disruption. The time available requirement could be based on a variety of approaches similar to the discussion in C2 of the hazard analysis. [This requirement traces to H2 and C2 in the hazard analysis as well as the *do not interfere* need inspired by aircraft systems.]

[RMS15] The system shall set risk to ground operator determined risk acceptance thresholds.

Rationale: The operator's risk tuning would result in larger or smaller reachable set for collision predictions. This is a sister of the RGC09 requirement which sets this value on the ground computer and uplinks it to spacecraft. This requirement updates the value on the spacecraft based on the model. [This requirement traces to the *risk level selection* need inspired by aircraft collision avoidance systems.]

[RMS16] The uncertainty buffer around each object shall remain below risk threshold specified for the risk setting; i.e. the additional radius buffer r_b added to each spacecraft's radius r_{sc} is less than a maximum buffer radius $r_{b_{max}}$.

Rationale: There is an assumption here that a safety buffer will be added to the radius of the satellite to define a safe occupancy region. Without a threshold limit on uncertainty, a

spacecraft reachable space could be too large to be useful for collision avoidance. [This requirement traces to the *risk level selection* need inspired by aircraft collision avoidance systems.]

[RMS17] (Collision Avoidance Maneuver Specific) The probability of a collision P_c shall be greater than a probability of collision required for maneuvering $P_{c_{min}}$ to trigger the condition to maneuver.

Rationale: The probability of collision metric is used in current operations to determine if a maneuver should be conducted to avoid a maneuver. It is assumed that some version of probability of collision metric will play into the spacecraft maneuver selection logic. [This requirement traces to the nuisance free operations criteria inspired by aircraft collision avoidance systems.]

[RMS18] The automatic maneuver system shall accept as input the radius of obstacles within local neighborhood r_{sc_i}.

Rationale: Knowing the radius of obstacles within a neighborhood (is it a CubeSat or the ISS?) helps to predict the reachable space and may allow closer approach distances. [This requirement traces to the nuisance free operations criteria inspired by aircraft collision avoidance systems.]

[RMS19] The automatic maneuver system shall accept as input the state $\mathcal{X}_{sc_i}$ (at least position, velocity, but potentially other state information) of spacecraft in a predefined neighborhood.

Rationale: A neighborhood may be defined as a limited number of satellites that will pass within a threshold distance and time horizon. [This requirement traces to scalability, inspired by aircraft midair collision avoidance systems.]

[RMS20] The automatic maneuver system shall accept as an input the radius r_{sc} of the spacecraft.

Rationale: The radius of the spacecraft for a specific configuration should factor into the automatic maneuver algorithm. [This requirement traces to *special configurations* requirements in aircraft collision avoidance systems.]

Non-Behavioral Requirements

The following design requirements should also be considered, because they describe performance and not behavior, or the behavior is tangential to the operation of the system (such as logging data), it is not formalized in this work. These requirements are not formalized as the method of ranking threats and determining collisions is left for future work. All of these requirements are inspired by aircraft collision avoidance systems. Since these requirements are not formally specified, alternative verification approaches are recommended.

[M17] Automatic maneuver system shall use best available information to determine the need to maneuver.

Rationale: Multiple sensing and information sources may provide similar or conflicting information. A fusion or prioritization strategy should be employed. Best available information will need to be encoded a priori. For instance, if onboard LIDAR and uplinked TLE information are both uploaded, LIDAR may be the best available information.

Verification Method: Inspection.

[M21] The system shall log maneuver data.

Rationale: Logging data allows for further study and improvement of automatic maneuver systems.

Verification Method: Simulation and Hardware in the Loop Test.

[M22] The system shall transmit maneuver data to ground station within transmit time horizon.

Rationale: Some flexibility should be given to not interrupt other messages with maneuver data; however maneuver data should be collected in ground-based repositories to prevent data loss.

Verification Method: Simulation and Hardware in the Loop Test.

[M23] The logged data shall be recorded at a rate greater than or equal to a threshold value.

Rationale: Data rate should be selected to provide sufficient post-maneuver, ground-based analysis, while not overtaxing spacecraft on board resources.

Verification Method: Inspection. Simulation and Hardware in the Loop Test.

[M26] The automatic maneuver system shall adjust trajectory prediction uncertainty at uncertainty update rate.

Rationale: For instance, in a collision avoidance scenario, reducing uncertainty, reduces reachable set and allows objects to approach closer, and possibly a more conservative collision avoidance maneuver to be employed.

Verification Method: Inspection, analysis, and hardware in the loop test.

[M33] The automatic maneuver system shall accept as input the state and radius of a threshold number N_{sc} of objects within local neighborhood.

Rationale: Being able to accept the state and radius of multiple objects helps when avoiding a collision with one object could result in a collision with another object.

Verification Method: Inspection.

[M34] The automatic maneuver system shall predict trajectory for a threshold number of spacecraft N_{sc}.

Rationale: In order to predict collisions, the system must be able to predict trajectories of possible colliding objects.

Verification Method: Inspection.

[M35] The automatic maneuver control system redundancy shall be equivalent to the primary control system redundancy.

Rationale: If the control system (computing hardware) is redundant, the system should have an equivalent level of redundancy to aid in fault management.

Verification Method: Inspection.

[M36] The automatic maneuver system shall compare predicted trajectories with a rate greater than or equal to a threshold value.

Rationale: Trajectories should be compared at a rate fast enough so that tunneling (missing a collision prediction because the timestep is too large) doesn't occur. In addition, it allows for more appropriate avoidance maneuver selection criteria.

Verification Method: Inspection.

[M37] The automatic collision avoidance maneuver system shall prevent a threshold percentage of

known historical collision cases.

Rationale: Metric for "success" of avoidance maneuver.

Verification Method: Simulation and Hardware in the Loop Test.

[M38] The automatic collision avoidance system shall avoid digital virtual obstructions (for testing purposes).

Rationale: Testing purposes.

Verification Method: Simulation and Hardware in the Loop Test.

5.4 Analysis of Design Specifications and Requirements

5.4.1 Requirement Analysis with QVScribe

Early in the research, an initial set of 82 natural language requirements was analyzed with a trial version of QVScribe. The requirements were evaluated using both the word processor and spreadsheet versions of the tool. From installing the tool to initial analysis results took less than ten minutes. The quality analysis in QVScribe identified problems with 43 of the 82 natural language requirements divided into 3 categories: negative imperatives, vague words, and universal quantifiers. A summary of the findings and the decisions to edit the requirements based on them are as follows:

- *negative imperatives*: In 14 cases, a requirement stated that the system "shall not" rather than "shall," which is a requirements best practice. All 14 requirements were changed from describing undesirable conditions to a maintaining a desirable condition or specific behavior that should prevent an undesirable condition. For example, a requirement that stated the system "Automatic maneuvers shall not exceed a threshold fuel consumption in a specified time window" was changed to "Automatic maneuvers shall remain below the threshold fuel consumption limit within a specified time window."

- *universal quantifiers*: The word "no" appeared in 10 requirements, and it was decided to keep them because they appropriately describe the necessary absence of conditions that trigger transition between states.

- *vague words*: QVScribe identified "should" (6 cases), "be able to" or "be capable of" (7 cases), "its" (6 cases), "all" (2 cases), "feasible," "reasonable," "appropriately," and "to re-

flect" as vague words in the requirements. All cases of "should" were changed to "shall," instances of "all" were deleted, and "its" were changed to specific component names. Requirements containing "be able to" or "be capable of" were modified to describe how the capability was defined. For instance, "The ground operator shall be able to tune uncertainty/risk acceptance thresholds" was updated to "The system shall set risk to ground operator determined risk acceptance thresholds." Cases of "feasible," "reasonable," "appropriately," and "to reflect" were changed to meet a specific criterion. For example, one requirement that stated "as quickly as feasible" was changed to "within the time requirement."

In the process of using QVScribe, the following strengths and weaknesses of the tool were identified:

- Strengths
  - Installation was easy and operation was intuitive.
  - Quick and accurate identification of requirements. While many requirements documents include other descriptive information, QVScribe allows for identification and analysis of just the requirements in that document, so there is no need for a separate version of the requirements just for analysis.
  - It works with word processor and spreadsheet-based requirements which are very common requirements capture methods.
  - While someone could theoretically check their document for trigger words, it is very labor intensive and time consuming and QVScribe can check for all of them in seconds.
  - Identifies and checks for consistency in terms and units.
  - Identifies similar terms which helps to identify conflicts in requirements, or the use of the same term for multiple concepts.
- Weaknesses
  - Limited analysis capability for advanced users.
  - The consistency of terms was not included in an analysis report output by QVScribe. However, when this feedback was provided to QRA, they responded that the capability was in development already for another release.

Analysis in QVScribe was helpful to provide initial feedback on natural language requirements before formal specification activities commenced.

5.4.2 Requirement Analysis with SpeAR

Specification in SpeAR

Several limitations in SpeAR restricted the specifications. For instance, nonlinear requirements based on norms such as the magnitude of a distance, velocity, or acceleration were not possible, so instead linear inequality constraints were used in their place. Also, the collision avoidance maneuvers were assumed to be in translational motion only, so attitude dynamics were not specified. However, attitude requirements were formalized as observers so that they could be utilized later with the addition of attitude dynamics.

Several conventions and features of SpeAR were used in the specification. A *definitions* file was used to capture units, types, constants, and patterns. Units were used throughout the specification and SpeAR completed unit consistency checking. Specific types were created to describe different variables throughout, and some used type primitives such as defining a mass type that is a real number greater than or equal to 0.0 kilograms (`massT is a {i:real |i >= 0.0} kg`). These type primitives are checked as part of the logical entailment analysis. Other types took the form of enumerated types. For instance, risk level is defined as an enumerated type with low, medium, and high values (`risklevelT is an enum LOW, MEDIUM, HIGH`). More complex types are used to store state information. For instance, the state of spacecraft within the local neighborhood of the ownship spacecraft are stored as a record of local state records, each containing the position, velocity, angular position, angular velocity, whether the spacecraft is cooperative, and the radius of the spacecraft. Constant variables are captured in all capital letters and include initial conditions for the spacecraft physical states (position, velocity, etc.) and system states (power on, risk level, etc.). Patterns captured in SpeAR were functions like computing the minimum of 3 values, the time since the last maneuver, or the cumulative maneuver time.

A set of *specification* files were generated for each component, as described in Fig. 5.26, with connections described earlier in Fig. 5.22-5.23. Each specification file starts by importing the definitions file and the specifications of its subsystems. Next inputs and outputs are described as well

as any local state variables. A macros section is used to track previous states and specify some of the labeling function. Assumptions are used to constrain internal state variables or input conditions. Finally the design specifications and requirements are documented. A separate file was created to capture the safety constraints to ensure logical consistency, and the constraints dealing with translational motion were included in the main system specification.

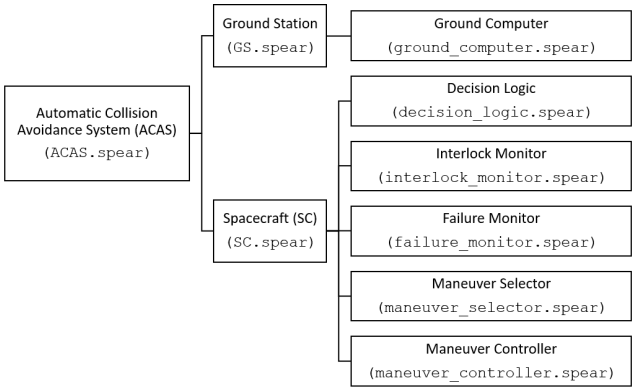


Figure 5.26: SpeAR specification files.

Analysis

The formal requirements were documented and analyzed in SpeAR for logical consistency, logical entailment, realizability, and in limited cases traceability. The summary of the results of this analysis are presented in Table 5.8.

Table 5.8: SpeAR Analysis Status

Component	Specified	Logical Consistency	Logical Entailment	Realizability
Hazards	✓	✓	NA	NA
Definitions	✓	NA	NA	NA
ACAS (top level)	✓	✓	✓	?
GS (level 2)	✓	✓	✓	✓
Ground Computer (level 3)	✓	✓	✓	?
SC (level 2)	✓	✓	✓	?
Decision Logic (level 3)	✓	✓	✓	✓
Interlock Monitor (level 3)	✓	✓	✓	✓
Maneuver Selector (level 3)	✓	✓	✓	?
Maneuver Controller (level 3)	✓	✓	✓	?

The logical entailment analysis proved at every level of the design, indicating that the RTA

architecture and design specifications satisfied the formal requirements of the system. Logical consistency also proved for every file, indicating there are no conflicts in the specifications. A few of the files were undecided in realizability, meaning that a counter example wasn't found but SpeAR wasn't able to prove realizability. It is possible that these files are realizable. In addition, SpeAR analyzed unit consistency across all variables and equations and all units are consistent throughout.

5.4.3 Evaluation of Requirements on Representative System

During a one week Tech Sprint in January 2020, the requirements presented in this chapter were evaluated for applicability to a real RTA system for spacecraft. During the week, researchers from the Air Force Research Laboratory (AFRL), Verus Research, and the Johns Hopkins Applied Physics Laboratory investigated what it would take to adapt an RTA system developed and flown on aircraft to spacecraft. During the Tech Sprint six subject matter experts from the AFRL's Aerospace Systems and Space Vehicles directorates evaluated applicability of the requirements for use on an on-orbit space platform. In particular the list of requirements from the literature and the hazard analysis were attractive because such a comprehensive list was not previously generated. No flaws in the requirements were identified. The work presented in this dissertation and the Tech Sprint were used to develop a research proposal for development of an on-orbit autonomy testbed. The general architecture presented in Chapter 4 is envisioned on the proposed platform.

5.5 Summary

This chapter developed and analyzed formal design specifications and requirements for a hypothetical spacecraft last instant collision avoidance system. Scoping of the collision avoidance problem to preventing collisions during rendezvous and docking enabled the use of linearized dynamics and assumed escape trajectories based on natural motion trajectories. By examining regulations and standards, previous literature, conducting a hazard analysis, and considering requirements inspired by aircraft collision avoidance systems, a set of formal design specifications and requirements were developed. Rationale and traceability was provided for all identified hazards, safety constraints, and requirements. For non-functional requirements that are not formalized, a verification approach was recommended. These requirements were analyzed using two different tools. QVScribe analyzed

the natural language requirements to identify negative imperatives, universal quantifiers, and vague words. Then SpeAR was used to analyze logical consistency, logical entailment, realizability, and traceability. Finally, the requirements were evaluated during a one week Tech Sprint with experts from government and industry to determine applicability to a practical RTA for spacecraft.

CHAPTER 6
CONCLUSIONS AND RECOMMENDATIONS

Backup safety systems like automatic collision avoidance will be vital for verifying, validating, certifying, and assuring advanced, complex, and increasingly autonomous control system designs. While a variety RTA-style approaches have been used to develop ad hoc and domain specific designs, in this research it was hypothesized that a common RTA architecture and set of requirements patterns would apply across domains. Additionally, it was hypothesized that this requirements could be formally specified and analyzed using formal methods. Auto GCAS was studied for vital design elements in Chapter 3, which were generalized in Chapter 4, and then demonstrated on a hypothetical spacecraft last instant automatic collision avoidance system in Chapter 5.

6.1 Conclusions

The contributions of this dissertation are discussed below.

- A case study of the Automatic Ground Collision Avoidance System (Auto GCAS) as a successful RTA system synthesized information from engineering and psychology studies to identify critical components of a general aerospace automated collision avoidance RTA, forming the basis for the architecture portion of the design approach. The generalized inclusion of fault monitor, interlock monitor, and human-machine interface components in an RTA architecture was identified as critical in the Auto GCAS system. The importance of these components was echoed in the development of the requirements of the hypothetical spacecraft collision avoidance system.

- While RTA, formal methods, and STAMP/STPA hazard analysis have all been done independently, this was the first demonstration of how RTA boundary violation safety requirements could be developed and formally analyzed based on top-down system safety analysis.

- The viability, flexibility, and capability of the proposed RTA architecture and requirements elicitation, specification, and analysis approach was evaluated in the development of a hypo-

thetical automatic collision avoidance system for spacecraft.

- Prior to this research, a cohesive set of requirements for automatic maneuvering of spacecraft did not exist in regulations, standards, or literature. This dissertation is the first development of a set of safety requirements for automated or autonomous maneuvering of spacecraft. The requirements from the case study were evaluated for applicability to a practical spacecraft RTA solution and found to be relevant, useful, and correct.

- This dissertation evaluated the application of two requirements analysis tools applied to the requirements and architecture. Recommendations for improvements were communicated to the developers of both tools.

6.2 Recommendations for Future Work

6.2.1 Integration of Multiple Backups

An area of active research for the development of RTA systems is determining the best approach to integrate multiple safety recovery controllers. In some cases, these backup controllers protect against different safety boundaries. For instance, Auto GCAS monitors trajectories that intersect digital terrain elevation data and prevents ground collisions, Auto ACAS monitors the trajectories of the ownship and other aircraft to prevents midair collisions, and Geofencing monitors the aircraft trajectory within three dimensional physical boundaries for violations. Each of these boundaries are disjoint; however it is possible that violations of multiple boundaries could be predicted to occur at the same time.

AFRL's approach to this problem has been to combine multiple RTA systems as an Integrated solution. The Automatic Integrated Collision Avoidance System (Auto ICAS) [161, 162] integrates Auto GCAS and Auto ACAS into a comprehensive collision avoidance system. The advantage of this is that each system can consider possible conflicts between boundary violations and select a better solution than either might select on their own. For instance, in a situation where an aircraft flying close to the ground is on a collision course with another aircraft, a ground-aware Auto ACAS will filter from the nine possible maneuvers to select maneuver that avoids the midair collision while at the same time ensuring none of those maneuvers will fly the aircraft into the ground. An

integrated solution has the benefit of generating a better solution than either separate system might otherwise choose. However, this increase in performance also comes at a large increase in cost and schedule. The Auto ICAS program was the same order of magnitude of cost of either the Auto GCAS or Auto ACAS. It is unclear at this point whether the decision to integrate solutions with this approach will scale linearly or exponentially with the number of systems integrated together; however, if the growth of complexity in military aircraft development and accompanying exponential growth in program cost and schedule over the decades is any indication, the latter is more likely. In addition, when a new system must be integrated, or an existing system upgraded, then entire integrated solution will need to be reevaluated as part of the certification process.

An alternative approach proposed in the NASA/FAA/DoD Resilient Autonomy program is to use an RTA network architecture where each boundary monitor and recovery controller function is separate [272]. On the one hand, this modular approach eases certification of each individual component and facilitates quick integration of new components and upgrades to legacy components. On the other hand, additional stress is placed on the certification of the decision module/switching component to determine which action to take in the case where multiple safety boundaries may be violated at the same time. The Resilient Autonomy program's answer to this challenge is to approach the decision like a human pilot, who responds to the most critical boundary violation first before moving onto the next violation. They recommend that decisions be based on standards, though the author of this work proposes this prioritization would also be a function of owner/operator/pilot preference and the output of the hazard analysis. A disadvantage of the modular approach is that it is possible to violate a safety boundary. For instance, if a ground collision and geofence collision were both eminent, the system might decide to engage Auto GCAS to first clear the threat of ground collision before engaging a geofence controller and violating the geofence boundary. Depending on the scenario, this may or may not be acceptable. An integrated solution in the same vein as Auto ICAS on the other hand could adjust the boundary of a geofence controller based on awareness of Auto GCAS and Auto ACAS, or could select a collision avoidance maneuver that also kept the aircraft within the geofence.

6.2.2 Investigation of Taylor Flowpipes for Long Duration Orbital Uncertainty Propagation

An abstraction technique is needed in which reach-set growth is dominated by perturbation forces (process noise including gravitational variation, atmospheric drag, and unknown forces such as radiation pressure, etc.) and/or errors from measurement uncertainty (measurement noise from measurement frequency & quality of measurements), rather than growth from the abstraction technique. Taylor models and Taylor flow pipes [273, 274] are able to model the nonlinear dynamics more directly and can be used to represent non-convex reachable sets as shown in 6.1. It may be possible to use Taylor methods to propagate a reachable space that looks close to a Gauss von Mises Filter as shown in Fig. B.3.

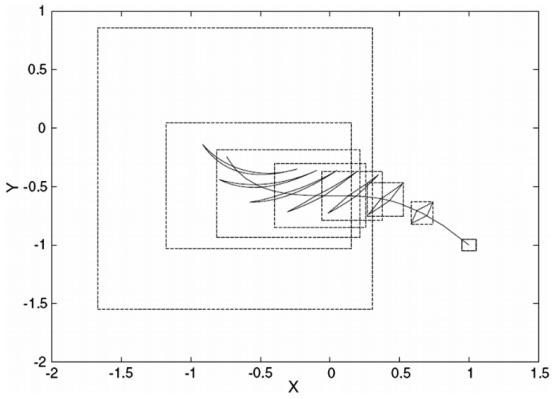


Figure 6.1: Taylor Flow pipe Example [273]

6.2.3 Fault Model Development

One of the elements of this analysis that was left for future work is the development of a fault model and fault monitoring to inform when the system should go into a failed state. This analysis should account for sensor faults, actuator faults, and computation faults with causes such as single event upsets. While there are many promising approaches, linear temporal logic monitors with Bayesian reasoning for isolation have been used in aircraft [275, 276] and compliment the formal approach used in this thesis.

6.2.4 Missed Detections and False Alarms

This research focused on the decision to maneuver assuming that a maneuver would be conducted if the probability of collision was sufficiently high. However, an important design consideration in any collision avoidance system is the acceptable rate of missed detections and false alarms, which can also be thought of as Type 1 and Type 2 errors in statistical hypothesis testing. Hypothesis testing is a formal way of determining whether a system meets a specification or requirement [277]. Each hypothesis tested consists of a null hypothesis H_o, such as a collision is not imminent, and an alternative hypothesis H_1, such as a collision is imminent. As summarized in Table 6.1, a false alarm or Type I error occurs when a collision avoidance system maneuvers to avoid a collision that would not have happened, and a missed detection or Type II error occurs when a collision avoidance system does not maneuver to prevent a collision.

Table 6.1: Collision Detection Type I and Type II Errors

	H_0 Collision Not Imminent	H_1 Collision Imminent
Do Not Reject H_0 System Decides Not Imminent	Correct	Type II Error False Negative Missed Detection β probability
Reject H_0 System decides Collision Imminent	Type I Error False Positive False Alarm α Probability	Correct

The rates of missed detections and false alarms tie directly to the nuisance criteria of the collision avoidance system. In Auto GCAS the acceptable rate of Type I Errors, maneuvering to avoid a collision when a collision was not imminent, was very low because the consequence of an unnecessary interruption was high. Type II errors, failing to maneuver to prevent an imminent collision, may be more acceptable depending on the risk tolerance of the application and consequences of interrupting the primary function of the system. In the spacecraft domain, while a Type I error could interrupt the primary mission of the spacecraft, the bigger consequence is the use of irreplaceable fuel to avoid the collision. However, the consequences of a Type II error are very high in the space domain. Collisions not only result in the loss of the spacecraft, but create an environmental hazard

in the form of thousands of pieces of debris that take years to millennia to deorbit depending on altitude. Future studies are required to determine acceptable α and β rates for spacecraft collision avoidance.

6.2.5 Defining "Do Not Interfere" Criteria

Related to the topic of missed detections and false alarms is the challenge of defining nuisance-free or "do not interfere" criteria. In Auto GCAS, an experiment was conducted to identify an appropriate time available metric. If the system maneuvered to avoid a collision before this time, it would be viewed as an unnecessary interruption to normal operations. While this criteria may be applicable to a highly maneuverable aircraft, the same metric may not be appropriate for commercial aircraft, urban air mobility, or package delivery drones. Likewise in the space domain, a criteria developed for a highly capable autonomous satellite servicing or active debris removal satellite may not be appropriate for other satellites with various capability levels. In the space domain, there is a different cost/benefit analysis on the need to maneuver because fuel is a valuable and limited resource. Defining an appropriate metric or set of metrics is a ripe area for additional research.

6.2.6 Interlocks and Deadlock Avoidance

In the process of designing interlock conditions, it is important to ensure the system is free from deadlock conditions. A deadlock condition occurs when two or more components wait on each other to progress [13]. A famous example of this in concurrent systems is the Dining Philosophers example [278, 279, 280], where five philosophers share five total chopsticks and an request and release approach must be developed so that each philosopher can eat and think infinitely often. Each philosopher needs two chopsticks to eat and a deadlock occurs when each philosopher has one chopstick. When designing compositional systems, like those in this research, where multiple subsystems compute in parallel and depend on each other to progress. However, deadlocks were not specifically analyzed in this research.

6.2.7 Different Approaches to Run Time Assurance for Inner Loop versus Outer Loop Control

The approach to RTA and requirements development introduced in this thesis was developed primarily for outer loop control. As discussed in [11], RTA switching in outer loop control systems

is intended to only occur rarely, no more than once per flight in aircraft, and only for rare events in spacecraft. However inner loop RTAs like those developed for engine control [157, 158], may switch controllers more frequently and a hybrid-systems analysis process may be more appropriate to ensure frequent switching cannot lead to instability in the system.

6.2.8 Alternative Requirements Analysis methods

The analysis in this research was limited to searching for specific words in QVScribe and analysis of linear design specifications and requirements in SpeAR. More capable formal methods tools such as theorem provers could be applied like PVS, which has been applied to properties of detect and avoid systems [219], or ACL2, which has been applied to aerospace design [128, 129].

6.2.9 Computational Concerns for CPS

This research primarily focused on the physics concerns for RTA. However, much research is needed into methods to synthesize monitors that enforce desired requirements (system properties). Some considerations have been made in this research when defining requirements so they could be analyzed using open source SMT solvers. Several steps exist between this and writing executable software for embedded systems and the topic of synthesis and software verification is left for future work. In [256], considerations are made for signal routing, real time feedback and control concerns, and real time communication.

6.2.10 Resolution and Accuracy

Resolution and accuracy of the collision predictions are an important factor in the design of automatic systems that was not explored in this research. A study could be conducted to better understand the state of the art available and practical onboard spacecraft as well as the minimal resolution required.

6.2.11 Other practical considerations

Beyond the considerations in this thesis, several other practical design features, especially in terms of computation and criticality should be considered. For instance, trustworthy monitors must be

independent of what is monitored, which could be implemented in hardware and/or software isolation. In addition safety criticality of different functions could dictate where they reside in both hardware and software. Safety critical functions must be separate from lower criticality functions and the criticality of the RTA monitor and backup much be equivalent to the critically of the primary control system. In addition data transformation assurance should be generated to ensure that complex trigonometric functions, coordinate and data transformations, error analysis are correct. Some functions such as unit checking are completed in SpeAR.

Appendices

APPENDIX A

TIME-BASED AEROSPACE COLLISION AVOIDANCE SYSTEM TAXONOMY

As the spacecraft environment becomes increasingly congested and contested [281], the speed of human-driven spacecraft maneuver decisions becomes insufficient to meet mission assurance requirements. A growing population of millions of pieces of space debris [282] and concerns over the possibility of cascading collisions between artificial space objects [283, 284] highlight the need for collision avoidance as a critical component of any future Space Traffic Management (STM) system [48, 46]. Spacecraft operations today, including collision avoidance, are developed by human teams days in advance [20]. Automation and autonomy in spacecraft decision and control offer more agile and responsive approaches to collision avoidance, especially without limitations of latency or ground communication windows. However, while there are handbooks [43, 42] and limited regulations [45, 242] for spacecraft operation, no general guidance is provided to define requirements for automatic or autonomous maneuvering. This appendix introduces a time-based categorization of aerospace collision avoidance systems. While air and space examples exist for all four categories, there are no space domain examples for the most automated two categories. Analysis of air domain last instant collision avoidance systems provides insight into high-level requirements for short time horizon collision avoidance in the space domain.

A.1 Related Work

Previous research has categorized conflict detection and resolution approaches, reviewed operational ground collision avoidance systems, and surveyed spacecraft conflict detection and resolution. An influential review of 68 conflict detection and resolution systems for aircraft, robotic, automobile, and naval applications in [285] first categorized approaches based on the state propagation method (nominal, worst case, or probabilistic), then further classified using each system's state information dimensions [vertical (V), horizontal (H), or three-dimensional (HV) (3-D)], conflict detection threshold, conflict resolution method (prescribed, optimized, force field, or manual), maneuvering dimensions (speed change, lateral, vertical, or combined maneuvers), and management

of multiple aircraft conflicts (pairwise or global). At the time the review was published, avoidance maneuver execution largely relied on a human operator. In contrast, the classifications proposed in this paper consider whether automation or a human engages the maneuver. In addition, while the previous work focused primarily on distance metrics for detecting a collision (with time as a secondary consideration), the categories in this paper are based on time metrics and then further classifications are made based where computation takes place (on-board or ground-based), whether or not the maneuver is automated, the relative level of uncertainty associated with the propagation, and the relative amount of fuel used.

In addition, this paper is scoped to focus only on operational air and space collision avoidance systems, or systems under development with an intention to be used in operational systems and a strong emphasis in formal methods verification. An updated thorough review of operational ground collision avoidance systems for aircraft was completed in [190]; however, this research focuses on only one of these examples for last instant collision avoidance. A survey of spacecraft conflict detection and resolution was conducted [222]; however, this research focuses on collision avoidance maneuvering rather than the detection problems. This research focuses on conceptual requirements for the decision and control to automatically maneuver to avoid, while related survey papers [222] have focused more heavily on collision detection and prediction, which is often more challenging.

A.2 Taxonomy

To compare collision avoidance approaches across both the air and space domain, a time-based taxonomy is proposed whereby systems are divided into four categories: *strategic*, *tactical*, *detect and avoid*, and *last instant* based on factors summarized in Table A.1. In this section, each category is explored with a general description and common traits between the air and space domain, followed by mature (operational system or demonstrated in a relevant environment) examples from both the air and space domain or a discussion of applicable concepts in the absence of examples.

The four approaches to collision avoidance have different resource implications, particularly in how far into the future collision predictions are computed, where the computation of the prediction generally takes place, level of automation vs. human interaction, uncertainty, fuel usage and importance of minimizing mission disruption. For a scaling context between the categories, consider a

Table A.1: Summary of Collision Avoidance Categories for both Aircraft and Spacecraft

Category	Prediction Horizon	Computation	Automated Maneuver	Uncertainty	Fuel Use	Maneuver Description
Strategic	long term	on-board or ground -based	no	high	low	modification of complex flight or operational plans
Tactical	medium term	on-board or ground -based	no	medium	medium	simple maneuver to deconflict paths
Detect and Avoid	near term	on-board or ground -based	no (with exceptions)	low	high	simple maneuver to maintain minimum safe separation
Last Instant	imminent term	on-board	yes	very low	very high	automatic selection and engagement of preplanned maneuver from templates

rough comparison of representative collision avoidance system boundaries and prediction horizons as depicted in Fig. A.1. Note strategic collision avoidance predictions are two orders of magnitude larger than tactical collision avoidance and therefore excluded from this image. Also, the last instant collision avoidance prediction cones are barely visible as they are less than a third the length of the smallest, red detect and avoid safety volume. In both the air and space domain, human expertise

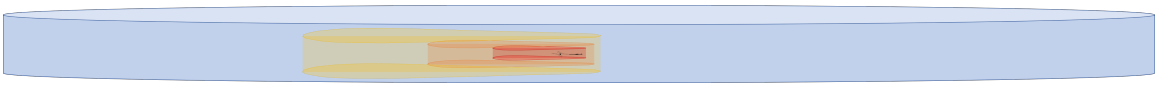


Figure A.1: Rough size comparison of air domain tactical Chorus "well clear" safety cylinder, detect and avoid TCAS II traffic advisory volumes, and a last instant collision avoidance Auto ACAS prediction cone.

(of pilots, traffic controllers, and operators) and fuel (particularly in the space domain) are valuable and limited resources [285, 20]. In both the air and space domains, the longer the position of a system is projected into the future, the greater the uncertainty becomes [285, 18, 222]. Maneuvering far in advance could conserve time or fuel for a maneuver [286]; however, under high uncertainty the maneuver would be very conservative and possibly unnecessary [287]. Decision makers must consider mission criticality and acceptable risks when determining how far to maneuver in advance

Table A.2: Air and Space Domain Category Time Horizon, Descriptions and Examples

Category	Aircraft Domain			Spacecraft Domain		
	Time	Description	Ex.	Time	Description	Ex.
Strategic	>10 minutes	modifies flight plans, 4D waypoints	NASA Stratway, NASA TASAR	>72 hours	modify operation plan	NASA CARA, ESA CRASS
Tactical	2-10 minutes	simple maneuvers based on position, velocity vectors	NASA Chorus	24-72 hours	Uses CDM, HIE plus manual maneuver planning	18SPCS CDM, NASA CARA
Detect and Avoid	15 seconds to 5 minutes	Directs action (climb, descend, etc); operator (or automation) executes	AFRL SAA, TCAS, ACAS-X, NASA DAIDALUS	1-24 hours	Directs action (ΔV, Timing, etc); operator executes	None known
Last Instant	<1-30 seconds	System automatically maneuvers without operator input	Auto GCAS, Auto ACAS	<1 hour to seconds	System automatically maneuvers without operator input	None known (focus of this research)

of an anticipated collision [191, 192, 17, 16]. Approaches that maneuver earlier often come at the expense of human-centered planning and instructions to maneuver but provide large margin and are generally low risk [288]. However, optimization of the long horizon maneuver planning can lead to more efficient paths [286]. Approaches that maneuver closer to the potential collision require automation (where human planning is too slow) in planning and maneuvering but provide smaller margins with higher risk [191]. Longer horizon predictions may lead decision makers to conduct unnecessary low fuel maneuvers under high uncertainty, while shorter horizon predictions may require more aggressive, high fuel maneuvers with greater certainty of position and velocity near the predicted collision. While slight changes to a flight plan earlier have little impact on missions such as the transportation of goods or people, some mission-critical operations should only be disrupted for a collision avoidance maneuver when a collision would otherwise be certain, such as in military

applications [191, 192, 17, 16].

Timelines, descriptions, and examples of collision avoidance categories in the air and space domain are summarized in Table A.2. Overlap in time until safety violation, collision, or closest approach between the different categories occurs because the systems that fit into each category feature some overlap. The times for each aircraft category are based on the types of tools that operate in those environments; for instance, the lower limit of the detect and avoid category is based on the lower limit of the TCAS resolution advisory time window [211]. Likewise, timelines in the spacecraft domain reflect timelines of the tools; for instance, NASA CARA predicts 7 days out (strategic), the 4th Air Force, 18th Space Control Squadron (18SPCS) CDMs are issued up to 72 hours in advance [18] (used to separate strategic and tactical). While spacecraft collision warnings can occasionally be received with less than 24 hours notice [20], this is rare and difficult to manage with human-centric processes.

A.2.1 Strategic Collision Avoidance Systems

Strategic collision avoidance approaches use long range predictions to modify complex flight or operational plans far in advance of a potential collision, a definition inspired by [289]. These modifications are done on the ground or on board [286] far enough in advance that they can generally be optimized.

Aircraft Strategic Collision Avoidance Systems

In the aircraft domain, much of the strategic collision avoidance function is provided by air traffic control which approves flight plans, and coordinates aircraft clearances for takeoff, departing an airport terminal area, flying in different altitude bands to avoid weather, entering an airport terminal area, and landing. Beyond human-centric planning, automated systems may aid in air domain strategic collision avoidance.

Two examples of aircraft domain system strategic collision avoidance system are the *NASA Traffic Aware Strategic Aircrew Request (TASAR)* program [286, 290, 291, 292] and the *NASA Stratway* program [289, 293]. The NASA TASAR program optimizes routes using data from onboard flight management systems like aircraft state and route, onboard avionics like the Automatic Dependent

Surveillance-Broadcast (ADS-B) traffic, and ground-based data sources such as weather forecasts and airspace restrictions. These optimized routes are presented to the aircraft pilot, who can request approval for the change from air traffic control, and on approval can follow the optimized path. Operational studies using TASAR showed a significant average cost and time savings [294, 295]. Stratway also adjusts the route of an ownship aircraft to avoid conflicts with other aircraft or weather. Stratway avoids other aircraft by receiving their planned routes or trajectories and moving waypoints in the vicinity of the conflict. The system assumes the original flight plan was nearly optimal, so slight deviations are the best resolution. While Stratway is not at the same technology readiness level as other examples given in this paper, it is worth noting because it has been rigorously developed with modularity, reusability, and formal verification of the conflict resolution in mind.

Spacecraft Strategic Collision Avoidance Systems

In the spacecraft domain, much of the spacecraft operations, including collision avoidance, remain human-planned and scripted [20]. One mitigation is that satellite owners and operators can select orbits that are less congested for lower conflict frequency, or can operate in well-regulated orbits like geostationary, where satellites stay in an assigned box of space (bounded orbital inclination, antenna axis attitude, and longitudinal drift) to avoid conflicts [45, 288]. One of the biggest challenges in collision avoidance for spacecraft lies in the tracking uncertainty used to detect possible collisions [222].

The *NASA Robotic Conjunction Assessment Risk Analysis (CARA)* and *ESA Collision Risk Assessment Software (CRASS)* are examples of strategic collision avoidance approaches in the spacecraft domain. These tools only predict collisions and do not recommend action for specific satellites. In response to a growing risk of collision to unmanned assets, NASA applied lessons from manned asset conjunction assessments to the development of the CARA tool in 2004 [296]. The CARA tool started by predicting collisions up 7 days out (later adjusted to less than 5.5 days) for low earth orbit (LEO) and less than 10 days out for geosynchronous equatorial orbits (GEO) or highly eccentric orbits (HEO). If a collision is predicted, CARA will include it in the Summary Report, which includes the current risk, risk history, probability of collision P_c, P_c history, current miss vector, miss distance history, relative geometry visualization, miss vector uncertainty, and secondary object

tracking information [297]. ESA's CRASS tool also detects high risk conjunctions in the next 7 days [21, 22, 23].

A.2.2 Tactical Collision Avoidance Systems

Tactical collision avoidance systems use medium term predictions and a single, simple (often less optimal) collision avoidance maneuver to provide significant margin by temporarily diverting from a planned flight path in near term encounters; like strategic, this definition is inspired by [289]. Since tactical collision avoidance systems do not propagate position as long as strategic, the uncertainty in a collision prediction and maneuver decision is lower. However, this uncertainty must still be considered in the context of mission criticality and risk tolerance when deciding to maneuver. While strategic collision avoidance systems operate with long durations and use modifications to flight plans, tactical collision avoidance systems typically focus on maneuvers that prevent violation of a large safety volume boundary around aircraft or spacecraft. A tactical maneuver may involve some automation or rely heavily on human planners like air traffic controllers or spacecraft operators to provide deconfliction instructions.

Aircraft Tactical Collision Avoidance Systems

In the aircraft domain, an example of this type of system is the *NASA Chorus* program [298], which iteratively searches through nine conflict resolution algorithms to determine which satisfies desired properties for a specific conflict scenario. If a solution is not found in an iterative approach, an analytical solution from the ACCoRD CRSS is returned, which has been formally verified using the PVS theorem prover [299]. This system uses a cylindrical safety volume with a 5 nautical mile diameter and 1000 ft above and below the aircraft, as depicted in Fig. A.3, to prevent violations of the FAA's "well-clear" status.

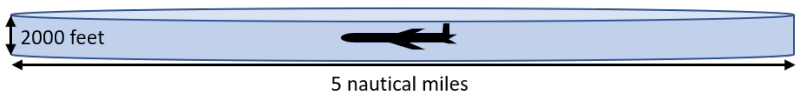


Figure A.2: Notional depiction (not to scale) of the "well-clear" 5 nautical mile diameter cylindrical volume and 1000 ft above and below the aircraft.

Spacecraft Tactical Collision Avoidance Systems

In the spacecraft domain, conjunction assessment has been partially automated, while the risk mitigation maneuver analysis is conducted by a human team [20]. For collision assessment, the United States Air Force (USAF), and more specifically the Air Force Space Command's (AFSPC) 14th Air Force, 18th Space Control Squadron (18SPCS) has the mission to maintain the authoritative space catalog and to conduct the routine space situational awareness. The 18SPCS provides Conjunction Data Messages (CDMs) to warn satellite owners and operators of potential collisions up to 72 hours in advance [18]. This warning is used here as the separation between Strategic and Tactical categories in the spacecraft domain. At this point, spacecraft owners/operators evaluate the CDM, compare the data to their own sources and decide a course of action. At NASA, the CARA tool [297] outputs a "RED" event when the probability of collision $P_c > 8.4 \times 10^{-4}$, "YELLOW" when the $P_c > 1 \times 10^{-7}$, and "GREEN" for lower probabilities. When maneuver planning is needed (for all "RED" events), CARA outputs a High Interest Event (HIE) report with additional analysis. At ESA, CRASS [21, 22, 23] sends high risk warnings when the probability of collision is greater than 10^{-4} or the closest miss distance is less than 300 meters. 18SPCS provides collision warnings when radial separation is less than 200 meters with an overall miss distance of 1 kilometer [23]. Since the warning sources have different thresholds, it is possible to receive a warning from one or multiple systems.

Collision detection information is used by a team to determine a course of action that tends to fit the single, simple collision avoidance maneuver definition of the tactical category. At NASA, the satellite owner/operator mission management and flight operations teams use data from CARA's HIE report to plan and execute risk mitigating actions [297]. At ESA, once CRASS or 18SPCS sends a conjunction assessment, the ESA space debris office (SDO) team confirms the risk contacts the on-call Spacecraft Controller (SC), Operations Engineer (SOE), Operations Manager (SOM), and Flight Dynamics (FD) support team by phone. The SOM and SOE perform a preliminary risk assessment to identify the date and time of the conjunction, the radial miss distance D_u, and a probability of collision (P_c). If $P_c > 10^{-4}$ or $D_u < 100m$, then a detailed Collision Avoidance Maneuver (CAM) assessment and maneuver scenarios are discussed by the Flight Control Team (FCT), FD team and SDO. Once the maneuver decision has been made and coordinated through-

out the organization, the FD team provides detailed maneuver planning including delta-V required, thruster on-time(s), telecommand files, position on orbit, and fuel consumption. To provide an example of how often this process occurs: in the first four years of the Cryosat-2 mission, 76 warnings were received, resulting in 6 avoidance maneuvers with a warning time that ranged from less than 24 hours to 6 days [20]. Both the NASA and ESA processes are highly dependent on subject matter expertise. Tactical collision avoidance approaches are the shortest time horizon currently used for spacecraft collision avoidance, in part because the human-centered planning is too time consuming for shorter term collision resolution maneuvers.

A.2.3 Detect and Avoid Collision Avoidance Systems

Detect and Avoid (DAA) or *Sense and Avoid (SAA)* systems rely on transponders or sensors (rather than ground based traffic controllers or operators) to detect near term collisions on-board and recommend actions to maintain minimum safe separation, generally with enough time for a human operator to engage the maneuver or stop the maneuver before it engages. The use of on-board transponders or sensing reduces uncertainty and confirms nearer term that a collision is imminent. There are not currently any mature detect and avoid spacecraft systems in the literature, so this section examines air domain systems and recommends elements that may be applied to the space domain.

Aircraft Detect and Avoid Collision Systems

In the case of manned aircraft, a DAA system provides a course of action for the human operator to follow but does not automatically engage the maneuver. In unmanned air systems, DAA may also automatically engage the collision avoidance maneuver. Example aircraft domain systems that fall in this category include the *Traffic Collision Advisory System (TCAS)* [209, 210], the *Traffic Alert and Collision Avoidance System II (TCAS II)* [211, 212], the *Aircraft Collision Avoidance System X (ACAS-X)* [213, 214, 215], *AFRL's Sense and Avoid (SAA)* system [210, 216, 217], and *NASA's Detect and Avoid Alerting Logic for Unmanned Systems (DAIDALUS)* program [218, 219]. For example, *TCAS* and the currently deployed version, *TCAS II*, are designed to operate independently of air traffic control and monitor aircraft fitted with transponders in the national air space to predict possible mid-air collisions. TCAS-equipped aircraft use a safety volume with dimensions that are

a function of ownship altitude as well as the range, range closure rate, altitude separation, and vertical closure rate with respect to the collision threat aircraft. If a possible collision is detected within a future time horizon, TCAS provides a traffic advisory. A notional depiction of the safety volumes used for TCAS II is shown in Fig. A.3. When that time horizon is sufficiently small, TCAS provides a resolution advisory to the pilots of both aircraft with recommended maneuvers (e.g. altitude change) to avoid the collision. The limitation of an advisory system is that it still requires human intervention; however, the human interaction with such a system may be to simply accept or reject the proposed maneuver. Including a human in-the-loop requires providing the advisory early enough that the pilot or operator can assess and respond to the situation.

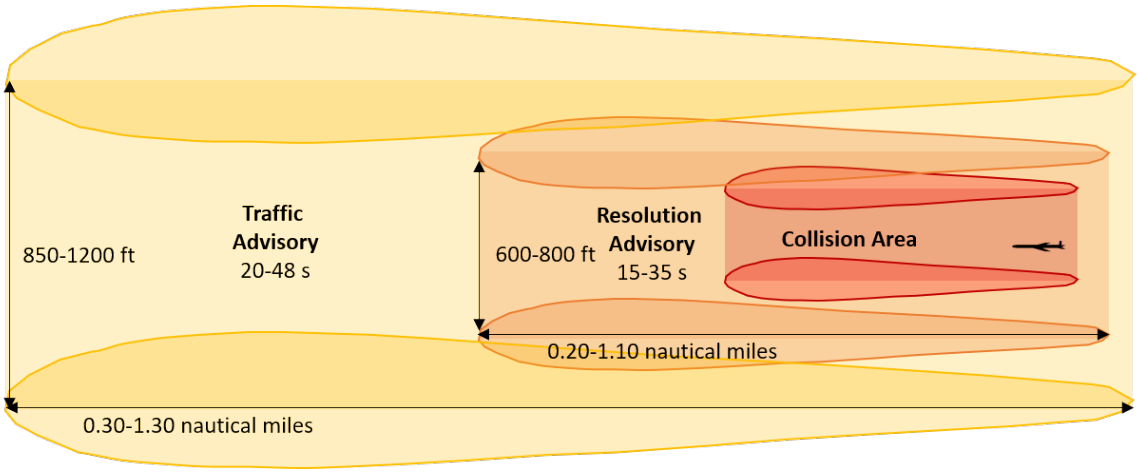


Figure A.3: Notional depiction (not to scale) of the safety volumes around an aircraft using TCAS II. Inspired by [211].

Recommended Elements for Spacecraft Detect and Avoid Collision Systems

While no operational or mature research DAA spacecraft systems are discussed in the literature, the aircraft domain offers some possible, though potentially challenging ways forward. Aircraft DAA systems may inspire the use of a transponder-like system with collision advisories for spacecraft. However, these transponders could present significant size, weight and power challenges. Sensing capabilities onboard spacecraft are confined to line-of-sight (which is limited, especially in LEO) and may require a significant portion of size, weight, and power resources to maintain an active sensing system specifically for collision avoidance. A publish and subscribe architecture like ROS [300] or UXAS [301], may be useful in traffic management [302] and may allow a collision

avoidance system to leverage sensor data from sensors that are not dedicated to collision avoidance. Another possible scenario may be the use of ground-based radar networks for sensing of near term, but not last instant collisions; however, this would likely fall into the tactical category due to the reliance on ground-based detection.

A.2.4 Last Instant Collision Avoidance Systems

Last instant collision avoidance systems detect collisions on-board and select from a limited set of pre-defined maneuvers to avoid the collision. One of the defining characteristics of this category is that maneuvers are engaged very close to the time of closest approach, often violating nominal safe separation requirements while still preventing the collision. In the case of automated avoidance maneuvers, the systems engage later than a human operator would respond. Minimal mission impact is exchanged for higher risk. Like the detect and avoid category, no last instant collision avoidance systems are described in the literature for the spacecraft domain. Two air domain last instant collision avoidance systems are examined and used to recommend approaches for the space domain, which are expanded upon in later sections. Detection in last instant collision avoidance systems relies on the use of GPS or Inertial Navigation Systems (INS) to compute ownship position and velocity and compare it to a digital terrain database on board (for ground collisions), or collision threat aircraft state data from messages sent between aircraft via datalink, transponders, or sensors like radar.

Automated Aircraft Collision Avoidance Systems

In the air domain, the *Automatic Ground Collision Avoidance (Auto GCAS)* and *Automatic Air Collision Avoidance (Auto ACAS)* are examples of last instant collision avoidance systems that automatically detect when a collision between the aircraft and the ground or another aircraft is imminent and then automatically maneuvers with seconds to spare. Both of these systems are designed with the guiding principles that the system should first *do no harm* (should not cause damage or harm to aircraft or pilot), then *do not interfere* (perhaps the most challenging principle that the system should allow the pilot to fly in the canyon, low to the ground, perform air combat maneuvers, and fly in formation, only turning on when absolutely necessary), and finally *prevent collisions*.

Auto GCAS [303, 191, 304, 192, 270, 186, 202] is designed to prevent ground collisions and has

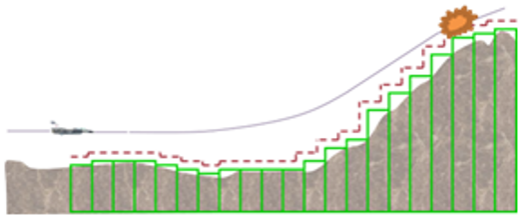


Figure A.4: Notional depiction (not to scale) of Auto GCAS collision prediction.

a single avoidance maneuver: roll the aircraft to wings level and conduct a five-g pull. To predict collisions, the aircraft uses a combination of GPS and INS to determine its location relative to a digital terrain database map. The avoidance maneuver is projected ahead of the current aircraft location and compared to a two-dimensional projection of the ground with buffers for uncertainty. When a predicted collision is imminent, the system maneuvers automatically. A critical metric in the nuisance-free (*do not interfere*) design criteria was the concept of time available. Zero time available was measured as the point at which activation of the maneuver would cause the aircraft to scrape the ground, and positive time available was measured as earlier in time from that point. To be nuisance free, early studies [191, 192] determined the system should aim to maneuver after a human pilot would have commanded a maneuver, often with less than a second of time available.

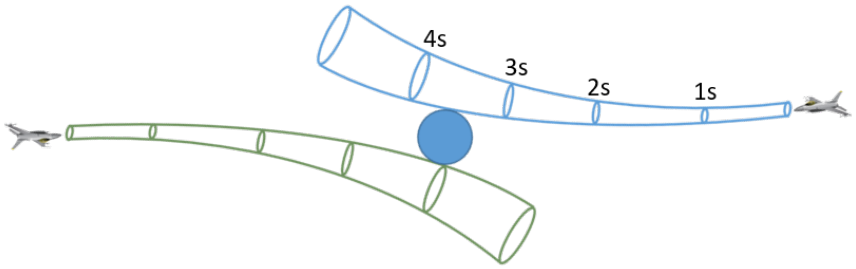


Figure A.5: Notional depiction (not to scale) of Auto ACAS collision prediction.

Auto ACAS [220, 204, 187] is designed to prevent midair collisions of aircraft and may select from one of nine possible avoidance maneuvers in a selection process that considers aircraft orientation, pilot rules of the road (e.g. two aircraft approaching head-on will both turn to the right), and pilot preferences (e.g. the selection should favor maneuvers that allow pilots to maintain line of sight with the other aircraft). Auto ACAS predicts and compares the trajectories of aircraft approximately four seconds into the future and automatically maneuvers to avoid collisions at the last instant. Cooperative aircraft are defined as those that also carry Auto ACAS and communicate their

position and intended avoidance maneuver over a data link. Non-cooperative aircraft may be observed by other means, such as radar, and automatically avoided. One of the key differences between cooperative and non-cooperative avoidance maneuvers is the cooperative maneuvers feature less uncertainty allowing aircraft to avoid later and get much closer. In non-cooperative maneuvers, the aircraft without Auto ACAS is assumed to maintain course and has a larger buffer of uncertainty, resulting in an earlier activation of an automatic maneuver and larger miss distance. The consequence of maneuvering earlier is that it may become a nuisance, interrupting the primary mission.

Spacecraft Last Instant Collision Avoidance

No operational or high-technology readiness level last instant collision avoidance systems for spacecraft are discussed in the literature. However, there are methods to predict collisions for short duration projections of spacecraft [222]. The problem of computing a last instant collision avoidance maneuver for the spacecraft domain is the focus of the remainder of this paper. Drawing inspiration from the Auto GCAS and Auto ACAS system, concepts such as the use of uncertainty buffers, metrics such as time available, reducing mission interruption, use of communication and collaboration between spacecraft, and the concept of selecting from a predefined set of maneuver options are explored in the next section.

APPENDIX B
BACKGROUND INFORMATION ON SPACECRAFT COLLISION DETECTION AND AVOIDANCE

The background information in this section provides context for the development of a last instant collision avoidance system in Section 5. First, motivation is provided for developing an automatic collision avoidance system by describing the current state of space traffic management and orbital debris as well as the anticipated growth of these challenges. Second, historic spacecraft collision provide context for the development of a collision avoidance system. Third, the collision prediction and avoidance approaches used in practice are described. Fourth, methods to predict collisions are explored, including general approaches for collision prediction, approaches to propagate uncertainty and predict collisions for spacecraft, and methods to compare collision prediction approaches. Fifth, collision avoidance maneuver approaches are described. Finally, related work in rendezvous and proximity operations for spacecraft is introduced.

B.1 Motivation

As of 2014, U.S. Space Surveillance Network (SSN) tracked a growing population of around 23,000 orbiting objects larger than 10 cm in orbit around Earth, although it is estimated there are hundreds of thousands of objects between 1 cm and 10 cm, and millions smaller than 1 cm [282]. In 1978 Kessler and Cour-Palais [283] published a paper predicting the cascading effect of artificial satellites colliding with increased frequency, creating a debris belt around the Earth. This concept became commonly known as "Kessler Syndrome" [284]. Due to the high velocities involved (an object in low-earth orbit moves about 7.8 km/s and relative velocities between objects can be larger), even collisions with small objects can cause catastrophic damage, and further magnify the problem by creating additional space debris. Although small amounts of atmospheric drag may eventually deorbit objects so they burn up in the Earth's atmosphere, this process can take dozens to hundreds of years or longer, depending on the orbit add the ballistic coefficient of the object.

In addition to the risk of collisions with existing orbiting spacecraft and debris, there is signif-

icant growing commercial interest in large constellations of satellites, with most of those objects concentrated in the already congested low Earth orbit (LEO) regime [305, 306]. Among the companies with approved proposals are SpaceX with proposed constellations of 4,425 and 7000 satellites[1], OneWeb with 720 satellites[2], Kepler Communication with 140, TeleSat with 117 and LeoSat with 117[3]. This increased congestion prompted the 2018 release of the White House released Space Policy Directive-3, National Space Traffic Management Policy [46], which discusses collision detection and avoidance extensively, stating: "Timely warning of potential collisions is essential to preserving the safety of space activities for all." Organizations like AIAA have begun describing the need for a comprehensive Space Traffic Management (STM) system, stating in a 2017 positions paper [48] that collision avoidance is the "top priority and metric for success of the STM Program." As the spacecraft environment becomes even more congested and contested [281], more agile approaches are required and the speed of subject-matter expert driven collision avoidance decisions may be insufficient to meet requirements.

Since 1999, the International Space Station (ISS) has conducted 25 debris collision avoidance maneuvers, and in 2017 alone NASA assisted in the execution of 21 collision avoidance maneuvers, one of which was to avoid the ISS [307], and the challenge of avoiding collisions is growing. Affordable access to space and spacecraft technology are driving an increase in the recent and planned launches of satellites, which poses an increased risk of satellite conjunctions. Many commercial entities are planning to launch large numbers or constellations of satellites in the next several years. Iridium Communications Inc. began deploying their 72-satellite constellation in 2017. SpaceX applied to launch 4,425 satellites between 2019-2025. OneWeb LLC is deploying 650 to 900 satellites starting in 2018. LeoSat plans to launch a constellation of 78 to 108 satellites. TeleSat and Viasat applied for launches of a 117-satellite and 24-satellite constellation, respectively [305, 306]. On November 15, 2018 the Federal Communication Commission approved additional satellite constellation proposals including a very low Earth orbit (VLEO) constellation of more than 7000 satellites for SpaceEx. At the time of this research, Space-Track.org[4] regularly publishes two-line element (TLE) sets for almost 17,000 unclassified objects. Proposals for new launches could nearly double

[1] https://docs.fcc.gov/public/attachments/DOC-349998A1.pdf
[2] https://docs.fcc.gov/public/attachments/DOC-345467A1.pdf
[3] https://docs.fcc.gov/public/attachments/DOC-355102A1.pdf
[4] https://www.space-track.org/

that number in the next eight years, with most of those objects concentrated in the already congested low Earth orbit (LEO) regime.

Today spacecraft collision avoidance maneuvers are planned by human subject matter experts and conducted days to hours in advance. As space becomes increasingly congested, new Space Traffic Management (STM) approaches are needed, and automated techniques for spacecraft maneuvering become increasingly attractive for tasks such as collision avoidance, rendezvous and proximity operations, and station keeping. As the number of objects continues to increase both from collisions and new launches, automating the decision and control to maneuver under a limited set of circumstances could keep the manpower required for continued operations within reason.

B.2 Historic Spacecraft Collisions

Historic examples of spacecraft collisions provide insight into the source of the collision and provide context for spacecraft collision prediction and avoidance system design. Three major historical events demonstrate the realistic risk and consequence of collisions. In January 2007, China tested an anti-satellite missile by destroying the Fengyn-1C Chinese weather satellite. The test resulted in the largest recorded creation of orbital debris, comprised of an estimated 150,000 particles and more than 2,000 trackable pieces 10 centimeters or larger [308]. In February 2009, the Iridium 33 satellite collided with the Russian military satellite Cosmos-2251 at a relative velocity of over 10 km/s 770 km above earth, creating cloud of debris including an estimated 1000 pieces larger than 10 centimeters [262]. India's 2019 anti-satellite test created 400 pieces of orbital debris, including 60 greater than 10 centimeters, and 24 of which are in eccentric orbits that pass through the International Space Station's orbital altitude, increasing risk of a small debris impact to the station by an estimated 44 percent [5].

Several examples of collision events may be found in the literature that motivate research in collision prediction and the development of automatic avoidance maneuvers. These include:

- *collisions during rendezvous*:
  - 1994: collision of the Soyuz TM-17 ferry spacecraft with the MIR space station, resulting in only minor damage [234];

[5] https://www.space.com/nasa-chief-condemns-india-anti-satellite-test.html

  - 1997: collision of the Progress M-34 spacecraft with mir, causing damage to MIR solar panels, radiators, and a hull puncture [235, 236];
  - 2005: collision of DART with MUBLCOM sending MUBLCOM into a higher orbit with no significant damages [237];

- *anti-satellite missile or anti-debris tests*
  - 1985: destruction of the P78-1 or "Solwind" satellite seven years into the mission after five onboard instruments have failed with the ASM-135 anti-sat missile launched from an F-15 [309]; all debris deorbited by 2008 [261];
  - 1986: collision of a satellite carrying an explosive with a Delta second stage at 192 km to ensure rapid debris reentry [264];
  - January 2007: anti-satellite missile destruction of the Fengyn-1C weather satellite generating 2,087 pieces of debris [310, 308] expected to stay in orbit up to 100 years;
  - Februrary 2008: destruction of the USA-193 satellite by the SM-3 Missile, with all debris deorbiting in 40 days [311];
  - March 2019: anti-satellite test by India creating 400 pieces of debris [6];

- *confirmed random, accidental collisions*
  - 23 December 1991: (but not recognized until 2005) collision of a defunct Cosmos navigation satellite with a piece of debris from another Cosmos satellite [259];
  - 24 July 1996: collision of the French Ceris Satellite with a fragment of Ariane-1 H-10 upper rocket stage [260];
  - 2001: an 800kg, 2-meter diameter cylindrical Russian satellite launched in 1998 was struck by Cosmos 926 debris [261];
  - 17 January 2005: collision of a U.S. rocket body with a fragment of the third stage of a Chinese launch vehicle [259, 261];
  - February 2009: Iridium 33 and Cosmos-225 Collision [262]; and

[6] https://www.space.com/nasa-chief-condemns-india-anti-satellite-test.html

– 22 January 2013: BLITS retroreflector satellite was impacted by a piece of orbital debris [263].

In addition to the collision events listed here, several suspected but unconfirmed orbital debris collisions have occurred, and are excluded here for brevity; however more details may be found in [264, 261].

B.3 Current State of Practical Spacecraft Collision Prediction and Avoidance

While many approaches are proposed in the literature for spacecraft prediction and avoidance, this research first gains context by examining the how collision prediction and avoidance is implemented in practice by the United States and European Space Agency.

In the United States, the United States Air Force (USAF) is the primary US government organization tasked with cataloging data on all the objects in Earth orbit, and the Combined Space Operations Center (CSpOC) currently provides Conjunction Data Messages (CDMs) to warn owners and operators of potential collisions [18]. The owners and operators of satellites are responsible for evaluating risk and potentially maneuvering their systems to avoid the collision. While some users of CDMs like NASA and ESA have established procedures and devote significant manpower to computing probabilities of collision, many newer commercial users have difficulty computing collision probabilities and having confidence in the data [18]. One of the biggest challenges for operators is the large number of conjunction predictions, which may increase as Earth orbit becomes increasingly congested. In 2014, the CSpOC predecessor issued 671,727 conjunction warnings (an average of over 1800 per day) [312], a number that may continue to increase without improvements in conjunction prediction frameworks.

Using the 1996 collision of the French Ceris satellite and a fragment of an Ariane-1 H-10 upper stage as a motivating example, work by [19] describes the collision prediction and risk estimation procedures used by the ESA to decide whether to conduct and avoidance maneuver. An updated description of the ESA conjunction assessment procedure is included in [20, 21, 22, 23], which was briefly discussed in Section B.3. The European Space Agency's (ESA) approach to collision avoidance is also described in a 2014 case study of the Cryosat-2 mission in [20] and summarized as follows:

- The process initiates in two ways: when ESA's Collision Risk Assessment Software (CRASS) detects a high risk conjunction within the next 7 days, or when the when a close conjunction warning is emailed to ESA's Space Debris Office (SDO) from USSTRATCOM/CSpOC 72 hours or less in advance of the conjunction. CRASS sends high risk warnings when the probability of collision is greater than 1/10000 ($P_c > 10^{-4}$) or the closest miss distance is less than 300 meters. CSpOC provides collision warnings when radial separation is less than 200 meters with an overall miss distance of 1 kilometer. Since the two warning sources have different thresholds, it is possible to receive a warning from one or both systems. CRASS uses USSTRATCOM TLEs and is considered less accurate than the information used by CSpOC.
- If the SDO team confirms the risk, the on-call Spacecraft Controller, Operations Engineer (SOE) and Operations Manager (SOM) are contacted by phone. The SOE immediately calls the on-call Flight Dynamics (FD) support team.
- The SOM and SOE perform a preliminary risk assessment to identify the date and time of the conjunction, the radial miss distance D_u, and a probability of collision (P_c).
- If $P_c > 10^{-4}$ or $D_u < 100m$, then a detailed Collision Avoidance Maneuver (CAM) assessment and maneuver scenarios are discussed by the Flight Control Team (FCT), FD team and SDO.
- Once the maneuver decision has been made and coordinated throughout the organization, the FD team provides detailed maneuver planning including delta-V required, thruster on-time(s), telecommand files, position on orbit, and fuel consumption.

The overall process is highly dependent on subject matter expertise. In the first four years of the Cryosat-2 mission, 76 warnings were received, resulting in 6 avoidance maneuvers. The warning time for cases that resulted in a maneuver ranged from less than 24 hours to 6 days.

Collision avoidance has long been discussed as a strategy for space debris mitigation [313]. Tradeoffs between retiring a satellite at the end of its planned mission or continuing use of the satellite at the risk of collision or unexpected loss of capabilities to complete the end-of-life procedure are discussed in [288]. The ESA's conjunction event detection, collision risk assessment, orbit determination, orbit and covariance propagation, and process control and data handling are presented before providing guidelines for collision avoidance maneuvers in [314]. ESA now uses a

combination of the Debris Risk Assessment and Mitigation Analysis (DRAMA) and Assessment of Risk Event Statistics (ARES) tools early in mission planning to determine the number of collision avoidance maneuvers expected each year based on a risk threshold for the mission. A description of operational collision avoidance at the ESA is presented in [287] and a statistical look at ESA conjunction predictions is presented in [23].

In summary, spacecraft collision prediction algorithms must operate significantly faster than real time (predicting 72 hours to 7 days in advance) to be able to predict collisions and warn satellite operators with sufficient time to make orbital adjustments. When collisions are predicted, operators must decide whether to maneuver their active assets. If the decision is made to maneuver, a team of experts on the ground designs and evaluates safe maneuver trajectories.

B.4 Approaches to Collision Prediction

This section expands upon the state of collision prediction and avoidance in practice to examine a variety of approaches to satellite collision prediction and avoidance. First, general approaches for collision prediction are explored. Next, approaches to propagate uncertainty and predict collisions for spacecraft specifically are described. Finally, and methods to compare collision prediction approaches are discussed.

B.4.1 General Collision Prediction Approaches

Before exploring collision prediction approaches specific to spacecraft, the general collision prediction problem is explored in this section. Conjunction prediction is a multi-object trajectory prediction problem to show that now two satellites occupy the same location in space at the same time within a finite time horizon and set of initial conditions. Extensive surveys are available of collision detection methods for graphics and physics applications [315]. Conjunction prediction falls in the category of space-time intersection methods, but rather than extruding volumes in 4D, initial work in [316] computes 4D bounding boxes of the space-time paths of objects. The observation that splitting trajectories by time improves accuracy has been used for collision detection based on swept volumes [317]. While adaptive time-step methods have been used [318], initial work in [316] uses per-object time steps. Other than AABB trees, additional data structures are available

for collision detection, such as oriented bounding-box (OBB) trees [319]. In this case, boxes can be arbitrarily rotated, and fast collision check is done using the separating axis theorem [320]. This can be advantageous since rotated boxes may have less over approximation than axis-aligned boxes, so that tree query operations will become more efficient. Since spacecraft dynamics generally stay within a plane, orienting bounding boxes relative to the plane maid reduce false positives. Interval arithmetic [321] has been used to reason between time steps and have also been used in combination with OBB trees to provide continuous-time collision detection [322]. Other data structures based on sphere hierarchies have also been considered [323]. In order to scale to higher dimensions (initial work only propagates one variable per satellite) other methods would be needed to compute occupancy intervals such as those based on reachability [324].

B.4.2 Uncertainty Propagation and Collision Predictions for Spacecraft

A variety of academic models have been used to predict spacecraft position over time, typically using simulation-based analysis [227, 325, 326, 327, 328]. A large source of position and velocity propagation uncertainty for near-Earth satellites is atmospheric modeling, which impacts the force of atmospheric drag on satellites, while the biggest source of uncertainty for objects far from Earth is solar radiation pressure [18]. When considering orbit propagation of objects, high area-to-mass ratio (HAMR) objects (e.g. rocket bodies) have special dynamics considerations [329]. These and other uncertainties have motivated the common use of Monte Carlo simulations for collision prediction. For long duration predictions, the P_c is most often synthetically created by many trials using one of two approaches [222]:

1. propagate objects forward in time and draw Monte Carlo samples from the probability distribution function (PDF) at the time of closest approach (TCA) and to calculate the P_c from the ratio of miss distances d_{ij} less than the combined radii of the two objects to the total number of samples (less computationally burdensome and accurate), as depicted in Fig. B.1; or

2. draw Monte Carlo samples from each PDF at observation time, propagate each forward to compute the miss distance d_{ij} between each of the objects at the time of closest approach (TCA), to calculate the P_c from the ratio of miss distances less than the combined radii of the two objects to the total number of samples (more computationally burdensome and accurate

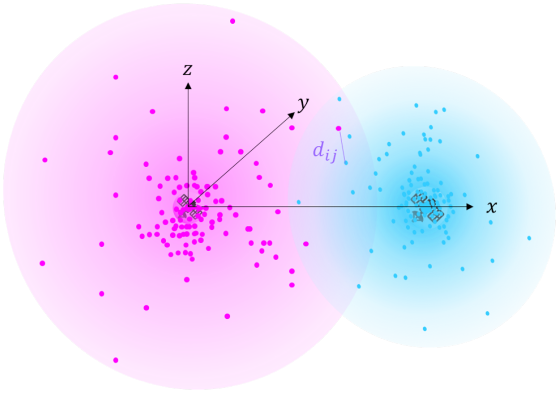


Figure B.1: Conceptual depiction of long duration collision prediction where Monte Carlo samples are selected from the probability density function at the time of closest approach after a single simulation to compute probability of collision.

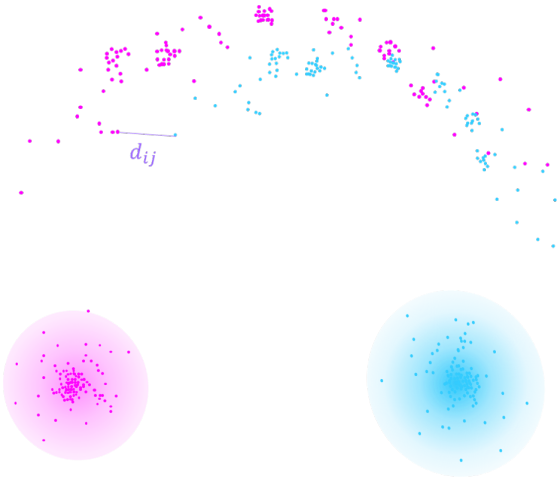


Figure B.2: Conceptual depiction of long duration satellite propagation where Monte Carlo samples are selected from the probability density function at the time of observation and propagated forward to the time of closest approach (often forming a non-convex distribution) to compute probability of collision.

approach), as depicted in Fig. B.2.

While many prediction techniques assume Gaussian probabilities, orbital mechanics can often feature non-Gaussian error in propagation and a variety of methods have been developed to cope with this [223, 222]. Figure B.3 compares 3 commonly used methods to propagate this type of uncertainty: Extended Kalman Filters (EKF), Unscented Kalman Filters (UKF), and Gauss von Mises Filters (GVM) versus Monte Carlo. In addition, Gaussian mixtures methods have been developed to model uncertainty at initial orbit determination [224, 225], during propagation [226], and for collision probability between multiple objects [227]. Öpik's formula has also been applied for collision risk assessment in LEO and MEO[330]. Some modeling techniques have taken a multi-fidelity approach [331] to speed up prediction computations. Methods have been developed to fuse information from two line element sets (TLEs) and radar measurements to better inform object tracking [332]. This speaks to the general problem of fusing data from multiple information sources, which is a significant challenge for space situational awareness. When trying to associate data from an observation to a known or new object, [333] suggests evaluating whether it is part of a centralized or decentralized system, interface and quality of different data sources, capacity to manage data sources and information, computational resources available, and acceptable level of robustness

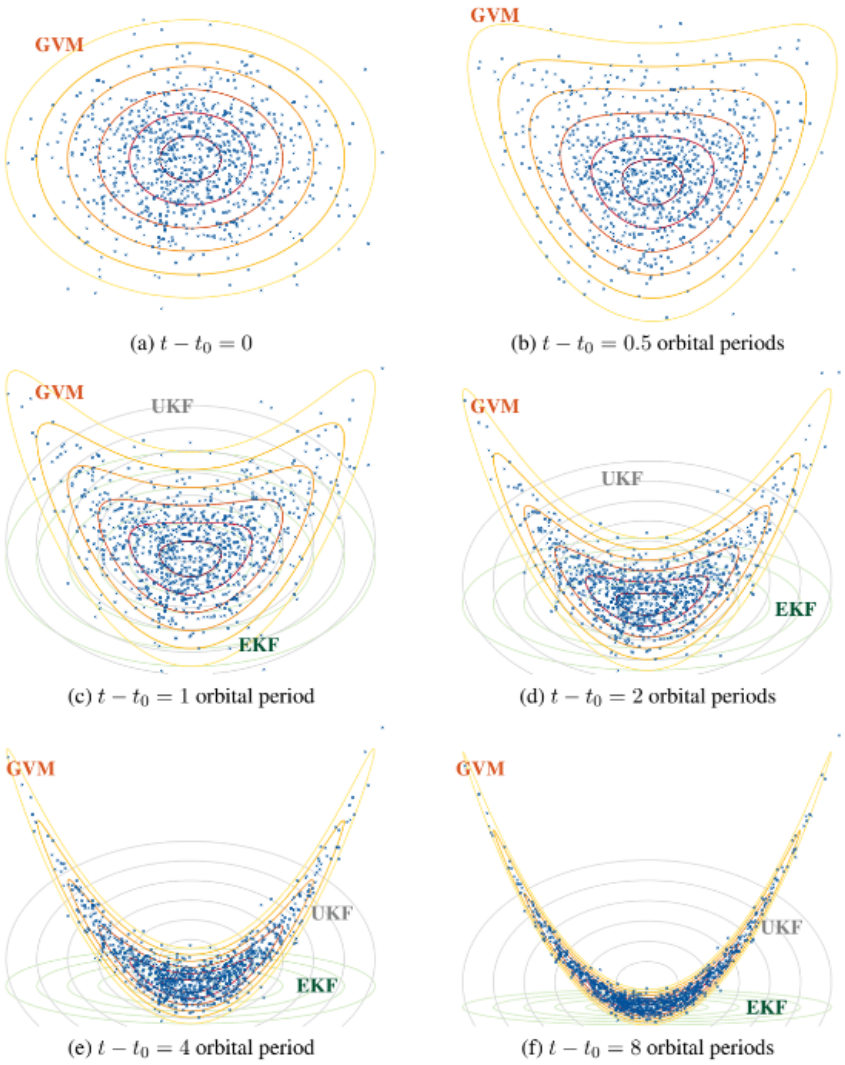


Figure B.3: Uncertainty Propagation Comparison of Extended Kalman Filter, Unscented Kalman Filter, and Gauss von Mises Filter with Monte Carlo [222]

to data association errors. When considering orbit propagation of objects, high area-to-mass ratio (HAMR) objects, like rocket bodies, have special dynamics considerations [334, 329].

A variety of methods have been developed to describe the distance or similarity between collisions which can be leveraged in collision prediction [335]. A body of work by Patera in [336, 337, 338, 339] describes various collision prediction and avoidance maneuver techniques. In [336], a computationally efficient method is presented to calculate collision probability using state vectors and error covariances for a general case, applicable to satellites of irregular shape, based on a 2D probability density function and combined hard-body of the colliding objects. In [337, 338],

this method is improved upon for nonlinear and linear relative motion by shifting the origin of the coordinate system and transforming the integral to one dimensional polar angle. In [339], Patera discusses the evolution of collision prediction from using states directly, to using position error probability density ellipsoids and determining the degree of overlap for two objects, to computing collision position using position with covariances and object size, to spherical conflict probability volume penetration. The conflict volume penetration collision prediction method is cited to be like the cylindrical conflict volumes used to predict aircraft midair collisions. The work in that paper extends the spherical conflict volume to compute conflict probability for an ellipsoidal conflict volume and analyses data from three actual collisions.

A variety of techniques have been developed for stochastic reachability and related probability (or chance) constrained optimal control problems. Level set methods have been to approximate solutions to stochastic necessary and sufficient Hamilton-Jacobi-Bellman equations for an optimum in continuous time [340, 341]. Dynamic programming has been used to approximate solutions in discrete time [342, 343]; however these approaches struggle to scale. For example, four dimensional problems are outside the scope for dynamical programming discrete time reachability problems. One way to address it was to assume worst-case bounded disturbances on linear dynamics [344].

Another related work is [345] which looks at two propagation methods and a variety of reachability techniques with bounded uncertainty to recreate a prediction of the 2009 Cosmos-Iridium collision. Figure B.4 shows an interval enclosure of an orbit, while Fig. B.5 shows how error accumulated from linearizing the dynamics is added to the conservative over approximation and can quickly grow. This highlights a big challenge: most abstractions used in verification break the physics of spacecraft orbits (conservation of energy and/or momentum), and yield reach sets that will eventually cover the entire state space if propagated for enough time.

B.4.3 Methods to Compare Collision Prediction Approaches

A 2011 comparison of five different satellite conjunction analysis tools was applied to detect all conjunctions between 11,807 objects in the public space catalog with a range of less than 9 km for a seven day period [346]. Two definitions are provided:

- *Point of closest approach (PCA) Conjunction*: local minimum of the range between objects

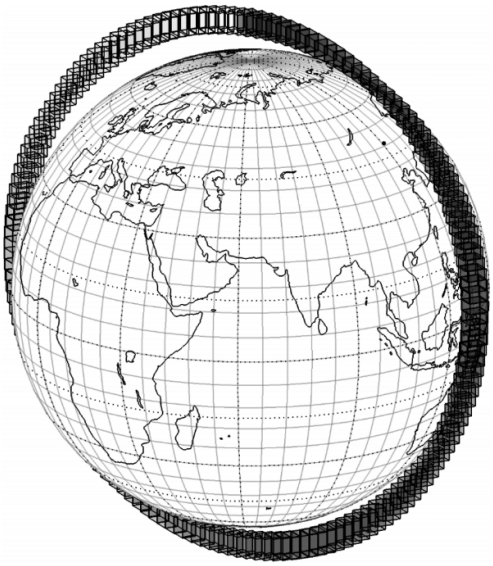


Figure B.4: Interval Orbit Enclosure [345]

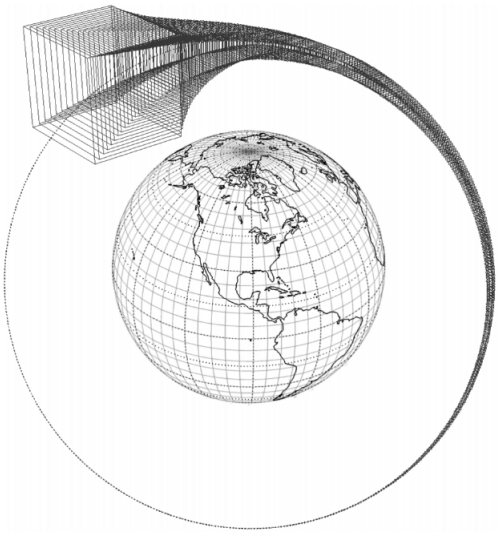


Figure B.5: Explosion due to over approximation error accumulation [345]

such that the range d is less than or equal to some screening threshold. The minimum corresponds to the root of the dot product of the position and velocity vectors.

$$\begin{aligned} &\text{Minimize } d, \vec{r} \cdot \vec{v} \\ &\text{such that } d \leq d_{th} \end{aligned} \tag{B.1}$$

- *Interval Conjunction*: the time interval during which the range between two objects is less than the screening threshold.

The five tools examined were Intelligent Software Solution and Data Fusion & Neural Networks's Continuous Anomalous Orbital Situation Discriminator (CAOS-D), Aerospace Corporation's Conjunction Sieve (CSieve), the USAF Computation of Miss Between Orbits (COMBO), Aerospace Corporation's Satellite Orbit Analysis Program (SOAP), Analytical Graphics' STK Advanced Conjunction Analysis Tools (CAT) and ShadowCAT. CAOS-D, CSieve, and SOAP found the same set of 116,746 "conjunctions" while STK Advanced CAT and ShadowCAT found slightly less. It was found that the STK Advanced CAT, which was not designed for all versus all conjunction prediction, had significant error when the Orbit Path filter was left in default, but all conjunctions were found when it was disabled. CAOS-D only checked PCA conjunctions and Advanced CAT only checked Interval conjunctions, while the other tools checked for both. While the tools treated satel-

lite ephemeris data differently, and used different propagators and techniques, the agreement in conjunction checking was remarkable.

B.5 Spacecraft Collision Avoidance Approaches

Once a collision is predicted by one of the many approaches discussed earlier, an avoidance maneuver must then be determined. The space domain lacks a governing body like air traffic control; however, individual satellite owners and operators can manually adjust operations plans. In addition, satellite owners and operators can select orbits that are less congested for lower conflict frequency, or can operate in well-regulated orbits like geostationary, where satellites stay in an assigned box of space to avoid conflicts. Much of the spacecraft operations, including collision avoidance, remain human-planned and scripted. The current position and velocity of a satellite may be updated by on-board GPS (if the satellite is equipped with one and enough power is available), or through ground observations via radar, lasers, or optical telescopes. If a probability of collision is detected up to a week in advance, modifications to the operations plan and script for the next week may be made. For example, an optimized maneuver may be included in the operation script for the week, or a maneuver that was planned might be canceled or delayed to avoid a collision. In some cases, although the threat is detected up to five days in advance, operators will prefer to take no action beyond monitoring to see if the collision threat is self-mitigated as the uncertainty shrinks with observation updates nearer to the time of closest approach (TCA).

Orbit data quality and propagation and collision detection techniques discussed earlier are an integral part of strategies for collision avoidance [347]. In [255], a method is introduced to maneuver space vehicles specifically for collision probability risk reduction. The method uses a simple two-body Kelper propagation with a gradient technique to find maneuver direction and a linear one-dimensional root finding scheme to find maneuver magnitude. In [326], Patera borrows the concept of conflict probability from aviation for spacecraft collision avoidance and collision risk reduction.

Several optimization techniques have also been proposed for collision avoidance strategies. The NLP2 algorithm [348] designs minimal delta-V maneuvers to reduce collision probability when the conjunction analysis predicts the probability of collision is above a threshold value. The algorithm can be configured to include an optional miss distance constraint and to include uncertainties in

individual or joint covariances. Bombardelli and Henando-Ayuso [251] developed an accurate analytical method to compute miss distance and collision probability of two objects after an impulsive collision avoidance maneuver. The maneuver is optimized as an eigenvalue problem in the collision "b-plane" that minimizes delta-V magnitude to maximize collision distance or minimize Gaussian collision probability. Results show that the linearized formula is valid by showing that the J2 perturbation negligibly affects the accuracy of the method. Maneuvers occur a few orbits prior to the predicted collision. Other optimal control techniques have examined using drag and solar radiation pressure for collision avoidance [349]. Investigation into collision avoidance strategies for geostationary satellites avoiding inclined geosynchronous satellites found that radial separation between the satellites was the most important factor in the strategy and was achieved when the avoidance maneuver was conducted twelve hours prior to the time of closest approach [350].

In addition to optimization of collision avoidance maneuvers, robust control techniques have been examined for collision avoidance. Model predictive control and an extended command governor were developed for robust relative motion guidance and control [351]. Techniques were considered for robust online optimization of collision avoidance maneuvers in [352]. Collision avoidance of obstacles using safe positively invariant sets was recently developed in [221]. Nonlinear orbital dynamics has also been considered in the case of coaxial and coplanar orbits [353] where hybridization [354, 355] was used to analyze the nonlinear dynamics and prove collision avoidance for the two-satellite case.

B.6 Related Research in Autonomous Spacecraft Rendezvous, Proximity Operations and Docking

The AFRL Space Vehicles Directorate published a benchmark paper [232] for academic exploration of autonomous spacecraft rendezvous, proximity operations, and docking (ARPOD) with an example of in-space assembly of a space station with a "chaser" spacecraft that transports the "target" passive components. The paper described the phases of the ARPOD mission, including the sensors and dynamics available at each phase. In addition, the following on-board sensors are listed: radar, GPS, Laser range finders ($<$ 1 km), stereo cameras ($<$100 m). Additional orbit and rendezvous benchmarks based on the Clohessy-Wiltshire Hill (CWH) equations have been proposed in [356,

357]. One example of autonomous servicing of satellites is autonomous fluid fuel and component transfer in the 2007 Orbital Express Mission by DARPA [358, 359, 360].

Stochastic reachability has been applied to characterize the initial set of states where spacecraft rendezvous and docking maneuvers could be performed safely [361]. This work focused on close proximity maneuvers based on linearized time-invariant dynamics Clohessy-Wiltshire-Hill equations for relative spacecraft motion. Stochastic noise was added separately to in-plane and out-of-plane motion. The work compared particle (Monte Carlo) approximations [362, 363] to a convex, overapproximated, reachable set [364, 365].

While not specifically designed for collision avoidance, recent developments in spacecraft relative motion control and planning [366, 367, 368, 369, 370, 351] could potentially be used for collision avoidance and are the focus of the application of the verified run time assurance methodology to spacecraft in this work.

B.7 Reference Frames and Coordinate Systems

There are dozens of ways to represent spacecraft dynamics depending on the properties desired for computation and a survey of 22 different representations are included in [371]. Just a subset is summarized here.

Spacecraft dynamics can be uniquely described using a set of at least six variables. The two most common approaches are position and velocity *cartesian coordinates* in an Earth-centered inertial reference frame (x, y, z, $\dot{x}$, $\dot{y}$, and $\dot{z}$) and *classic orbital elements* (semi-major axis a, eccentricity e, inclination i, argument of periapsis ω, right ascension of the ascending node Ω, and a variable representing position in the orbit which might be time of perifocal passage T, true anomaly ν, mean anomaly M, or time after periapsis passage t_p) [372, 373]. Beyond these traditional sets the following are also commonly used:

- The *equinoctial orbital elements* [374] uses the semi-major axis a and five other elements that are functions of the classical elements and feature the advantage of freedom from singularities in the classical orbital elements at $e = 0$ and $i = 0 or \pi/2$.
- The *modified equinoctial elements* [375, 376] uses the semi-latus rectum p and a slight variation of five other equinoctial orbital elements.

- The *unified state model* [377, 378] uses a total of seven state variables to describe the state of an orbiting objects. Three of these variables are functions of radial and angular momentum and describe the shape of the orbit in velocity phase space. The remaining four variables comprise a quaternion that orients the orbital plane with respect to an inertial reference frame.

While the techniques discussed above address longer duration orbital mechanics, short duration maneuvers and relative maneuvering are often described by the Clohessy-Wiltshire model in Hill's frame. Hill's frame is centered on the target spacecraft with the following coordinates: $\hat{i}$ radially outward from the center of the earth, $\hat{j}$ in the orbital velocity direction, and $\hat{k}$ pointing out of the orbital plane to complete orthogonal coordinate set. Several simplifying assumptions are often made including: a circular orbit, point mass satellites, and no rotational motion (3 degrees of freedom equations only). The dynamics are modeled with the Clohessy-Wiltshire Model:

$$\begin{aligned} \ddot{x} &= 2n\dot{y} + 3n^2 x + \frac{F_x}{m} \\ \ddot{y} &= -2n\dot{x} + \frac{F_y}{m} \\ \ddot{z} &= -n^2 z + \frac{F_z}{m} \end{aligned} \tag{B.2}$$

where x, y, and z are Cartesian positions; ρ is a 2-norm of position ($\rho = \sqrt{x^2 + y^2 + z^2}$); F_x, F_y, and F_z are thrust force applied by chaser spacecraft; μ is Earth's gravitational constant; a is length of the semi-major axis of the target's orbit; n is satellite mean motion ($n = \sqrt{\mu/a^3}$); and m is the mass of chaser spacecraft.

REFERENCES

[1] N. Minorsky, "Directional stability of automatically steered bodies," *Journal of the American Society for Naval Engineers*, vol. 34, no. 2, pp. 280–309, 1922.

[2] H. Nyquist, "Regeneration theory," *Bell System Technical Journal*, vol. 11, no. 1, pp. 126–147, 1932.

[3] H. W. Bode, "Relations between attenuation and phase in feedback amplifier design," *The Bell System Technical Journal*, vol. 19, no. 3, pp. 421–454, 1940.

[4] W. I. Caldwell, *Control system with automatic response adjustment*, US Patent 2,517,081, Aug. 1950.

[5] K. Åström, "History of adaptive control," *Encyclopedia of Systems and Control*, pp. 526–533, 2015.

[6] P. Parks, "Liapunov redesign of model reference adaptive control systems," *IEEE Transactions on Automatic Control*, vol. 11, no. 3, pp. 362–367, 1966.

[7] R. Avram, X. Zhang, J. A. Muse, and M. Clark, "Nonlinear adaptive control of quadrotor uavs with run-time safety assurance," in *AIAA Guidance, Navigation, and Control Conference*, 2017, p. 1896.

[8] "MIL-HDBK-516C: Airworthiness Certification Criteria," Department of Defense, Guidance, Dec. 2014.

[9] R. W. Butler and G. B. Finelli, "The infeasibility of quantifying the reliability of life-critical real-time software," *IEEE Transactions on Software Engineering*, vol. 19, no. 1, pp. 3–12, 1993.

[10] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[11] "Standard Practice for Methods to Safely Bound Flight Behavior of Unmanned Aircraft Systems Containing Complex Functions," ASTM International, Standard, 2017.

[12] N. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT press, 2011.

[13] C. Baier, J. Katoen, and K. Larsen, *Principles of Model Checking*. MIT Press, 2008.

[14] R. Alur, *Principles of Cyber-Physical Systems*. Cambridge, MA: The MIT Press, 2015.

[15] "Formal Methods Supplement to DO-178C and DO-278A," RTCA Inc., Guidance, 2011.

[16] N. T. Ho, G. G. Sadler, L. C. Hoffmann, J. B. Lyons, and W. W. Johnson, "Trust of a military automated system in an operational context," *Military Psychology*, vol. 29, no. 6, pp. 524–541, 2017.

[17] J. B. Lyons, N. T. Ho, W. E. Fergueson, G. G. Sadler, S. D. Cals, C. E. Richardson, and M. A. Wilkins, "Trust of an automatic ground collision avoidance technology: A fighter pilot perspective," *Military Psychology*, vol. 28, no. 4, pp. 271–277, 2016.

[18] National Research Council, *Continuing Kepler's Quest: Assessing Air Force Space Command's Astrodynamics Standards*. National Academies Press, 2012.

[19] H. Klinkrad, "On-orbit risk reduction - collision avoidance," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 221, no. 6, pp. 955–962, 2007.

[20] K. G. Symonds, T. Flohrer, N. Mardle, D. Fornarelli, X. Marc, and T. Ormston, "Operational reality of collision avoidance manoeuvres," in *SpaceOps 2014 Conference*, 2014, p. 1746.

[21] T. Flohrer, H. Krag, and H. Klinkrad, "Esa's process for the identification and assessment of high-risk conjunction events," *Advances in Space Research*, vol. 44, no. 3, pp. 355–363, 2009.

[22] T. Flohrer, H. Klinkrad, H. Krag, B. Bastida Virgili, and K. Merz, "Operational collision avoidance for leo satellites at esa," in *Proceedings of the 28th International Symposium on Space Technology and Science (ISTS), Okinawa, Japan*, 2011.

[23] T. Flohrer, H. Krag, S. Lemmens, B. Bastida Virgili, K. Merz, and H. Klinkrad, "Statistical look on esa's conjunction event predictions," in *Proceedings of the 6th European Conference on Space Debris, Darmstadt, Germany ESA SP-723*, 2013.

[24] K. Y. Rozier, "Specification: The biggest bottleneck in formal methods and autonomy," in *Working Conference on Verified Software: Theories, Tools, and Experiments*, Springer, 2016, pp. 8–26.

[25] C. Reis, A. Barth, and C. Pizano, "Browser security: Lessons from google chrome," *Commun. ACM.*, vol. 52, pp. 45–49, 2009.

[26] M. Bodson, J. Lehoczky, R. Rajkumar, L. Sha, and J. Stephan, "Analytic redundancy for software fault-tolerance in hard real-time systems," in *Foundations of Dependable Computing*, Springer, 1994, pp. 183–212.

[27] J. G. Rivera, A. A. Danylyszyn, C. B. Weinstock, L. R. Sha, and M. J. Gagliardi, "An architectural description of the simplex architecture.," Carnegie-Mellon University Software Engineering Institute, Pittsburgh, PA, Tech. Rep., 1996.

[28] L. Sha, R. Rajkumar, and M. Gagliardi, "Evolving dependable real-time systems," in *1996 IEEE Aerospace Applications Conference. Proceedings*, IEEE, vol. 1, 1998, pp. 335–346.

[29] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The simplex architecture for safe online control system upgrades," in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No. 98CH36207)*, IEEE, vol. 6, 1998, pp. 3504–3508.

[30] L. Sha, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[31] S. Bak, D. K. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha, "The system-level simplex architecture for improved real-time embedded system safety," in *2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, IEEE, 2009, pp. 99–107.

[32] S. Bak, K. Manamcheri, S. Mitra, and M. Caccamo, "Sandboxing controllers for cyber-physical systems," in *International Conference on Cyber-physical systems*, ser. ICCPS, 2011.

[33] S. Bak, A. Greer, and S. Mitra, "Hybrid cyberphysical system verification with simplex using discrete abstractions," in *2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, IEEE, 2010, pp. 143–152.

[34] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical computer science*, vol. 138, no. 1, pp. 3–34, 1995.

[35] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager, "The theory of timed i/o automata," *Synthesis Lectures on Distributed Computing Theory*, vol. 1, no. 1, pp. 1–137, 2010.

[36] S. Mitra, "A verification framework for hybrid systems," PhD thesis, Massachusetts Institute of Technology, 2007.

[37] M. Aiello, J. Berryman, J. Grohs, and J. Schierman, "Run-time assurance for advanced flight-critical control systems," in *AIAA Guidance, Navigation, and Control Conference*, 2010, p. 8041.

[38] P. Sorokowski, M. Skoog, S. Burrows, and S. Thomas, "Small UAV automatic ground collision avoidance system design considerations and flight test results," 2015.

[39] L. R. Hook, M. Skoog, M. Garland, W. Ryan, D. Sizoo, and J. VanHoudt, "Initial considerations of a multi-layered run time assurance approach to enable unpiloted aircraft," in *2018 IEEE Aerospace Conference*, IEEE, 2018, pp. 1–11.

[40] "Software Considerations in Airborne Systems and Equipment Certification," RTCA Inc., Guidance, 2011.

[41] J. A. Davis, M. Clark, D. Cofer, A. Fifarek, J. Hinchman, J. Hoffman, B. Hulbert, S. P. Miller, and L. Wagner, "Study on the barriers to the industrial adoption of formal methods," in *International Workshop on Formal Methods for Industrial Critical Systems*, Springer, 2013, pp. 63–77.

[42] U.S. Air Force Space & Missile Systems Center, *SMC Systems Engineering Primer & Handbook*, 2nd Edition, 2004.

[43] NASA, *NASA Systems Engineering Handbook*, NASA/SP-2007-6105 Rev1, 2007.

[44] E. Vassev and M. Hinchey, *Autonomy requirements engineering for space missions*. Springer, 2014.

[45] *Satellite Communications*, 47 CFR §25, 2010.

[46] *Space Policy Directive-3, National Space Traffic Management Policy*, Presidential Memoranda, 2018.

[47] M. Smith, *ESA Urges Automated Satellite Collision Avoidance Systems After Aeolus/Starlink Maneuver*, 2019.

[48] M. Jah, D. Greiman, M. Sengupta, S. Magnus, P. Melroy, S. Helms, and M. Brown, "Space Traffic Management (STM): Balancing Safety, Innovation, and Growth," American Institute of Aeronautics and Astronautics, Inc, Reston, VA, An Institute Position Paper, Nov. 2017.

[49] Defense Acquisition University, *Verification*, https://www.dau.mil/glossary/pages/2860.aspx, Accessed 14 November 2018, 2018.

[50] ——, *Validation*, https://www.dau.mil/glossary/pages/2852.aspx, Accessed 14 November 2018, 2018.

[51] J. William R. Nichols and T. Scanlon, "DoD Developer's Guidebook for Software Assurance," Carnegie Mellon University Software Engineering Institute, DoD Developer's Guidebook for Software Assurance, Dec. 2018.

[52] J. A. Estefan *et al.*, "Survey of model-based systems engineering (MBSE) methodologies," *Incose MBSE Focus Group*, vol. 25, no. 8, pp. 1–12, 2007.

[53] Federal Highway Administration and Federal Transit Administration, "Systems engineering for intelligent transportation systems: An introduction for transportation professionals," U.S. Department of Transportation, Tech. Rep., 2007.

[54] C. Haskins, K. Forsberg, and M. Krueger, "Systems engineering handbook: A guide for system life cycle processes and activities," International Council on Systems Engineering (INCOSE), Tech. Rep., 2007.

[55] W. W. Royce, "Managing the development of large software systems," *IEEE WESCON*, pp. 328–338, 1970.

[56] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, no. 5, pp. 61–72, 1988.

[57] J. G. Hall, M. Jackson, R. C. Laney, B. Nuseibeh, and L. Rapanotti, "Relating software requirements and architectures using problem frames," in *Proceedings IEEE Joint International Conference on Requirements Engineering*, IEEE, 2002, pp. 137–144.

[58] B. W. Boehm, *Software Engineering Economics*. Prentice-hall Englewood Cliffs (NJ), 1981.

[59] NIST, "The economic impacts of inadequate infrastructure for software testing," Tech. Rep. Planning Report 02-3, 2002.

[60] P. H. Feiler, *Supporting the ARP4761 safety assessment process with AADL*, Software Engineering Institute, Carnegie Mellon University, 2014.

[61] D. Galin, *Software Quality Assurance: from Theory to Implementation*. Pearson Education India, 2004.

[62] M. Clark, K. Kearns, J. Overholt, K. Gross, B. Barthelemy, and C. Reed, *Air Force Research Laboratory Test and Evaluation, Verification and Validation of Autonomous Systems Challenge Exploration*, Air Force Research Laboratory, Wright-Patterson AFB, OH, 2014.

[63] K. L. Hobbs, "Verification and validation of complex and autonomous systems," in *Proceedings of the FAA's 11th Annual Verification and Validation Summit*, ser. FAA V&V Summit 16, Atlantic City, NJ, 2016.

[64] ——, "Verification and validation of complex and autonomous systems," in *Proceedings of the 2018 IFAC Networked, Autonomous Air & Space Systems*, ser. NAASS18, Santa Fe, NM, 2018.

[65] ——, "Verification and validation in the age of autonomy," in *AIAA SciTech Forum 360 Panel*, ser. SciTech19, San Diego, CA, 2019.

[66] ——, "Trust and certification for autonomy," in *Proceedings of the Cognizant Autonomous Systems for Safety Critical Applications Conference*, ser. CASSCAC19, Miami, FL, 2019.

[67] N. G. Leveson, "Intent specifications: An approach to building human-centered specifications," *IEEE Transactions on software engineering*, vol. 26, no. 1, pp. 15–35, 2000.

[68] E. Clarke, S. German, Y. Lu, H. Veith, and D. Wang, "Executable protocol specification in esl," in *International Conference on Formal Methods in Computer-Aided Design*, Springer, 2000, pp. 217–236.

[69] "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment," SAE International, Guidance, 1996.

[70] "Guidelines For Development Of Civil Aircraft and Systems," SAE International, Guidance, 2010.

[71] "Design Assurance Guidance for Airborne Electronic Hardware," RTCA Inc., Guidance, 2000.

[72] "MIL-STD-882E: System Safety," Department of Defense, Standard Practice, May 2012.

[73] Federal Aviation Administration (FAA), *Standard airworthiness certification regulations: Title 14, code of federal regulations*, Website, `https://www.faa.gov/aircraft/air_cert/airworthiness_certification/std_awcert/std_awcert_regs/regs/`, Oct. 2017.

[74] M. Wooldridge, "Agents and software engineering," *AI* IA Notizie*, vol. 11, no. 3, pp. 31–37, 1998.

[75] C. A. R. Hoare, "An axiomatic basis for computer programming," *Communications of the ACM*, vol. 12, no. 10, pp. 576–580, 1969.

[76] P. Cousot and R. Cousot, "Abstract interpretation frameworks," *Journal of logic and computation*, vol. 2, no. 4, pp. 511–547, 1992.

[77] G. Brat, E. Denney, K. Farrell, D. Giannakopoulou, A. Jónsson, J. Frank, M. Boddy, T. Carpenter, T. Estlin, and M. Pivtoraiko, "A robust compositional architecture for autonomous systems," in *2006 IEEE Aerospace Conference*, IEEE, 2006, 8–pp.

[78] R. Alur, *Principles of Cyber-Physical Systems*. Cambridge, MA: The MIT Press, 2015.

[79] D. A. Peled, *Software Reliability Methods*. Springer Science & Business Media, 2013.

[80] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge university press, 2004.

[81] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Workshop on Logic of Programs*, Springer, 1981, pp. 52–71.

[82] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.

[83] Formal Systems Laboratory of the Department of Computer Science at the University of Illinois at Urbana-Champaign, *Past time linear temporal logic*, `http://fsl.cs.illinois.edu/index.php/Past_Time_Linear_Temporal_Logic`, Accessed 15 April 2019, 2008.

[84] N. Markey, "Temporal logic with past is exponentially more succinct," 2003.

[85] A. Cimatti, M. Roveri, and D. Sheridan, "Bounded verification of past ltl," in *International Conference on Formal Methods in Computer-Aided Design*, Springer, 2004, pp. 245–259.

[86] D. L. Parnas, "Tabular representation of relations," Communications Research Laboratory,McMaster University, Hamilton, Ontario Canada L8S 4K1, Tech. Rep. CRL Report No. 260, Oct. 1992.

[87] R. Janicki, D. L. Parnas, and J. Zucker, "Tabular representations in relational documents," in *Relational methods in computer science*, Springer, 1997, pp. 184–196.

[88] T. A. Alspaugh, S. R. Faulk, K. H. Britton, R. A. Parker, and D. L. Parnas, "Software requirements for the a-7e aircraft.," Naval Research Laboratory, Washington DC, Tech. Rep., 1992.

[89] C. Heitmeyer, J. Kirby, B. Labaw, and R. Bharadwaj, "Scr: A toolset for specifying and analyzing software requirements," in *International Conference on Computer Aided Verification*, Springer, 1998, pp. 526–531.

[90] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Property specification patterns for finite-state verification," in *Proceedings of the Second Workshop on Formal Methods in Software Practice*, ACM, 1998, pp. 7–15.

[91] ——, "Patterns in property specifications for finite-state verification," in *Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No. 99CB37002)*, IEEE, 1999, pp. 411–420.

[92] C. Menghi, C. Tsigkanos, T. Berger, and P. Pelliccione, "Psalm: Specification of dependable robotic missions," in *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings*, IEEE Press, 2019, pp. 99–102.

[93] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, and T. Berger, "Specification patterns for robotic missions," *arXiv preprint arXiv:1901.02077*, 2019.

[94] J.-P. Queille and J. Sifakis, "Specification and verification of concurrent systems in cesar," in *International Symposium on Programming*, Springer, 1982, pp. 337–351.

[95] S. B. Akers, "Binary decision diagrams," *IEEE Transactions on computers*, no. 6, pp. 509–516, 1978.

[96] L. De Moura, H. Rueß, and M. Sorea, "Bounded model checking and induction: From refutation to verification," in *International Conference on Computer Aided Verification*, Springer, 2003, pp. 14–26.

[97] N. Een, A. Mishchenko, and R. Brayton, "Efficient implementation of property directed reachability," in *Proceedings of the International Conference on Formal Methods in Computer-Aided Design*, FMCAD Inc, 2011, pp. 125–134.

[98] A. Gacek, A. Katis, M. W. Whalen, J. Backes, and D. Cofer, "Towards Realizability Checking of Contracts using Theories," *ArXiv e-prints*, Feb. 2015. arXiv: `1502.03005 [cs.SE]`.

[99] E. Ghassabani, A. Gacek, and M. W. Whalen, "Efficient Generation of Inductive Validity Cores for Safety Properties," *ArXiv e-prints*, Mar. 2016. arXiv: `1603.04276 [cs.SE]`.

[100] SysML.org, *SysML Open Source Project - What is SysML?* https://sysml.org/, Accessed: 27 July 2018, 2018.

[101] F. G. C. Ribeiro, C. E. Pereira, A. Rettberg, and M. S. Soares, "Model-based requirements specification of real-time systems with uml, sysml and marte," *Software & Systems Modeling*, vol. 17, no. 1, pp. 343–361, 2018.

[102] IBM, *ibm rational doors family*.

[103] QRA, *Leveraging natural language processing in requirements analysis: How to eliminate over half of all design errors before they occur*, Website, http://qracorp.com/wpcontent/uploads/2017/03/Leveraging-NLP-in-Requirements-Analysis.pdf, Mar. 2017.

[104] E. Vaz, "Delivering better projects on time by ensuring requirements quality upfront," in *INCOSE International Symposium*, Wiley Online Library, vol. 28, 2018, pp. 575–586.

[105] M. G. Oliveira, "Automated requirements analysis using natural language processing," Master's thesis, Faculdade de Engenharia do Universidade do Porto, 2018.

[106] A. Naeem, Z. Aslam, and M. A. Shah, "Analyzing quality of software requirements; a comparison study on nlp tools," in *2019 25th International Conference on Automation and Computing (ICAC)*, IEEE, 2019, pp. 1–6.

[107] V. Gervasi, A. Ferrari, D. Zowghi, and P. Spoletini, "Ambiguity in requirements engineering: Towards a unifying framework," in *From Software Engineering to Formal Methods and Tools, and Back*, Springer, 2019, pp. 191–210.

[108] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated checking of conformance to requirements templates using natural language processing," *IEEE transactions on Software Engineering*, vol. 41, no. 10, pp. 944–968, 2015.

[109] S. Nejati, M. Sabetzadeh, C. Arora, L. C. Briand, and F. Mandoux, "Automated change impact analysis between sysml models of requirements and design," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ACM, 2016, pp. 242–253.

[110] M. Rahim, A. Hammad, and M. Ioualalen, "A methodology for verifying sysml requirements using activity diagrams," *Innovations in Systems and Software Engineering*, vol. 13, no. 1, pp. 19–33, 2017.

[111] A. Cheng, S. Christensen, and K. H. Mortensen, "Model checking coloured petri nets-exploiting strongly connected components," *DAIMI report series*, no. 519, 1997.

[112] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured petri nets and cpn tools for modelling and validation of concurrent systems," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3-4, pp. 213–254, 2007.

[113] K. Wiegers, *The bottom line: Better requirements add business value*, Jama Software Whitepaper, https://resources.jamasoftware.com/paper/the-bottom-line-better-requirements-add-business-value, Sep. 2012.

[114] V. Navarro, A. Sisask, K. Hanson, R. Gill, J. Marcos, M. Fernandez, J. C. Segovia, and C. Arviset, "Towards a common software engineering environment for science operations," in *14th International Conference on Space Operations*, 2016, p. 2328.

[115] S Ghosh, N Shankar, P Lincoln, D Elenius, W Li, and W Steiener, *Automatic Requirements Specification Extraction from Natural Language (ARSENAL)*, Final Report, SRI International, Menlo Park, CA, Oct. 2014.

[116] S. Ghosh, D. Elenius, W. Li, P. Lincoln, N. Shankar, and W. Steiner, "ARSENAL: Automatic requirements specification extraction from natural language," in *NASA Formal Methods Symposium*, Springer, 2016, pp. 41–46.

[117] G. Brat, M. D. Davies, J. H. Koelling, J. M. Maddalon, and P. Miner, "Moving the validation and verification frontier: The system-wide safety march towards scalability and autonomy," in *AIAA Aviation 2019 Forum*, 2019, p. 2834.

[118] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic web*, vol. 8, no. 3, pp. 489–508, 2017.

[119] G. Brat, "Reducing v&v cost of flight critical systems: Myth or reality," in *AIAA Information Systems-AIAA Infotech@ Aerospace*, 2017, p. 0563.

[120] E. Vaz, C. M. Elliott, and P. O. Stanfill, "A viable approach to formal verification of complex aerospace systems," in *AIAA Scitech 2019 Forum*, 2019, p. 0504.

[121] C. Elliott, G. Tallant, and P. Stanfill, "On example models and challenges ahead for the evaluation of complex cyber-physical systems with state of the art formal methods v&v," in *Air Force Research Laboratory Safe and Secure Systems and Software Symposium (S5) Conference, Dayton, OH*, 2015.

[122] I. The MathWorks, *Simulink design verifier: Identify design errors, prove requirements compliance, and generate tests*, Website, https://www.mathworks.com/products/simulink-design-verifier.html.

[123] S. Nejati, K. Gaaloul, C. Menghi, L. C. Briand, S. Foster, and D. Wolfe, "Evaluating model testing and model checking for finding requirements violations in simulink models," *arXiv preprint arXiv:1905.03490*, 2019.

[124] L. Lúcio, S. Rahman, C.-H. Cheng, and A. Mavin, "Just formal enough? automated analysis of ears requirements," in *NASA Formal Methods Symposium*, Springer, 2017, pp. 427–434.

[125] L. Lúcio, S. Rahman, S. bin Abid, and A. Mavin, "Ears-ctrl: Generating controllers for dummies.," in *MODELS (Satellite Events)*, 2017, pp. 566–570.

[126] A. W. Fifarek, L. G. Wagner, J. A. Hoffman, B. D. Rodes, M. A. Aiello, and J. A. Davis, "Spear v2.0: Formalized past ltl specification and analysis of requirements," in *NASA Formal Methods*, C. Barrett, M. Davies, and T. Kahsai, Eds., Cham: Springer International Publishing, 2017, pp. 420–426, ISBN: 978-3-319-57288-8.

[127] A. Moitra, K. Siu, A. Crapo, H. Chamarthi, M. Durling, M. Li, H. Yu, P. Manolios, and M. Meiners, "Towards development of complete and conflict-free requirements," in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, IEEE, 2018, pp. 286–296.

[128] A. Crapo, A. Moitra, C. McMillan, and D. Russell, "Requirements capture and analysis in assert (tm)," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, IEEE, 2017, pp. 283–291.

[129] K. Siu, A. Moitra, M. Durling, A. Crapo, M. Li, H. Yu, H. Herencia-Zapana, M. Castillo-Effen, S. Sen, C. McMillan, *et al.*, "Flight critical software and systems development using assert™," in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, IEEE, 2017, pp. 1–10.

[130] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, "The synchronous data flow programming language lustre," in *Proceedings of the IEEE*, 1991, pp. 1305–1320.

[131] A. Gacek, J. Backes, M. Whalen, L. Wagner, and E. Ghassabani, "The jkind model checker," in *International Conference on Computer Aided Verification*, Springer, 2018, pp. 20–27.

[132] L. De Moura and N. Bjørner, "Satisfiability modulo theories: Introduction and applications," *Communications of the ACM*, vol. 54, no. 9, pp. 69–77, 2011.

[133] ——, "Z3: An efficient smt solver," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2008, pp. 337–340.

[134] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli, "Cvc4," in *International Conference on Computer Aided Verification*, Springer, 2011, pp. 171–177.

[135] J. Christ, J. Hoenicke, and A. Nutz, "Smtinterpol: An interpolating smt solver," in *International SPIN Workshop on Model Checking of Software*, Springer, 2012, pp. 248–254.

[136] B. Dutertre, "Yices 2.2," in *International Conference on Computer Aided Verification*, Springer, 2014, pp. 737–744.

[137] W3C, *Owl web ontology language reference: W3C recommendation*, Website, `http://www.w3.org/TR/owl-ref`.

[138] A. Crapo and A. Moitra, "Toward a unified english-like representation of semantic models, data, and graph patterns for subject matter experts," *International Journal of Semantic Computing*, vol. 7, no. 03, pp. 215–236, 2013.

[139] SWI-Prolog, *Swi-prolog for the (semantic) web*, Website, https://www.swi-prolog.org/web/.

[140] P. H. Winston and B. K. Horn, "Lisp," 1986.

[141] P. Manolios, *Scalable methods for analyzing formalized requirements and localizing errors*, US Patent 9,639,450, May 2017.

[142] H. R. Chamarthi and P. Manolios, *The ACL2 Sedan Theorem Prover*, Website, http://acl2s.ccs.neu.edu/acl2s/doc/#.

[143] E. Bartocci and Y. Falcone, *Lectures on Runtime Verification: Introductory and Advanced Topics*. Springer, 2018, vol. 10457.

[144] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[145] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, IEEE, 2014, pp. 6271–6278.

[146] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 509–521, 1998.

[147] E. Bartocci, Y. Falcone, A. Francalanza, and G. Reger, "Introduction to runtime verification," in *Lectures on Runtime Verification*, Springer, 2018, pp. 1–33.

[148] G. Reger and K. Havelund, "What is a trace? a runtime verification perspective," in *International Symposium on Leveraging Applications of Formal Methods*, Springer, 2016, pp. 339–355.

[149] J. Rushby, "Software verification and system assurance," in *2009 Seventh IEEE International Conference on Software Engineering and Formal Methods*, IEEE, 2009, pp. 3–10.

[150] A. E. Goodloe and L. Pike, "Monitoring distributed real-time systems: A survey and future directions," 2010.

[151] L. Pike, N. Wegmann, S. Niller, and A. Goodloe, "Copilot: Monitoring embedded systems," *Innovations in Systems and Software Engineering*, vol. 9, no. 4, pp. 235–255, 2013.

[152] J. Laurent, A. Goodloe, and L. Pike, "Assuring the guardians," in *Runtime Verification*, Springer, 2015, pp. 87–101.

[153] A. Kane, *Runtime monitoring for safety-critical embedded systems*, Doctoral Dissertation, Carnegie Mellon University, Pittsburg, PA, 2015.

[154] L. Sha, R. Rajkumar, and M. Gagliardl, "A software architecture for dependable and evolvable industrial computing systems.," Carnegie-Mellon University Software Engineering Institute, Pittsburgh, PA, Tech. Rep., 1995.

[155] J. Schierman, D. Ward, B. Dutoi, A. Aiello, J. Berryman, M. DeVore, W. Storm, and J. Wadley, "Run-time verification and validation for safety-critical flight control systems," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, p. 6338.

[156] J. D. Schierman, M. D. DeVore, N. D. Richards, N. Gandhi, J. K. Cooper, K. R. Horneman, S. Stoller, and S. Smolka, "Runtime assurance framework development for highly adaptive flight control systems," Barron Associates, Inc. Charlottesville, Tech. Rep., 2015.

[157] E. Wong, J. D. Schierman, T. Schlapkohl, and A. Chicatelli, "Towards run-time assurance of advanced propulsion algorithms," in *50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, 2014, p. 3636.

[158] J. D. Schierman, D. Neal, E. Wong, and A. K. Chicatelli, "Runtime assurance protection for advanced turbofan engine control," in *2018 AIAA Guidance, Navigation, and Control Conference*, 2018, p. 1112.

[159] A. J. Bateman, W. Gressick, and N. Gandhi, "Application of run-time assurance architecture to robust geofencing of suas," in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 1985.

[160] L. R. Hook, M. Clark, D. Sizoo, M. A. Skoog, and J. Brady, "Certification strategies using run-time safety assurance for part 23 autopilot systems," in *2016 IEEE Aerospace Conference*, IEEE, 2016, pp. 1–10.

[161] S. E. Jones, A. K. Petry, C. A. Eger, R. M. Turner, E. M. Griffin, *et al.*, "Automatic integrated collision avoidance system," in *17th Australian International Aerospace Congress: AIAC 2017*, Engineers Australia, Royal Aeronautical Society, 2017, p. 13.

[162] R. M. Turner, A. J. Albert, E. M. Griffin, A. C. Burns, F. Barfield, K. L. Price, *et al.*, "Automatic collision avoidance technology," in *AIAC18: 18th Australian International Aerospace Congress (2019): HUMS-11th Defence Science and Technology (DST) International Conference on Health and Usage Monitoring (HUMS 2019): ISSFD-27th International Symposium on Space Flight Dynamics (ISSFD)*, Engineers Australia, Royal Aeronautical Society., 2019, p. 495.

[163] C. Torens and F.-M. Adolf, "Formal requirements and model-checking for V&V automation of a RPAS mission management system," in *AIAA Infotech @ Aerospace*, Kissimmee, FL, Jan. 2015, pp. 1–13.

[164] I. Lee, H. Ben-Abdallah, S. Kannan, M. Kim, O. Sokolsky, and M. Viswanathan, "A monitoring and checking framework for run-time correctness assurance," 1998.

[165] I. Lee, S. Kannan, M. Kim, O. Sokolsky, and M. Viswanathan, "Runtime assurance based on formal specifications," *Departmental Papers (CIS)*, p. 294, 1999.

[166] K. Gross, J. Hoffman, M. Clark, E. Swenson, R. Cobb, M. Whalen, and L. Wagner, "Evaluation of formal methods tools applied to a 6u cubesat attitude control system," in *AIAA SPACE 2015 Conference and Exposition*, 2015, p. 4529.

[167] K. H. Gross, M. A. Clark, J. A. Hoffman, E. D. Swenson, and A. W. Fifarek, "Run-time assurance and formal methods analysis nonlinear system applied to nonlinear system control," *Journal of Aerospace Information Systems*, vol. 14, no. 4, pp. 232–246, 2017.

[168] K. H. Gross, M. Clark, J. A. Hoffman, A. Fifarek, K. Rattan, E. Swenson, M. Whalen, and L. Wagner, "Formally verified run time assurance architecture of a 6U cubesat attitude control system," in *AIAA Infotech@ Aerospace*, 2016, p. 0222.

[169] K. H. Gross, "Evaluation of verification approaches applied to nonlinear system control," Master's thesis, Air Force Institute of Technology, Mar. 2016.

[170] K. L. Hobbs, I. Perez, A. Fifarek, and E. M. Feron, "Formal verification of system states for spacecraft automatic maneuvering," in *AIAA Scitech 2019 Forum*, 2019, p. 1187.

[171] M. Abate, E. Feron, and S. Coogan, "Monitor-based runtime assurance for temporal logic specifications," *arXiv preprint arXiv:1908.03284*, 2019.

[172] N. G. Leveson, "Role of software in spacecraft accidents," *Journal of spacecraft and Rockets*, vol. 41, no. 4, pp. 564–575, 2004.

[173] ——, "A systems-theoretic approach to safety in software-intensive systems," *IEEE Transactions on Dependable and Secure computing*, vol. 1, no. 1, pp. 66–86, 2004.

[174] ——, "Software challenges in achieving space safety," 2009.

[175] T. Ishimatsu, N. G. Leveson, J. P. Thomas, C. H. Fleming, M. Katahira, Y. Miyamoto, R. Ujiie, H. Nakao, and N. Hoshino, "Hazard analysis of complex spacecraft using systems-theoretic process analysis," *Journal of Spacecraft and Rockets*, vol. 51, no. 2, pp. 509–522, 2014.

[176] C. K. Allison, K. M. Revell, R. Sears, and N. A. Stanton, "Systems theoretic accident model and process (stamp) safety modelling applied to an aircraft rapid decompression event," *Safety Science*, vol. 98, pp. 159–166, 2017.

[177] J. M. Rising and N. G. Leveson, "Systems-theoretic process analysis of space launch vehicles," *Journal of Space Safety Engineering*, vol. 5, no. 3-4, pp. 153–183, 2018.

[178] K. Johnson and N. G. Leveson, "Investigating safety and cybersecurity design tradespace for manned-unmanned aerial systems integration using systems theoretic process analysis.," in *GI-Jahrestagung*, 2014, pp. 643–647.

[179] C. H. Fleming, M. Spencer, J. Thomas, N. Leveson, and C. Wilkinson, "Safety assurance in nextgen and complex transportation systems," *Safety Science*, vol. 55, pp. 173–187, 2013.

[180] E. Harkleroad, A. Vela, and J. Kuchar, "Review of systems-theoretic process analysis (stpa) method and results to support nextgen concept assessment and validation," *Lexington, MA: Massachusetts Institute of Technology*, 2013.

[181] K. L. Hobbs, C. Cargal, E. Feron, and R. S. Burns, "Early safety analysis of manned-unmanned team system," in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 1984.

[182] J. J. R. Robertson, "Systems theoretic process analysis applied to manned-unmanned teaming," PhD thesis, Massachusetts Institute of Technology, 2019.

[183] J. Chen, S. Zhang, Y. Lu, and P. Tang, "Stpa-based hazard analysis of a complex uav system in take-off," in *2015 International Conference on Transportation Information and Safety (ICTIS)*, IEEE, 2015, pp. 774–779.

[184] K. A. Weiss, N. Dulac, S. Chiesi, M. Daouk, D. Zipkin, and N. Leveson, "Engineering spacecraft mission software using a model-based and safety-driven design methodology," *Journal of Aerospace Computing, Information, and Communication*, vol. 3, no. 11, pp. 562–586, 2006.

[185] J. D. Lee and K. A. See, "Trust in automation: Designing for appropriate reliance," *Human Factors*, vol. 46, no. 1, pp. 50–80, 2004.

[186] E. M. Griffin, R. M. Turner, S. C. Whitcomb, D. E. Swihart, J. M. Bier, K. L. Hobbs, and A. C. Burns, "Automatic ground collision avoidance system design for pre-block 40 f-16 configurations," in *Asia-Pacific International Symposium on Aerospace Technology*, 2012.

[187] R. Turner, R. Lehmann, J. Wadley, D. Kidd, D. Swihart, J. Bier, and K. Hobbs, "Automatic aircraft collision avoidance algorithm design for fighter aircraft," in *Asia-Pacific International Symposium on Aerospace Technology*, 2012.

[188] M. A. Skoog, K. Prosser, and L. Hook, *Ground collision avoidance system (iGCAS)*, US Patent 9,633,567, Apr. 2017.

[189] J. D. Carpenter, "Simulation and piloted simulator study of an automatic ground collision avoidance system for performance limited aircraft," Air Force Institute of Technology, Wright-Patterson AFB, OH, Tech. Rep., 2019.

[190] A. W. Suplisson, "Optimal Recovery Trajectories for Automatic Ground Collision Avoidance Systems (Auto GCAS)," Air Force Institute of Technology, Tech. Rep., 2015.

[191] D. E. Swihart, A. F. Barfield, E. M. Griffin, R. C. Lehmann, S. C. Whitcomb, B. Flynn, M. A. Skoog, and K. E. Prosser, "Automatic ground collision avoidance system design, integration, & flight test," *IEEE Aerospace and Electronic Systems Magazine*, vol. 26, no. 5, pp. 4–11, 2011.

[192] D. Swihart, A. Barfield, E. Griffin, R. Lehmann, S. Whitcomb, M. Skoog, B Flynn, and K. Prosser, "Design, integration and flight test of an autonomous ground collision avoidance system," *Gyroscopy and Navigation*, vol. 2, no. 2, pp. 84–91, 2011.

[193] R. D. Sorkin, "Why are people turning off our alarms?" *The Journal of the Acoustical Society of America*, vol. 84, no. 3, pp. 1107–1108, 1988.

[194] R. Lehmann and K. Trent, "F-16 ground proximity warning system study," General Dynamics, Fort Worth Division, Tech. Rep., 1987.

[195] Lockheed Martin, "F-16 M2 Modular Mission Computer Avionics Systems Manual," Lockheed Martin, Fort Worth Division, Tech. Rep., 1997.

[196] H. I. Inc, "Mk vi and mk viii enhanced ground proximity warning system (egpws) pilot's guide," 2004.

[197] ——, "MK V and MK VII Enhanced Ground Proximity Warning System (EGPWS) Pilot's Guide," 2011.

[198] Federal Aviation Administration, "Safer Skies General Aviation (GA) Controlled Flight into Terrain (CFIT) Joint Safety Implementation Team (JSIT) Final Report," Tech. Rep., 2000.

[199] M. L. Moroze and M. P Snow., "Causes and remedies of controlled flight into terrain in military and civil aviation," Air Force Research Laboratory, Tech. Rep., 1999.

[200] K. Geels-Blair, S. Rice, and J. Schwark, "Using system-wide trust theory to reveal the contagion effects of automation false alarms and misses on compliance and reliance in a simulated aviation task," *The International Journal of Aviation Psychology*, vol. 23, no. 3, pp. 245–266, 2013.

[201] A. Barfield, J. Probert, and D. Browning, "All terrain ground collision avoidance and maneuvering terrain following for automated low level night attack," *IEEE Aerospace and Electronic Systems Magazine*, vol. 8, no. 3, pp. 40–47, 1993.

[202] B. Eller, P. Stanfill, R. Turner, S. Whitcomb, D. Swihart, A. Burns, and K. Hobbs, "Test and evaluation of a modified f-16 analog flight control computer," in *AIAA Infotech@ Aerospace Conference*, 2013, p. 4726.

[203] A. Burns, D. Harper, A. F. Barfield, S. Whitcomb, and B. Jurusik, "Auto gcas for analog flight control system," in *2011 IEEE/AIAA 30th Digital Avionics Systems Conference*, IEEE, 2011, pp. 8C5–1.

[204] J. Wadley, S. Jones, D. Stoner, E. Griffin, D. Swihart, K. Hobbs, A. Burns, and J. Bier, "Development of an automatic aircraft collision avoidance system for fighter aircraft," in *AIAA Infotech@ Aerospace Conference*, 2013, p. 4727.

[205] K. A. Hoff and M. Bashir, "Trust in automation: Integrating empirical evidence on factors that influence trust," *Human Factors*, vol. 57, no. 3, pp. 407–434, 2015.

[206] P. Madhavan and D. A. Wiegmann, "Similarities and differences between human–human and human–automation trust: An integrative review," *Theoretical Issues in Ergonomics Science*, vol. 8, no. 4, pp. 277–301, 2007.

[207] Federal Aviation Administration (FAA), "Safety skies general aviation (ga) controlled flight into terrain (CFIT) joint safety implementation team (JSIT) final report," Federal Aviation Administration (FAA), Tech. Rep., 2000.

[208] M. Wilkins, *Automatic collision avoidance technologies (ACAT) update: Auto-GCAS and the fighter risk reduction program (FRRP)*, Briefing, Jan. 2007.

[209] W. H. Harman, "Tcas- a system for preventing midair collisions," *The Lincoln Laboratory Journal*, vol. 2, no. 3, pp. 437–457, 1989.

[210] T. Williamson and N. A. Spencer, "Development and operation of the traffic alert and collision avoidance system (tcas)," *Proceedings of the IEEE*, vol. 77, no. 11, pp. 1735–1744, 1989.

[211] Federal Aviation Administration, "Introduction to tcasii: Version 7.1," U.S. Department of Transportation Federal Aviation Administration, Tech. Rep., 2011.

[212] C. Munoz, A. Narkawicz, and J. Chamberlain, "A tcas-ii resolution advisory detection algorithm," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013, p. 4622.

[213] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, "Next-generation airborne collision avoidance system," Massachusetts Institute of Technology-Lincoln Laboratory Lexington United States, Tech. Rep., 2012.

[214] J. E. Holland, M. J. Kochenderfer, and W. A. Olson, "Optimizing the next generation collision avoidance system for safe, suitable, and acceptable operational performance," *Air Traffic Control Quarterly*, vol. 21, no. 3, pp. 275–297, 2013.

[215] J.-B. Jeannin, K. Ghorbal, Y. Kouskoulas, A. Schmidt, R. Gardner, S. Mitsch, and A. Platzer, "A formally verified hybrid system for safe advisories in the next-generation airborne collision avoidance system," *International Journal on Software Tools for Technology Transfer*, vol. 19, no. 6, pp. 717–741, 2017.

[216] O. Shakernia, W.-Z. Chen, S. Graham, J. Zvanya, A. White, N. Weingarten, and V. Raska, "Sense and avoid (saa) flight test and lessons learned," in *AIAA Infotech@ Aerospace 2007 Conference and Exhibit*, 2007, p. 3003.

[217] Advisory Circular, "Pilot's role in collision avoidance," Federal Aviation Administration, 800 Independence Avenue SW, Washington, DC, 20591, Tech. Rep. AC 90-48D, Apr. 2016, https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_90-48D.pdf.

[218] C. Munoz, A. Narkawicz, J. Chamberlain, M. C. Consiglio, and J. M. Upchurch, "A family of well-clear boundary models for the integration of uas in the nas," in *14th AIAA Aviation Technology, Integration, and Operations Conference*, 2014, p. 2412.

[219] C. Muñoz, A. Narkawicz, G. Hagen, J. Upchurch, A. Dutle, M. Consiglio, and J. Chamberlain, "Daidalus: Detect and avoid alerting logic for unmanned systems," in *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, IEEE, 2015, 5A1–1.

[220] Y. Ikeda, B Nguyen, A. Barfield, B Sundqvist, and S Jones, "Automatic air collision avoidance system," in *SICE 2002. Proceedings of the 41st SICE Annual Conference*, IEEE, vol. 1, 2002, pp. 630–635.

[221] A. Weiss, C. Petersen, M. Baldwin, R. S. Erwin, and I. Kolmanovsky, "Safe positively invariant sets for spacecraft obstacle avoidance," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 720–732, 2014.

[222] A. B. Poore, J. M. Aristoff, J. T. Horwood, R. Armellin, W. T. Cerven, Y. Cheng, C. M. Cox, R. S. Erwin, J. H. Frisbee, M. D. Hejduk, *et al.*, "Covariance and uncertainty realism in space surveillance and tracking," Numerica Corporation Fort Collins United States, Tech. Rep., 2016.

[223] J. L. Junkins, M. R. Akella, and K. T. Alfrined, "Non-gaussian error propagation in orbital mechanics," *Guidance and Control 1996*, pp. 283–298, 1996.

[224] K. J. DeMars and M. K. Jah, "Probabilistic initial orbit determination using gaussian mixture models," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 5, pp. 1324–1335, 2013.

[225] M. L. Psiaki, R. M. Weisman, and M. K. Jah, "Gaussian mixture approximation of angles-only initial orbit determination likelihood function," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 11, pp. 2807–2819, 2017.

[226] K. J. DeMars, R. H. Bishop, and M. K. Jan, "A splitting gaussian mixture method for the propagation of uncertainty in orbital mechanics," *Spaceflight Mechanics*, vol. 140, 2011.

[227] K. J. DeMars, Y. Cheng, and M. K. Jah, "Collision probability with gaussian mixture orbit uncertainty," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 3, pp. 979–985, 2014.

[228] A. Burns, K. Hobbs, J. Bier, and S. Whitcomb, "Advanced capabilities for the analog flight control f-16," in *NATO Sensors and Electronics Technology Panel 168 Symposium*, 2012.

[229] R. Lehmann, J. Wadley, D. Von Bose, R. Turner, K. Hobbs, M. Coy, and A. Burns, "Simulation tools for the development and evaluation of an automatic air collision avoidance system," in *AIAA Infotech@ Aerospace Conference*, 2013.

[230] S. Clark, *Attitude control failures led to break-up of japanese astronomy satellite*, `https://spaceflightnow.com/2016/04/18/spinning-japanese-astronomy-satellite-may-be-beyond-saving/`, Apr. 2016.

[231] D. H. Scheidt, *System for testing of autonomy in complex environments*, US Patent 9,443,436, Sep. 2016.

[232] C. Jewison and R. S. Erwin, "A spacecraft benchmark problem for hybrid control and estimation," in *Proceedings of the IEEE 55th Conference on Decision and Control (CDC)*, Las Vegas, NV: IEEE, Dec. 2016, pp. 3300–3305.

[233] A. Weiss, M. Baldwin, R. S. Erwin, and I. Kolmanovsky, "Model predictive control for spacecraft rendezvous and docking: Strategies for handling constraints and case studies," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1638–1647, 2015.

[234] B. Evans, "Dawn of a new era," in *Partnership in Space*, Springer, 2014, pp. 453–482.

[235] D. Holland, "6.4. 1 a case study of the near-catastrophic mir-progress 234 collision with emphasis on the human factors/systems-level issues surrounding this mishap," in *INCOSE International Symposium*, Wiley Online Library, vol. 12, 2002, pp. 820–827.

[236] J. Oberg, "Shuttle-mir's lessons for the international space station," *IEEE Spectrum*, vol. 35, no. 6, pp. 28–37, 1998.

[237] L. D. thomas, "Selected systems engineering process deificiencies and their consequences," National Aeronautics and Space Administration George C. Marshall Space Flight Center, Tech. Rep., 2007.

[238] J. A. Reiter and D. B. Spencer, "Solutions to rapid collision-avoidance maneuvers constrained by mission performance requirements," *Journal of Spacecraft and Rockets*, vol. 55, no. 4, pp. 1040–1048, 2018.

[239] E Frazzoli, M. Dahleh, E Feron, and R Kornfeld, "A randomized attitude slew planning algorithm for autonomous spacecraft," in *AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics Montreal, Quebec, 2001.

[240] G. W. Hill, "Researches in the lunar theory," *American journal of Mathematics*, vol. 1, no. 1, pp. 5–26, 1878.

[241] W. Clohessy and R. Wiltshire, "Terminal guidance system for satellite rendezvous," *Journal of the Aerospace Sciences*, vol. 27, no. 9, pp. 653–658, 1960.

[242] *Update to Parts 2 and 25 Concerning Non-Geostationary, Fixed-Satellite Service Systems and Related Matters*, IB Docket No. 16-408, 2016.

[243] D. Lee, A. K. Sanyal, and E. A. Butcher, "Asymptotic tracking control for spacecraft formation flying with decentralized collision avoidance," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 587–600, 2014.

[244] M. Mancini, N. Bloise, E. Capello, and E. Punta, "Sliding mode control techniques and artificial potential field for dynamic collision avoidance in rendezvous maneuvers," *arXiv preprint arXiv:1906.10945*, 2019.

[245] S Di Cairano, H Park, and I Kolmanovsky, "Model predictive control approach for guidance of spacecraft rendezvous and proximity maneuvering," *International Journal of Robust and Nonlinear Control*, vol. 22, no. 12, pp. 1398–1427, 2012.

[246] J. B. Mueller, P. R. Griesemer, and S. J. Thomas, "Avoidance maneuver planning incorporating station-keeping constraints and automatic relaxation," *Journal of Aerospace Information Systems*, vol. 10, no. 6, pp. 306–322, 2013.

[247] E. Denenberg and P. Gurfil, "Debris avoidance maneuvers for spacecraft in a cluster," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 6, pp. 1428–1440, 2017.

[248] K. Lee and C. Park, "Spacecraft collision avoidance with constrained control via discrete-time generating functions," in *2018 26th Mediterranean Conference on Control and Automation (MED)*, IEEE, 2018, pp. 1–5.

[249] A. Weiss, M. Baldwin, C. Petersen, R. S. Erwin, and I. Kolmanovsky, "Spacecraft constrained maneuver planning for moving debris avoidance using positively invariant constraint admissible sets," in *2013 American Control Conference*, IEEE, 2013, pp. 4802–4807.

[250] S. Alfano, "Collision avoidance maneuver planning tool," in *15th AAS/AIAA Astrodynamics Specialist Conference*, 2005, pp. 7–11.

[251] C. Bombardelli and J. Hernando-Ayuso, "Optimal impulsive collision avoidance in low earth orbit," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 2, pp. 217–225, 2015.

[252] A. Mazzoleni, M. Luengo, J. A. Santos, A. Iturri, I. Bueno, P. Pisabarro, and F. Pirondini, "Collision avoidance operations of deimos-1 and deimos-2 missions," in *14th International Conference on Space Operations*, 2016, p. 2450.

[253] K. Lee, C. Park, and S.-Y. Park, "Near-optimal guidance and control for spacecraft collision avoidance maneuvers," in *AIAA/AAS Astrodynamics Specialist Conference*, 2014, p. 4114.

[254] D. J. Kessler, "Sources of orbital debris and the projected environment for future spacecraft," *Journal of Spacecraft and Rockets*, vol. 18, no. 4, pp. 357–360, 1981.

[255] R. P. Patera and G. E. Peterson, "Space vehicle maneuver method to lower collision risk to an acceptable level," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 2, pp. 233–237, 2003.

[256] M. Gnat and P. Willburger, "Operations for parallel satellite support," *Space Operations: Experience, Mission Systems, and Advanced Concepts*, p. 285, 2012.

[257] "AFI 51-503: Aerospace and Ground Accident Investigations," United States Air Force, Air Force Instruction, 2015.

[258] P. S. Morgan, "Fault protection techniques in jpl spacecraft," 2005.

[259] NASA Orbital Debris Program Office, "Accidental collisions of cataloged satellites identified," *The Orbital Debris Quarterly News*, vol. 9, no. 2, pp. 1–2, 2005.

[260] F Alby, E Lansard, and T Michal, "Collision of cerise with space debris," in *Second European Conference on Space Debris*, vol. 393, 1997, p. 589.

[261] N. L. Johnson, E. Stansbery, D. O. Whitlock, K. J. Abercromby, and D. Shoots, "History of on-orbit satellite fragmentations," 2008.

[262] T. Wang, "Analysis of debris from the collision of the cosmos 2251 and the iridium 33 satellites," *Science & Global Security*, vol. 18, no. 2, pp. 87–118, 2010.

[263] T. Kelso, N. Parkhomenko, V. Shargorodsky, V. Vasiliev, V. Yurasov, A. Nazarenko, S Tanygin, and R. Hiles, "What happened to blits? an analysis of the 2013 jan 22 event," in *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, vol. 4, 2013.

[264] D. S. F. Portree and J. P. Loftus, "Orbital debris: A chronology," NASA Technical Report, Tech. Rep., 1999.

[265] E. J. Hoffman, W. Ebert, M. Femiano, H. Freeman, C. Gay, C. Jones, P. Luers, and J. Palmer, "The near rendezvous burn anomaly of december 1998," *Applied Physics Laboratory, Johns Hopkins University, Tech. Rep*, 1999.

[266] J. D. Koenig, *Exclusion zone guidance method for spacecraft*, US Patent 7,795,566, Sep. 2010.

[267] T. Burk, "Managing cassini safe mode attitude at saturn," in *AIAA Guidance, Navigation, and Control Conference*, 2010, p. 7558.

[268] K. A. Larson, K. M. McCalmont, C. A. Peterson, and S. E. Ross, "Kepler mission operations response to wheel anomalies," in *SpaceOps 2014 Conference*, 2014, p. 1882.

[269] J. R. Wertz and W. J. Larson, *Space Mission Analysis and Design, Third Edition*. Microcosm, 1999.

[270] F. Barfield, "Autonomous collision avoidance: The technical requirements," in *National Aerospace and Electronics Conference, 2000. NAECON 2000. Proceedings of the IEEE 2000*, IEEE, 2000, pp. 808–813.

[271] C. W. Johnson, "The natural history of bugs: Using formal methods to analyse software related failures in space missions," in *International Symposium on Formal Methods*, Springer, 2005, pp. 9–25.

[272] M. A. Skoog, L. R. Hook, and W. Ryan, "Leveraging astm industry standard f3269-17 for providing safe operations of a highly autonomous aircraft," in *2020 IEEE Aerospace Conference*, IEEE, 2020, pp. 1–7.

[273] M. Neher, "From interval analysis to taylor models-an overview," *Proc. IMACS 2005*, 2005.

[274] M. Neher, K. R. Jackson, and N. S. Nedialkov, "On taylor model based integration of odes," *SIAM Journal on Numerical Analysis*, vol. 45, p. 2007,

[275] K. Y. Rozier, J. Schumann, and C. Ippolito, "Intelligent Hardware-Enabled Sensor and Software Safety and Health Management for Autonomous UAS," NASA, NASA Ames Research Center, Moffett Field, CA 94035, USA, Technical Memorandum NASA/TM-2015-218817, May 2015.

[276] J. Schumann, P. Moosbrugger, and K. Y. Rozier, "R2U2: Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems," in *Proceedings of the 15th International Conference on Runtime Verification (RV15)*, Vienna, Austria: Springer-Verlag, Sep. 2015.

[277] J. Milton and J. C. Arnold, *Introduction to Probability and Statistics*. The McGraw-Hill Companies, Inc, New York, NY, 2003.

[278] E. W. Dijkstra, "Cooperating sequential processes," in *The origin of concurrent programming*, Springer, 1968, pp. 65–138.

[279] ——, "Hierarchical ordering of sequential processes," in *The origin of concurrent programming*, Springer, 1971, pp. 198–227.

[280] G. A. Ringwood, "Parlog86 and the dining logicians," *Communications of the ACM*, vol. 31, no. 1, pp. 10–25, 1988.

[281] J. Mattis, "Summary of the national defense strategy of the united states of america," *Washington, DC*, pp. 1–11, 2018.

[282] S. A. Hildreth and A. Arnold, "Threats to us national security interests in space: Orbital debris mitigation and removal," Library of Congress, Congressional Research Service, Washington DC, 2014.

[283] D. Kessler and B. G. Cour-Palais, "Collision frequency of artificial satellites: The creation of a debris belt," *Journal of Geophysical Research*, vol. 83, no. A6, pp. 2637–2646, 1978.

[284] D. J. Kessler, N. L. Johnson, J. Liou, and M. Matney, "The kessler syndrome: Implications to future space operations," *Advances in the Astronautical Sciences*, vol. 137, no. 8, p. 2010, 2010.

[285] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000.

[286] M. Ballin and D. Wing, "Traffic aware strategic aircrew requests (TASAR)," in *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2012, p. 5623.

[287] T Flohrer, V Braun, H Krag, K Merz, S Lemmens, B. B. Virgili, and Q Funke, "Operational collision avoidance at esoc," *Deutscher Luft-und Raumfahrtkongress, Rostok, Germany*, no. 0270, 2015.

[288] A. T. Monham, P. L. Righetti, and R. Dyer, "Life or death? maximising mission lifetime return in the space debris era," in *SpaceOps 2014 Conference*, 2014, p. 1773.

[289] G. Hagen, R. Butler, and J. Maddalon, "Stratway: A modular approach to strategic conflict resolution," in *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, including the AIAA Balloon Systems Conference and 19th AIAA Lighter-Than*, 2011, p. 6892.

[290] S. Koczo and D. Wing, "An operational safety and certification assessment of a TASAR EFB application," in *2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC)*, IEEE, 2013, 2A1–1.

[291] S. Koczo Jr, "TASAR certification and operational approval requirements-analyses and results," 2015.

[292] D. J. Wing, "Achieving TASAR operational readiness," in *15th AIAA Aviation Technology, Integration, and Operations Conference*, 2015, p. 3400.

[293] G. E. Hagen, R. W. Butler, and J. M. Maddalon, "The stratway program for strategic conflict resolution: User's guide," 2016.

[294] D. J. Wing, K. A. Burke, J. Henderson, R. A. Vivona, and J. Woodward, "Initial implementation and operational use of TASAR in alaska airlines flight operations," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3043.

[295] D. J. Wing, K. A. Burke, K. Ballard, J. Henderson, and J. Woodward, "Initial tasar operations onboard alaska airlines," in *AIAA Aviation 2019 Forum*, 2019, p. 3613.

[296] L. K. Newman, "The nasa robotic conjunction assessment process: Overview and operational experiences," *Acta Astronautica*, vol. 66, no. 7-8, pp. 1253–1261, 2010.

[297] L. K. Newman, R. C. Frigm, M. G. Duncan, and M. D. Hejduk, "Evolution and implementation of the nasa robotic conjunction assessment risk analysis concept of operations," 2014.

[298] R. W. Butler, G. E. Hagen, and J. M. Maddalon, "The chorus conflict and loss of separation resolution algorithms," 2013.

[299] S. Owre, J. M. Rushby, and N. Shankar, "Pvs: A prototype verification system," in *International Conference on Automated Deduction*, Springer, 1992, pp. 748–752.

[300] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, Kobe, Japan, vol. 3, 2009, p. 5.

[301] S. Rasmussen, D. Kingston, and L. Humphrey, "A brief introduction to unmanned systems autonomy services (uxas)," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2018, pp. 257–268.

[302] C. W. Heisey, A. G. Hendrickson, B. J. Chludzinski, R. E. Cole, M. Ford, L. Herbek, M. Ljungberg, Z. Magdum, D Marquis, A. Mezhirov, *et al.*, "A reference software architecture to support unmanned aircraft integration in the national airspace system," *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1-4, pp. 41–55, 2013.

[303] G. W. Bice, M. A. Skoog, and J. D. Howard, *Aircraft ground collision avoidance and autorecovery systems device*, US Patent 4,924,401, May 1990.

[304] D. E. Swihart, "Automatic ground collision avoidance system (auto gcas)," in *Proceedings of the 13th WSEAS International Conference on Systems*, ser. ICS'09, Rodos, Greece: World Scientific, Engineering Academy, and Society (WSEAS), 2009, pp. 429–433, ISBN: 978-960-474-097-0.

[305] Federal Aviation Administration's Office of Commercial Space Transportation (FAA AST), "The annual compendium of commercial space transportation: 2018," Federal Aviation Administration, 800 Independence Avenue SW, Washington, DC, 20591, Tech. Rep., Jan. 2018, https://www.faa.gov/about/office_org/headquarters_offices/ast/media/2018_AST_Compendium.pdf.

[306] Federal Communications Commission, "Application for approval for orbital deployment and operating authority for the spacex ngso satellite system," *Federal Communications Commission Memorandum Opinion, Order and Authorization*, Mar. 29, 2018.

[307] *USA Space Debris Environment, Operations, and Research Updates*, Vienna, Austria, Jan. 2018.

[308] J.-C. Liou and N. Johnson, "Characterization of the cataloged fengyun-1c fragments and their long-term effect on the leo environment," *Advances in Space Research*, vol. 43, no. 9, pp. 1407–1415, 2009.

[309] J Eberhart, "ASAT target was working research satellite," *Science News*, pp. 197–197, 1985.

[310] T. Kelso, "Analysis of the 2007 chinese asat test and the impact of its debris on the space environment," in *8th Advanced Maui Optical and Space Surveillance Technologies Conference, Maui, HI*, vol. 7, 2007.

[311] J. Mackey, "Recent us and chinese antisatellite activities," Air University, Air Force Research Institute, Maxwell AFB, AL, Tech. Rep., 2009.

[312] *Space Traffic Management and Orbital Debris: A Path Forward to Ensure Safe and Uninterrupted Space Operations*, Daytona Beach, Florida, Jan. 2018.

[313] W. H. Ailor and G. E. Peterson, "Collision avoidance as a debris mitigation measure," *Science and Technology Series*, vol. 110, pp. 193–203, 2005.

[314] H. Krag, T. Flohrer, K. Merz, S. Lemmens, B. Bastida Virgili, Q. Funke, and V. Braun, "Esa?s modernised collision avoidance service," in *14th International Conference on Space Operations*, 2016, p. 2449.

[315] P. Jiménez, F. Thomas, and C. Torras, "3d collision detection: A survey," *Computers & Graphics*, vol. 25, no. 2, pp. 269–285, 2001.

[316] K. Hobbs and S. Bak, "Efficient n-to-n collision detection using 4D AABB trees," unpublished, 2019.

[317] A Foisy and V Hayward, "A safe swept volume method for collision detection," in *International Symposium of Robotics Research*, 1994, pp. 62–68.

[318] E. G. Gilbert and S. Hong, "A new algorithm for detecting the collision of moving objects," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, IEEE, 1989, pp. 8–14.

[319] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtree: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd Annual Conference on Computer graphics and interactive techniques*, ACM, 1996, pp. 171–180.

[320] S. Gottschalk, "Separating axis theorem," Technical Report TR96-024, Department of Computer Science, UNC Chapel Hill, Tech. Rep., 1996.

[321] L. Jaulin, *Applied Interval Analysis: with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer Science & Business Media, 2001, vol. 1.

[322] S. Redon, A. Kheddar, and S. Coquillart, "Fast continuous collision detection between rigid bodies," in *Computer graphics forum*, Wiley Online Library, vol. 21, 2002, pp. 279–287.

[323] A. P. Del Pobil, M. A. Serna, and J. Llovet, "A new representation for collision avoidance and detection," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, IEEE, 1992, pp. 246–251.

[324] P. S. Duggirala, C. Fan, M. Potok, B. Qi, S. Mitra, M. Viswanathan, S. Bak, S. Bogomolov, T. T. Johnson, L. V. Nguyen, C. Schilling, A. Sogokon, H.-D. Tran, and W. Xiang, "Tutorial: Software tools for hybrid systems verification, transformation, and synthesis: C2e2, hyst, and tulip," in *Proceedings of the Multi-Conference on Systems and Control*, IEEE, 2016.

[325] S. Alfano, "Determining satellite close approaches, part 2," *Journal of the Astronautical Sciences*, vol. 42, pp. 143–152, 1994.

[326] R. P. Patera, "Space vehicle conflict-avoidance analysis," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 2, pp. 492–498, 2007.

[327] K. T. Alfriend, M. R. Akella, J. Frisbee, J. L. Foster, D.-J. Lee, and M. Wilkins, "Probability of collision error analysis," *Space Debris*, vol. 1, no. 1, pp. 21–35, 1999.

[328] K. Chan, "Spacecraft collision probability for long-term encounters," *Advances in the Astronautical Sciences*, vol. 116, no. 1, pp. 767–784, 2003.

[329] T. Schildknecht, R. Musci, and T. Flohrer, "Properties of the high area-to-mass ratio space debris population at high altitudes," *Advances in Space Research*, vol. 41, no. 7, pp. 1039–1045, 2008.

[330] A Rossi and G. Valsecchi, "Collision risk against space debris in earth orbits," *Celestial Mechanics and Dynamical Astronomy*, vol. 95, no. 1-4, pp. 345–356, 2006.

[331] B. A. Jones and R. Weisman, "Multi-fidelity orbit uncertainty propagation," *Acta Astronautica*, 2018.

[332] E. Delande, J. Houssineau, and M. Jah, "Physics and human-based information fusion for improved resident space object tracking," *arXiv preprint arXiv:1802.07408*, 2018.

[333] D. J. Clancy, "Associating data in system of systems using measures of information," *Journal of Aerospace Information Systems*, vol. 11, no. 4, pp. 145–169, 2014.

[334] T Schildknecht, R Musci, W Flury, J Kuusela, J de Leon, and L. d. F. D. Palmero, "Properties of the high area-to-mass ratio space debris population in geo," in *2005 AMOS Technical Conference Proceedings, Kihei, Maui, HI*, 2005.

[335] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *City*, vol. 1, no. 2, p. 1, 2007.

[336] R. P. Patera, "General method for calculating satellite collision probability," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 4, pp. 716–722, 2001.

[337] ——, "Satellite collision probability for nonlinear relative motion," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 5, pp. 728–733, 2003.

[338] ——, "Calculating collision probability for arbitrary space vehicle shapes via numerical quadrature," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 6, pp. 1326–1328, 2005.

[339] ——, "Space vehicle conflict probability for ellipsoidal conflict volumes," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 6, pp. 1819–1822, 2007.

[340] P. M. Esfahani, D. Chatterjee, and J. Lygeros, "On a problem of stochastic reach-avoid set characterization.," in *CDC-ECE*, 2011, pp. 7069–7074.

[341] I. M. Mitchell and J. A. Templeton, "A toolbox of hamilton-jacobi solvers for analysis of nondeterministic continuous and hybrid systems," in *International Workshop on Hybrid Systems: Computation and Control*, Springer, 2005, pp. 480–494.

[342] S. Summers and J. Lygeros, "Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem," *Automatica*, vol. 46, no. 12, pp. 1951–1961, 2010.

[343] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, 2008.

[344] A. Kurzhanski and P Varaiya, "Reachability under uncertainty," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, IEEE, vol. 2, 2002, pp. 1982–1987.

[345] B. Römgens, E. Mooij, and M. Naeije, "Satellite collision avoidance prediction using verified interval orbit propagation," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 3, pp. 821–832, 2013.

[346] E. R. George and S. Harvey, "A comparison of satellite conjunction analysis screening tools," Air Force Research Laboratory, Kirtland AFB, NM, Tech. Rep., 2011.

[347] A. B. Jenkin, "Effect of orbit data quality on the feasibility of collision risk management," *Journal of spacecraft and rockets*, vol. 41, no. 4, pp. 677–683, 2004.

[348] B. Kelly and S. De Picciotto, "Probability based optimal collision avoidance maneuvers," in *Space 2005*, 2005, p. 6775.

[349] D. Mishne and E. Edlerman, "Collision-avoidance maneuver of satellites using drag and solar radiation pressure," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 5, pp. 1191–1205, 2017.

[350] B.-S. Lee, "Geo satellite collision avoidance maneuver strategy against inclied gso satellite," in *SpaceOps 2012*, 2012, p. 1 294 441.

[351] C. Petersen, A. Jaunzemis, M. Baldwin, M. Holzinger, and I. Kolmanovsky, "Model predictive control and extended command governor for improving robustness of relative motion guidance and control," *Advances In The Astronautical Sciences*, vol. 152, pp. 701–718, 2014.

[352] J. Mueller, "Onboard planning of collision avoidance maneuvers using robust optimization," in *AIAA Infotech@ Aerospace Conference*, 2009, p. 2051.

[353] T. T. Johnson, J. Green, S. Mitra, R. Dudley, and R. S. Erwin, "Satellite rendezvous and conjunction avoidance: Case studies in verification of nonlinear hybrid systems," in *International Symposium on Formal Methods*, Springer, 2012, pp. 252–266.

[354] S. Bak, S. Bogomolov, T. A. Henzinger, T. T. Johnson, and P. Prakash, "Scalable static hybridization methods for analysis of nonlinear systems," in *International Conference on Hybrid Systems: Computation and Control*, Vienna, Austria, 2016.

[355] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "Algorithmic analysis of nonlinear hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 540–554, 1998.

[356] B. HomChaudhuri, M. Oishi, M. Shubert, M. Baldwin, and R. S. Erwin, "Computing reach-avoid sets for space vehicle docking under continuous thrust," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*, IEEE, 2016, pp. 3312–3318.

[357] N. Chan and S. Mitra, "Verifying safety of an autonomous spacecraft rendezvous mission," in *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, G. Frehse and M. Althoff, Eds., ser. EPiC Series in Computing, vol. 48, EasyChair, 2017.

[358] T. Mulder, "Orbital express autonomous rendezvous and capture flight operations, part 2 of 2: Ar&c exercise 4, 5, and end-of-life," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2008, p. 6768.

[359] M. R. Leinz, C.-T. Chen, P. Scott, W. Gaumer, P. Sabasteanski, and M. Beaven, "Modeling, simulation, testing, and verification of the orbital express autonomous rendezvous and capture sensor system (arcss)," in *Sensors and Systems for Space Applications II*, International Society for Optics and Photonics, vol. 6958, 2008, p. 69580C.

[360] R. B. Friend, "Orbital express program summary and mission overview," in *Sensors and Systems for space applications II*, International Society for Optics and Photonics, vol. 6958, 2008, p. 695 803.

[361] K. Lesser, M. Oishi, and R. S. Erwin, "Stochastic reachability for control of spacecraft relative motion," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, IEEE, 2013, pp. 4705–4712.

[362] A. Nemirovski and A. Shapiro, "Scenario approximations of chance constraints," in *Probabilistic and randomized methods for design under uncertainty*, Springer, 2006, pp. 3–47.

[363] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams, "A probabilistic particle-control approximation of chance-constrained stochastic predictive control," *IEEE transactions on Robotics*, vol. 26, no. 3, pp. 502–517, 2010.

[364] L. Blackmore and M. Ono, "Convex chance constrained predictive control without sampling," in *AIAA Guidance, Navigation, and Control Conference*, 2009, p. 5876.

[365] M. P. Vitus and C. J. Tomlin, "Closed-loop belief space planning for linear, gaussian systems," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 2152–2159.

[366] G. R. Frey, C. D. Petersen, F. A. Leve, I. V. Kolmanovsky, and A. R. Girard, "Constrained spacecraft relative motion planning exploiting periodic natural motion trajectories and invariance," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 12, pp. 3100–3115, 2017.

[367] G. R. Frey, C. D. Petersen, F. A. Leve, A. R. Girard, and I. V. Kolmanovsky, "Invariance-based spacecraft relative motion planning incorporating bounded disturbances and mini-

mum thrust constraints," in *2018 Annual American Control Conference (ACC)*, IEEE, 2018, pp. 658–663.

[368] G. R. Frey, C. D. Petersen, F. A. Leve, I. V. Kolmanovsky, and A. R. Girard, "Incorporating periodic and non-periodic natural motion trajectories into constrained invariance-based spacecraft relative motion planning," in *Control Technology and Applications (CCTA), 2017 IEEE Conference on*, IEEE, 2017, pp. 1811–1816.

[369] G. R. Frey, C. D. Petersen, F. A. Leve, E. Garone, I. V. Kolmanovsky, and A. R. Girard, "Time shift governor for coordinated control of two spacecraft formations," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 296–301, 2016.

[370] ——, "Parameter governors for coordinated control of n-spacecraft formations," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 11, pp. 3020–3025, 2017.

[371] G. R. Hintz, "Survey of orbit element sets," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 3, pp. 785–790, 2008.

[372] R. R. Bate, D. D. Mueller, and J. E. White, *Fundamentals of Astrodynamics*. Courier Corporation, 1971.

[373] W. E. Wiesel, *Spaceflight Dynamics*, 3rd ed. Beavercreek, Ohio, 45434: Aphelion Press, 2010, ISBN: 1452879591.

[374] R. A. Broucke and P. J. Cefola, "On the equinoctial orbit elements," *Celestial Mechanics*, vol. 5, no. 3, pp. 303–310, 1972.

[375] M. Walker, B. Ireland, and J. Owens, "A set of modified equinoctial orbit elements," *Celestial Mechanics*, vol. 36, no. 4, pp. 409–419, 1985.

[376] M. Walker, "A set of modified equinoctial orbit elements," *Celestial Mechanics and Dynamical Astronomy*, vol. 38, no. 4, pp. 391–392, 1986.

[377] S. P. Altman, "A unified state model of orbital trajectory and attitude dynamics," *Celestial mechanics*, vol. 6, no. 4, pp. 425–446, 1972.

[378] V Vittaldev, E Mooij, and M. Naeije, "Unified state model theory and application in astrodynamics," *Celestial Mechanics and Dynamical Astronomy*, vol. 112, no. 3, pp. 253–282, 2012.