Safety-driven Early Concept Analysis and Development

by

Cody Harrison Fleming

B.S. Engineering
Hope College, 2003

M.Eng. Civil & Environmental Engineering
Massachusetts Institute of Technology, 2004

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
at the
Massachusetts Institute of Technology

February 2015

Signature of Author: _____
Department of Aeronautics and Astronautics
19 January, 2015

Certified by: _____
Nancy G. Leveson
Professor of Aeronautics and Astronautics and Engineering Systems
Thesis Committee Chair

Certified by: _____
Jeffrey A. Hoffman
Professor of the Practice, Aeronautics and Astronautics
Thesis Committee Member

Certified by: _____
James K. Kuchar
Leader, Air Traffic Control Systems Group, Lincoln Laboratory
Thesis Committee Member

Accepted by: _____
Paulo C. Lozano
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

[Page intentionally left blank]

To Morris.

For shining a light on my life.

[Page intentionally left blank]

Safety-driven Early Concept Analysis and Development

by

Cody Harrison Fleming

Abstract

As aerospace systems become increasingly complex and the roles of human operators and autonomous software continue to evolve, traditional safety-related analytical methods are becoming inadequate. Traditional hazard analysis tools are based on an accident causality model that does not capture many of the complex behaviors found in modern engineered systems. Additionally, these traditional approaches are most effective during late stages of system development, when detailed design information is available. However, system safety cannot cost-effectively be assured by discovering problems at these late stages and adding expensive updates to the design. Rather, safety should be designed into the system from its very conception. The primary barrier to achieving this objective is the lack of effectiveness of the existing analytical tools during early concept development.

This thesis introduces a new technique, which is based on a more powerful model of accident causality that can capture behaviors that are prevalent in these complex, software-intensive systems. The proposed approach builds on a new accident causality model, called Systems-Theoretic Accident Model and Process, developing a methodology on the model so that it can be applied during the early concept development stages of systems engineering.

The goals are to (1) develop rigorous, systematic tools for the analysis of future concepts in order to identify hazardous scenarios, and (2) extend these tools to assist stakeholders in the development of concepts using a safety-driven approach.

This work first develops a methodology for hazard analysis of a concept of operations (ConOps) using control theory to generate a model of that ConOps. Formal, systems-theoretic concepts such as hierarchy, emergence, communication, and coordination are used to analyze the model and identify hazards in the concept. These hazardous scenarios then guide the development of requirements and the generation of a system architecture, defined as a hierarchical control structure.

This model-based approach represents a significant departure from the state of the art; in the new approach a concept is defined, developed, and analyzed according to a control theoretic model rather than free form, natural language text. The power of the proposed approach—called Systems-Theoretic Early Concept Analysis—is demonstrated on a concept currently being developed by the United States Federal Aviation Administration.

Thesis Supervisor: Nancy Leveson
Title: Professor of Aeronautics and Astronautics

[Page intentionally left blank]

Acknowledgements

Professor Nancy Leveson accepted me into this program. She saw in me a mindset that is synergistic with the lab, and she nurtured me along and guided me through some rough terrain. I cannot express enough how much I appreciate her vision, as well as her ability to give us the space to find and work through our own problems. Professor Leveson provided many opportunities above and beyond what is typically given to a graduate student, and these enriched my experience here at MIT. Thank you, Nancy.

Thanks also to my committee members, Professor Jeff Hoffman and Dr. Jim Kuchar. Without fail, at every committee meeting they asked questions and offered comments about things I had never thought of. This is the true reason for having a committee. Professor Sheila Widnall provided support and inspiration in countless ways, from early coursework during the PhD program, all the way to the end and beyond. Professor Hamsa Balakrishnan took time out of her extremely busy schedule, and took a foray into relatively unfamiliar subject matter, to provide valuable feedback for this thesis.

What would the graduate student experience be without a lab? John Thomas, the maestro, you are the kindest and most patient person I have ever met in an environment like this, and you are quite clever, too. Blandine Antoine and John Helferich always provided lively conversation and important feedback. Adam Williams, Bill Young, Dan Montes, and Kip Johnson, thank you for all your help and best wishes on your own PhD journeys. Dajiang, Melissa, Seth, Cameron, Aubrey, Ibrahim, and Connor—many thanks for your collaboration, sharpening me with your questions, and showing up to the office with smiles on your faces.

Several folks in the broader MIT community deserve acknowledgement. Yves Boussemart provided a soft landing when I arrived here and ensured that my stress level never got too high; Ani Mazumdar picked up right where Yves left off. Sophia Hasenfus and Anthony Zolnik provided a level of sanity within the department and made sure I was always comfortable and smiling. Kevin Ford provided a breath of fresh air, almost weekly. I admired the work of John Hansman and Oli de Weck from afar, and many of their students pointed me in the right direction. Thank you to Luke Jensen and Dani Selva.

Roger Veldman help start me on my academic journey long ago, while I was an undergraduate, and has walked alongside me since. Brian Potter was a most enthusiastic mentor during my foray in the "real world" and was equally enthusiastic and encouraging when I decided to leave. Evan Lapointe was there my first time around here at MIT, making sure I did well enough that they would let me come back and

supporting me the second time around. Likewise for Jerome Connor.

Of course I literally and figuratively would not be here without family and friends. Mom, you taught me how to love learning, among many other things. Brady and Kiley, you taught me about love, loyalty, and fun. Proper respect to Sam and E(than) from way back, Chad and Jamie in the not-too-distant past, and D(iana) and Matt more recently.

Last, and certainly not least, thanks to my "own" family. Sarah, you are so beautiful, sweet, supportive, and loving. Your intellectual curiosity inspires me every day, reminding me constantly of why I would want to be in academia in the first place. What more could a guy ask for? To the next adventure!

[Page intentionally left blank]

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> *Safety must be designed and built into airplanes, just as are performance, stability, and structural integrity.* [Stieglitz, 1948]

Often the perception among engineers and other stakeholders is that safety is expensive. Safety-related features are also seen as intrusive because they seem to result in reduced performance, increased weight, or unnecessary complexity. In fact safety often *is* costly, both in terms of economics and technical performance, but this is not due to any intrinsic property of safety itself. Rather, the reason safety costs so much is that it is often considered only after the major architectural tradeoffs and design decisions have been made. Once the basic design is finalized, the only choice is to add expensive redundancy or excessive design margins [Leveson, 2009].

It has been estimated in the defense community that 70-80% of the decisions affecting safety are made in the early concept development stages of a project [Frola and Miller, 1984]. As Figure 1 illustrates, compensating later for making poor choices at the beginning can be very costly. Stieglitz' quote is appropriate for all complex systems, not merely airplane design. Safety must be designed and built into systems from the very beginning of concept development.

Unfortunately, traditional tools used for analyzing and improving safety are only applicable in the later stages of system development, when detailed design information is available. These same tools were developed long ago, when the primary cause of accidents was due to mechanical failure [Vesely et al., 1981]. Modern systems exhibit hazardous behavior due to a series of factors that extend well beyond hardware failure. The introduction of new technology, such as computers and software, is changing the types of accidents we see today [Leveson, 1995, 2012].

Hazardous behavior arises in systems due to unsafe interactions between components, even when the components have not necessarily failed. Given the complexity of today's systems, these interactions are increasingly difficult to understand and predict. The underlying assumptions of traditional hazard analysis tools also oversimplify the role of human operators [Dekker, 2005; Rasmussen, 1997; Woods et al., 2010] and software requirements errors [Leveson, 2009; Lutz and Carmen Mikulski, 2003]. Not only

are traditional hazard analysis techniques incapable of analyzing systems that are immature in terms of design detail, they are also very limited with respect to these new accident causation factors, which will become increasingly prevalent in tomorrow's systems.

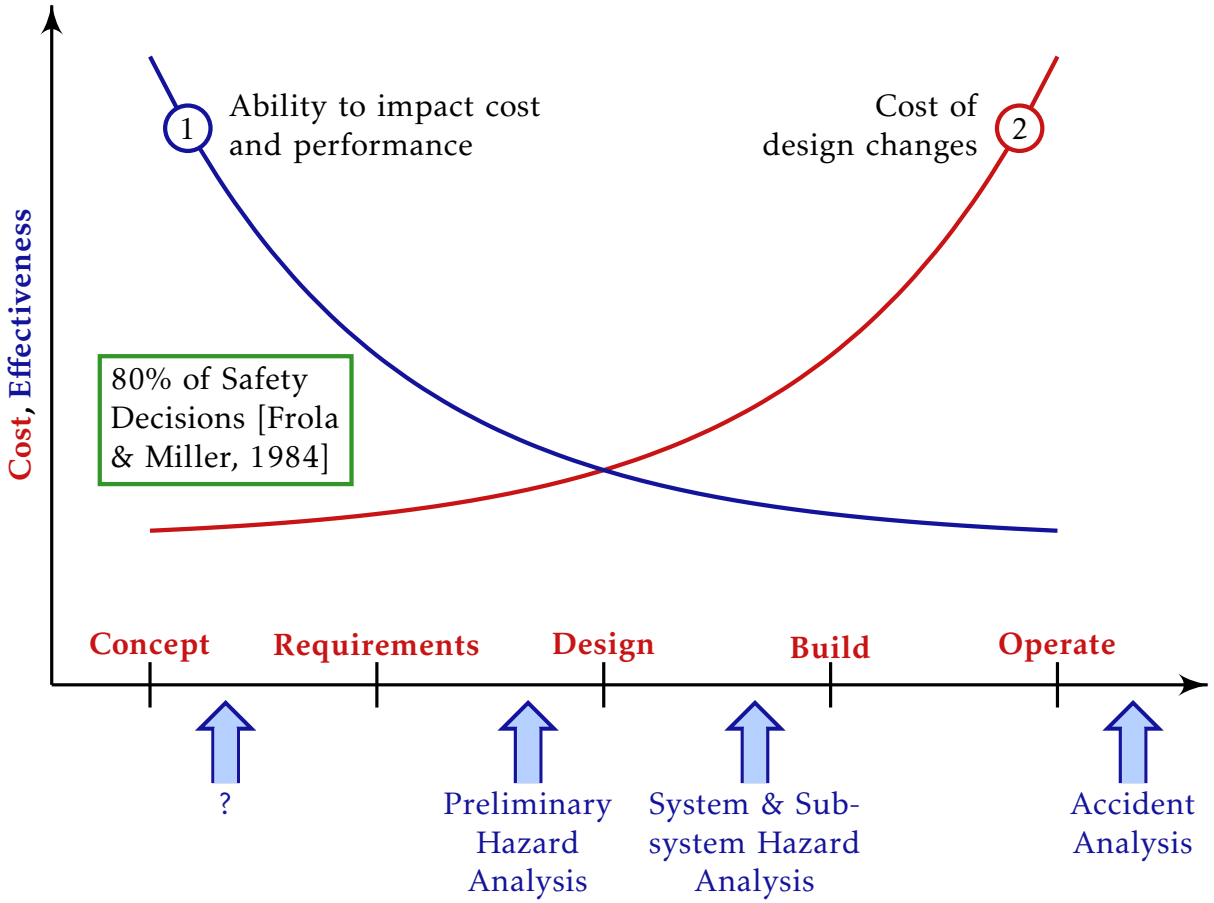


Figure 1: Decision Effectiveness during Life Cycle (adapted from [Strafaci, 2008])

1.1 Motivation

The current[1] national airspace (NAS) system in the United States has achieved historically low accident rates, with an exponential decrease in major[2] accidents since 1960 [Boeing, 2009]. However, the forecasted growth in passenger and freight flights is expected to be more than 5% in the coming decades [Netjasov and Janic, 2008], and the current air traffic management system cannot sustain this growth. In addition to increasing capacity demands, the United States national airspace faces increasing pressure to reduce greenhouse gas emissions and operator costs.

[1] "Current" refers here to the system as it exists (existed) before NextGen implementations.
[2] Major accidents, per the Boeing study, include those with fatalities and/or hull loss.

To meet these challenges, the Federal Aviation Administration (FAA) has developed a program called NextGen, which "integrates new and existing technologies, policies and procedures to reduce delays, save fuel and lower aircraft exhaust emissions to deliver a better travel experience" [FAA, 2013]. The proposed changes must also maintain or improve the FAA's stated top priority, ensuring safe skies and airfields. The European counterpart to the FAA, EUROCONTROL, faces similar challenges and has an analogous program to NextGen called Single European Sky ATM Research (SESAR) [Patteau, 2009].

NextGen calls for increased focus on more efficient flight paths, a shift in responsibility from ground-based crews to flight crews and their flight deck-based decision support tools, the use of trajectory-based operations instead of clearance-based maneuvering of aircraft, and many other changes. In short, NextGen will result in increased reliance on automation, greater coupling between ground and aircraft technology, a major shift in the way airspace information is gathered and disseminated, and a total revamping of how aircraft paths are managed. All of these changes will come as the result of incremental upgrades that span years and even decades.

The FAA and other institutions involved in NextGen recognize that these changes pose a risk to safety-related properties of the current national airspace. Fortunately, these institutions have also recognized the importance of understanding safety-related risk early in development of NextGen improvements as well as including safety during concept development and design selection activities [FAA, 2012; JPDO, 2012]. Unfortunately, however, many of the existing techniques are limited with respect to these goals [Harkleroad et al., 2013].

While the FAA and EUROCONTROL are faced with new issues in air traffic management, aircraft manufacturers have their own challenges. Aircraft manufacturers deal with customers who increasingly weigh cost factors as well as improvements in performance, comfort, and environmental effects [Croft, 2005]. As a result, Boeing developed the B787 and B737-MAX, Airbus developed the A380 and A350, and smaller companies like Embraer and Bombardier also continue to innovate. Boeing selected lithium cobalt oxide (LiCo) batteries for its B787 power system in order to save weight while meeting power requirements. Multiple incidents with these batteries caused the entire B787 fleet to be grounded [Brown, 2013]. The problem was extremely difficult to fix due to constraints on space and weight, taking months of design work and thousands of hours of testing to complete [Yeo, 2013].

Airbus has had its own problems, as Qantas grounded its entire A380 fleet after an uncontained engine failure [ATSB, 2013]. One suggestion was to reduce the allowed thrust on the engines, but this was deemed unprofitable [Yeo, 2010]. Ultimately the engines had to be replaced [Kollewe and Gabbatt, 2010].

The issues with the B787 and A380 highlight the difficulty of identifying and mitigating hazards in complex systems with highly coupled sub-systems and components. The issues also show how difficult, expensive, and sub-optimal it is to modify an *existing* design in order to mitigate safety-related issues.

Other aerospace domains face similar challenges. The United States ponders returning to the moon and sending astronauts to Mars while the European Union, Japan, China, India, and Russia attempt to advance their human spaceflight programs [Drake et al., 2010; Irvine, 2007]. As the United States National Aeronautical and Space Administration (NASA) learned throughout its space shuttle program, early architectural decisions have significant and sometimes irreversible implications for safety. As just one example, all previous US manned space vehicles had launch escape systems, and such a system was discussed for the shuttle program. However, NASA did not implement a launch escape system for programmatic and technical reasons [McCurdy, 1993]. After the Challenger disaster, the shuttle orbiters were fitted to allow for crew evacuation, though this could only be used when the shuttle was in a controlled glide. The crew would have had to reach the exit from their seats and jump out, and this solution is irrelevant for launch events [Petty, 2002; Dumoulin, 1988]. Like the Boeing and Airbus problems, the shuttle program illustrates the importance of safety considerations during early architectural decisions and the diminishing returns after design decisions have been made.

This thesis is not intended to be a critique of recent aircraft designs or of the shuttle program. However, the problems Airbus, Boeing, and NASA have experienced highlight the fact that early design decisions have significant implications for safety and can have major cost impacts later in the program (Figure 1). The importance of integrating safety analysis into early systems engineering activities cannot be overemphasized.

1.2 Research Objectives

Current preliminary hazard analysis and risk assessment techniques are limited with respect to the kinds of scenarios they identify and how risk is communicated to

decision makers. Instead of using traditional failure-based approaches to analyze a concept, this thesis uses a different approach. The proposed approach to analyzing and developing a concept is based on control- and systems-theory that identifies more hazardous scenarios and assists stakeholders in the development and refinement of a concept. By identifying more hazardous scenarios with limited design information, decision makers can eliminate or mitigate hazards by the selection of appropriate architectural options when the cost of doing so is much less than when a design is nearly complete.

Therefore, this research has two main objectives. The first objective is to develop rigorous, systematic tools for the analysis of future concepts in order to identify hazardous scenarios and undocumented assumptions. Related to this theme is the problem of assessing safety-related risk when little design detail is available, with the goal of assisting concept development and design when modifications are most effective. The second objective is to extend these tools to assist stakeholders in the development of concepts using a safety-driven approach. Ideally, this safety-guided concept development would supplement existing system engineering activities, including architectural and design studies that occur during tradespace exploration. Both objectives especially apply to systems where the tradespace includes human operation, automation or decision support tools, and the coordination of decision making agents.

[Page intentionally left blank]

Chapter 2

Literature Review

> *The earlier you find a problem in a...project, the better off you are. The cost to fix an error found during requirements or early design phases is orders of magnitudes less to correct than the same error found during testing.* [Clements et al., 2003]

Much of the literature regarding safety during early system engineering activities focuses on the development of preliminary hazard lists, preliminary hazard analyses, and the assessment of those results. Safety is rarely included explicitly or analytically in traditional early system engineering activities such as concept generation, concept selection, and tradespace exploration.

Preliminary hazard analysis (PHA) is one of the earliest safety-related activities, and many of these analyses not only attempt to assess risk but also suggest potential mitigations or design modifications. However, as section 2.2 describes, PHA is still limited with respect to influencing design and ensuring that safety-related properties are integrated into the system as early as possible. Recall from Chapter 1 the importance of considering safety in the design process, when design modifications have minimal cost and maximum influence.

In order to define what is meant by "preliminary" and to assess how safety is currently considered during preliminary phases, Section 2.1 first provides a brief review of general system engineering activities that occur before detailed design begins. Then Section 2.2 focuses on the state of the art of preliminary hazard analysis, safety risk assessment, and their limitations. Section 2.3 concludes with a review of existing hazard analysis techniques that may be used to improve preliminary hazard analysis.

2.1 Systems Engineering and Concept Development

It is beneficial to review define general system engineering activities and review if (and how) safety is integrated into these activities. The earliest phases of system development are particularly important for this review.

Systems engineering has its roots in systems theory and as a scientific discipline extends back to the 1950s and perhaps earlier [Booton and Ramo, 1984]. Systems

engineering places emphasis "on defining goals and relating system performance to these goals...on decision criteria, on developing alternatives, on modeling systems for analysis, and on controlling implementation and operation" [Miles Jr, 1973]. Importantly, systems engineering focuses on the design of the *whole*, rather than merely the design of the *parts*. Therefore, systems engineering activities should focus on identifying, analyzing, and controlling the interactions among components, in addition to developing those individual components. Notably, this thesis also relies on systems theory, as opposed to the reductionist approach of many traditional safety methods. Chapter 3 defines and describes several important concepts in systems theory, including emergence, hierarchy, control, and communication.

System engineering activities that occur before detailed design is performed include stakeholder analysis, system requirements definition, concept generation, architecture development, and tradespace exploration. These activities are called by many names, but they generally occur during the "concept" or "study" phase [Arnold, 2002; Kapurch, 2010; Haskins and Forsberg, 2011], which is followed by the design phase and ultimately system verification. Figure 2 depicts the system engineering activities and the relationships between these activities in what is referred to as the Vee Model. While safety is almost always a priority for stakeholders involved in devel-

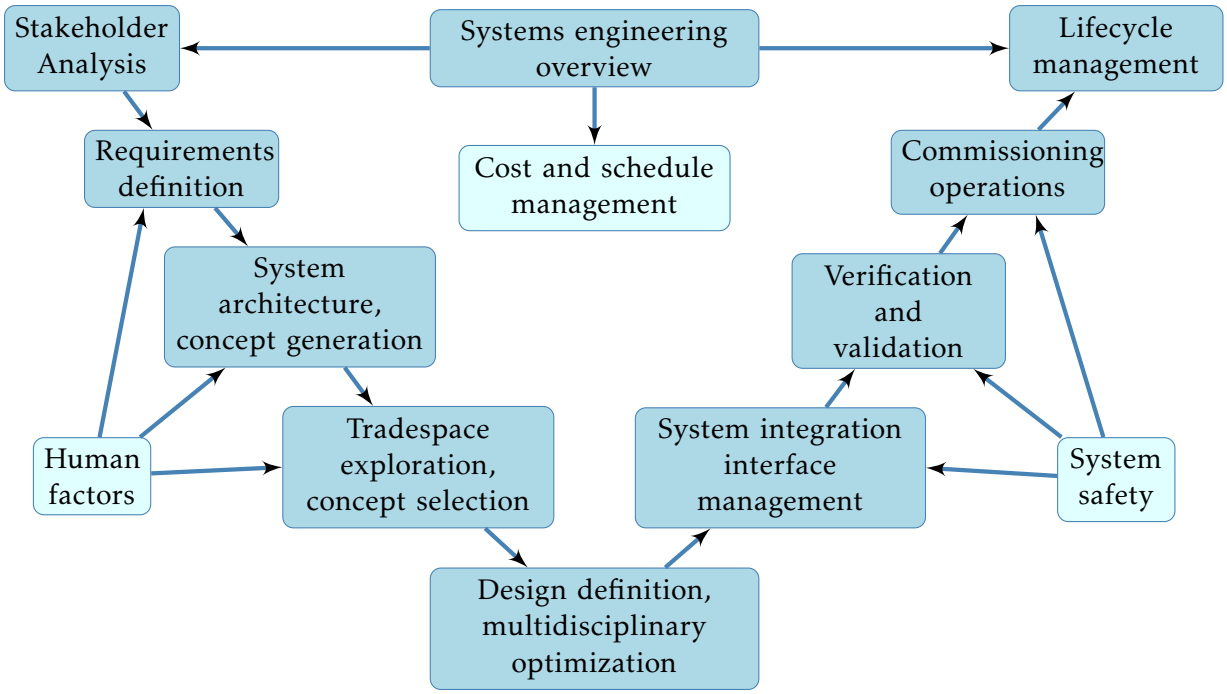


Figure 2: System Engineering Vee Model [de Weck, 2009]

oping a system with any societal impact, safety has not explicitly been considered in

many of these early phase system engineering activities [Crawley et al., 2004].

Most of the effort in proceeding from a set of stakeholder needs to a design that can be implemented involves the selection of *how* to meet those requirements. Stakeholders trade off different properties or design decisions in order to obtain a system that meets requirements and satisfies certain performance criteria. Common system properties used in tradespace exploration of aerospace systems include mass, range, fuel/propulsive efficiency, speed, mission life, and others [Ross and Hastings, 2006; O'Neill et al., 2011]. Safety properties are not included in any of these trade analyses.

Because preliminary hazard analyses (see Section 2.2.1) are not conducted until preliminary design, this portion of the literature review describes three aspects of system engineering that occur earlier in system development—developing a concept of operations, systems architecting, and developing requirements.

A Concept of Operations (ConOps) can be developed in many different ways, but usually share the same properties. In general, a ConOps will include a statement of the goals and objectives of the system; strategies, tactics, policies, and constraints affecting the system; organizations, activities, and interactions among participants and operators; and operational processes for fielding the system [McGregor, 2005]. A ConOps

> *describes how the system will be operated during the life-cycle phases to meet stakeholder expectations. It describes the system characteristics from an operational perspective and helps facilitate an understanding of the system goals* [Kapurch, 2010].

Systems architecting is

> *the process by which standards, protocols, rules, system structures, and interfaces are created in order to achieve the requirements of a system; trade-off studies may precede the determination of system requirements* [de Weck et al., 2011].

A system architecture is

> *an abstract description of the entities of a system and the relationships between those entities. Architecture is important in most technical fields, including not only civil architecture of buildings but of physical products, software, computer networks, large engineering systems, and infrastructures* [Crawley et al., 2004].

A slightly different definition from software engineering states that an architecture is

> *the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them* [Bass et al., 1998].

The above definitions of system architecture notably omit the role of human operators. A more appropriate definition for this thesis might be: system architecture—a functional description of the entities of a system and their relationships, including physical products, software, and human operators. Chapter 3 introduces and defines a *control structure*, which defines the part of the architecture in systems-theoretic terms and will be used throughout this thesis.

In addition to developing a concept of operations and system architecture, early systems engineering activities should result in defining technical requirements. Requirements are

> *statements of what the system must do, how it must behave, the properties it must exhibit, the qualities it must possess, and the constraints that the system and its development must satisfy* [Northrop et al., 2007].

Requirements should ensure that the system achieves stakeholder objectives and satisfies formally imposed standards, regulations, or contracts [Radatz et al., 1990].

Concept generation, architecting, and early requirements generation do not explicitly include safety even though it would be highly beneficial during this part of system development [Frola and Miller, 1984]. This exclusion of safety-related activities is perhaps appropriate given the current state of the art. Though some of the concept generation, system architecting, and requirements identification frameworks have become increasingly formalized, the techniques still do not yield the level of detail necessary to perform most traditional types of hazard analysis [Harkleroad et al., 2013].

The limitations for safety-related analysis during preliminary system activities are mostly due to limitations of the hazard analysis methods themselves, not due to limitations of general systems engineering activities. There is nothing inherent in many preliminary system engineering activities that prevents the inclusion of hazard analysis, except that detailed design information is necessarily limited during these activities.

2.2 Safety Activities during Preliminary Design

Traditionally, safety-related activities conducted during the preliminary phases of an engineering program include developing Preliminary Hazard Lists (PHL), performing Preliminary Hazard Analysis (PHA), and informing decision-makers by using risk assessment techniques. The primary objective of a preliminary hazard list is to identify very high level hazards during concept development, while the objective of preliminary hazard analysis is to categorize the risk level (hazard level) of the identified hazards.

Comparisons of system architectures and design alternatives are based on trade studies that incorporate performance objectives such as (for aerospace systems) mass, speed, range, and efficiency, as well as cost estimates. While PHA efforts typically begin before an architecture is selected, these safety efforts are performed in parallel with architecture studies and therefore have little impact the general systems engineering process. That is, the PHA is totally independent of the architectural study. In addition, the focus on component failure inherent in current PHA techniques severely restricts the ability to identify hazard causality due to human behavior, software requirements flaws, and the interaction between human operators and software.

Preliminary hazard analysis is a guided effort for identifying hazards, their associated causal factors, effects, and level of risk. Mitigating measures are also sometimes included [Ericson, 2005]. Currently and in the past, PHA has focused on failure modes of sub-systems or components, but such a focus is limited for several reasons.

First, there are many causes of hazardous behavior other than component failures or faults, which is explored further in the next section (Section 2.3). Second, PHA starts during concept formation when little design detail is available. This lack of design detail, combined with the novelty of many systems being developed today, makes it infeasible to assign a likelihood and/or consequence to a component failure. Before describing the preceding limitations in greater detail, current PHA guidance and their underlying assumptions must be discussed.

2.2.1 Preliminary Hazard Analysis

Standard preliminary hazard analyses include a list of hazards to be avoided, potential causes of those hazards, effects on the system, severity level of the hazards, and supporting comments or recommendations [Vincoli, 2005]. Table 1 shows a generic PHA table and expected contents. Most of the guidance for conducting a PHA comes

from military or regulatory bodies, although there are notable exceptions in the academic and practitioner literature. This sub-section explores both government and academic sources.

Table 1: Sample Preliminary Hazard Analysis (PHA) Worksheet, adapted from [Vincoli, 2005]

PRELIMINARY HAZARD ANALYSIS PROGRAM: _____ DATE: _____ ENGINEER: _____ PAGE: _____						
ITEM	HAZARDOUS CONDITION	CAUSE	EFFECTS	RAC	ASSESSMENTS	RECOMMENDATIONS
Assigned number sequence	List the nature of the condition	Describe what is causing the stated condition to exist	If allowed to go uncorrected, what will be the effect or effects of the hazardous condition	Hazard Level assignment	Probability, possibility of occurrence: -Likelihood -Exposure -Magnitude	Recommended actions to eliminate or control the hazard

Military and aerospace standards provide further guidance on expected inputs and outputs for a preliminary hazard analysis. The United States military standard for system safety includes a relatively systematic procedure for identifying hazards, classifying mitigation measures, and documenting all of the above. The U.S. military specifies that the PHA shall identify hazards by considering the potential contribution to mishaps from: system components, energy sources, hazardous materials, interfaces and controls, software, human factors engineering and human error analysis, and several other factors [US DoD, 2012]. Other military standards provide similar guidance for conducting a preliminary hazard analysis, see for example [UK MoD, 1996].

The United States Military Standard 882E [US DoD, 2012] additionally requires that a risk assessment be performed for all the factors identified in the PHA. Subsection 2.2.2 discusses general aspects of risk assessment during preliminary phases of system development and includes a description of military standards.

The United States Federal Aviation Administration prescribes a five-step process for performing a risk assessment and recommends several analytical processes or techniques for identifying hazards and causes [FAA, 2008]. Operational Hazard Assessment (OHA) and Comparative Safety Assessment (CSA) are touted as the most relevant tools for hazard analysis during concept development or preliminary design.

Both OHA and CSA share characteristics with efforts labeled *PHA* throughout this sub-section.

OHA contains a list of operational hazards[1], their effects, and severity classifications for each item. Like the other PHA outputs described in this sub-section, CSA includes a severity and likelihood classification for each hazard, cause, and effect. CSA is different from many of the other references in that it is intended for use when design or operational changes are proposed for the national airspace. Therefore, the results of CSA are meant to be compared with some baseline system.

Roland and Moriarty [2009] suggest a slightly different set of activities that should be included in PHA. While the artifacts of the analysis remain the same (hazards, causes, likelihood, consequence), Roland and Moriarty suggest specific activities and aspects of the system that should be scrutinized. The PHA should consist of a review of historical safety experience in similar systems; an examination of basic energy sources; an examination of safety-related interfaces; exposure to environmental hazards such as shock, vibration, extreme temperatures; an examination of software modules in their interfaces with hardware, operators, or other software for possible hazards; and safety related equipment including interlocks, redundancy, and fail-safe designs.

Ericson [2005] describes the necessary inputs, desired outputs, and expected process of a PHA. To perform a PHA, the analyst must have (1) a preliminary hazard list, (2) design knowledge, and (3) hazard knowledge. Preliminary hazard lists are developed even earlier in a project's development than a PHA and are typically the first safety-related assessment of a project. To possess design knowledge, Ericson emphasizes the importance of having a list of system *components* and their intended function. Hazard knowledge is derived from hazard checklists and consists of a basic knowledge of hazards, their sources and components (element, initiating mechanism, and target), and hazards in similar or heritage systems. According to Ericson, hazard checklists should include :

1. Energy sources
2. Hazardous functions
3. Hazardous operations
4. Hazardous components
5. Hazardous materials
6. Lessons learned from similar type systems

[1] Operational Hazards in the FAA's OHA framework are equivalent to hazard *causes* in the other PHA references.

7. Undesired mishaps
8. Failure mode and failure state considerations

Figure 3 depicts the mapping of top-level mishaps (TLM) to system safety requirements (SSR) and safety critical functions (SCF) via a decomposition of top-level mishaps into lower-level failure modes. In addition to outlining what should be expected in terms of inputs, outputs, and process, Ericson provides several real-world PHA examples as well as a list of common mistakes to avoid.

Although Ericson provides more comprehensive guidance and documentation than the other literature reviewed here, the underlying theory and expected outputs of all of the preceding literature are mutually consistent. PHA is intended to generate a list of hazard causes arising from faulted or failed components, assign a probability of occurrence along with severity of consequence, and then (combined with potential mitigation measures) assess the relative risk of each cause.

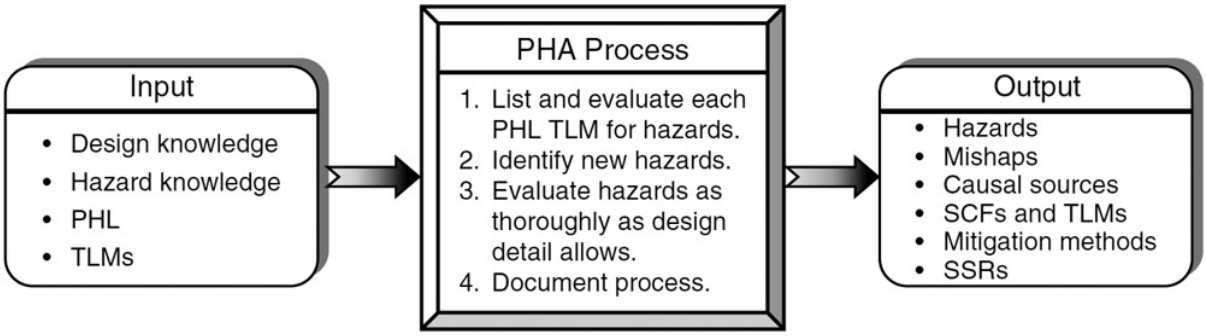


Figure 3: PHA Inputs, Process, and Outputs [Ericson, 2005]

For illustrative purposes consider a recent example of PHA from the aerospace domain. Table 2 on the following page includes a subset of the PHA results for trajectory-based operations (TBO), a proposed upgrade in the NextGen program. Observe the emphasis on component failure, likelihood of occurrence, and significance of failure in Table 2. The emphasis on component failures in this analysis is linked to an accident causality model, although this link is not explicitly stated in any of the literature on PHA. The next section describes and compares accident causality models.

In summary, PHA has historically been a framework for identifying hazard causes in terms of component and sub-system failures or faults. In addition to their effects on system hazards, these causes are further identified in terms of their severity of consequence and probability of occurrence. The results of PHA are used to assess risk, the primary form of which is called a Risk Assessment Matrix or Risk Assessment

Table 2: PHA for Trajectory-Based Operations, adapted from [JPDO, 2012]

Hazard ID	Hazard Name	Hazard Description	Causes	Significance	Likelihood	Assumed Mitigations	Strength of Mitigations	Outcome Risk	Justification
TBO-0004	ADS-B Ground System Comm Failure	GBA does not receive ADS-B message	Receiver failure	High	Low	Redundant equipment; SSR; Primary Radar; Overlapping ADS-B coverage; Multi-Lat; Design and Equipment Certification Requirements	Medium	Medium	Strength of Mitigations depends on the type of backup; Multi-lat should be used if spacing requirements are tighter than they are today. ...
TBO-0021	GBA fails to recognize dynamic situation and is unable to find a solution	The software lacks robustness in its implementation that leads to inability to find a solution	Design flaw, coding error, insufficient software testing, software OS problem	High	Med	Comprehensive system testing before certification and operational approval. TCAS; See and avoid. Pilot could recognize in some cases; Controller could recognize in some cases	Low / Medium	Medium / High	Anything that is complex can lead to this situation

Code.

2.2.2 Use of PHA in Risk Assessment

PHA is not merely intended to generate a list of hazard causes. Rather, it is intended to guide decision makers and system designers by assessing the highest areas of safety-related risk and thus influence how resources are (or are not) allocated. A risk assessment matrix is often used to identify and compare risk, and risk assessment matrices are required by certain military and government programs [US DoD, 2012; Kapurch, 2010; UK MoD, 1996; FAA, 2008].

For the purposes of assessing safety-related risk, risk assessment matrices contain the hazard causes identified in the PHA, with each cause categorized by its hazard level assignment[2] and probability[3]. Observe the areas of High, Serious, Medium, Low, and Eliminated risk in Table 3. A hazard cause with "Catastrophic" or "Critical" consequences, combined with a "Frequent" or "Probable" probability is deemed high risk. Alternatively, causes with "Negligible" severity and only "Occasional", "Remote", or "Improbable" consequences can be considered low risk. Each cell is labeled according to its location in the matrix, with entries such as "High", "Serious", "Medium", or "Low". The assignment in each cell is referred to as the Risk Assessment Code (RAC) and will sometimes be differentiated by numbers instead of words. In theory, by organizing hazard causes in this way, system developers are able to effectively focus resources and effort in order to improve the safety-related properties of the system.

Roland and Moriarty [2009] provide a more theoretical approach to modeling safety-related risk, which takes the form,

$$P(Ct_n) = \sum^{i}\sum^{j}\sum^{k} P(I_i)\,P\left(C_j/I_i\right)P\left(L_k/C_j\right)P(Ct_n/L_k) \tag{1}$$

[2] In the literature, "hazard level assignment" is also called "severity", "consequence", or some combination therein.

[3] Probability is used interchangeably with "likelihood" in the PHA literature.

Table 3: Risk Assessment Matrix [US DoD, 2012]

RISK ASSESSMENT MATRIX				
SEVERITY / PROBABILITY	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occasional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

where

$$
\begin{aligned}
n &= \text{Loss type} \\
P(Ct_n) &= \text{probability of cost, } Ct_n, \text{ per exposure unit} \\
P(I_i) &= \text{probability of initiating event, } I_i \\
P\left(C_j/I_i\right) &= \text{conditional probability of consequence, } C_j, \text{ given } I_i \\
P\left(L_k/C_j\right) &= \text{conditional probability of loss, } L_k, \text{ given } C_j \\
P(Ct_n/L_k) &= \text{conditional probability of cost, } C_t, \text{ given } L_k.
\end{aligned}
$$

The expected loss per exposure unit is computed by:

$$E(Ct_e) = \sum^{n} P(Ct_n)\, Ct_n, \tag{2}$$

and total risk is determined by evaluating the expected cost of the loss, $E(Ct_e)$ over all

exposure units.

$$\text{Risk} = \sum^{e} E\left(Ct_e\right) \tag{3}$$

where e = exposure units.

Though the formulation of Roland and Moriarty is more formalized relative to the classic risk assessment matrix approach, the underlying assumptions are much the same, with a subtle difference. Although using different nomenclature, both Roland and Moriarty and the risk assessment matrix framework use probability of failure condition along with severity of consequence to "measure" risk. Note that the probability and severity axes of a risk assessment matrix are *orthogonal*, implying that risk assessment matrices assume independence between probability and consequence. In equations 1 through 3, Roland and Moriarty assume some degree of coupling between probability of cost (which is analogous to severity of consequence) and probability of loss (which is analogous to the general likelihood/probability metric in RAC).

Modarres [2006] provides a survey of techniques for ranking the relative risk in a system, including the Birnbaum importance [Birnbaum, 1968], Fussell-Vesely [Fussell, 1975; Vesely et al., 1983], Risk Reduction Worth (RRW), Risk Achievement Worth (RAW) [Chadwell and Leverenz, 1999], and Differential Importance Measure (DIM). The formalism of probabilistic risk assessment (PRA) underlies these measures. That is, each metric involves ranking the probability of a scenario relative to the probability of all the identified scenarios. The metrics differ in how individual scenarios are weighted but are grounded in the following assumptions: (1) hazards are caused by a linear chain of events and (2) the probabilities of each event can be calculated. Each of these metrics is best suited for completed designs and still suffer the limitations below.

The hazard analysis and risk assessment approaches described thus far are intended for sub-system and component faults or failures. However, the approaches have been applied to systems with human operators and software controllers, in addition to mechanical electro-mechanical hardware. Estimates for probability of occurrence have been used for software failure[4] and human error [RTCA, 2011; JPDO, 2012].

At later stages of system development, when more design detail becomes available, there are several statistically-driven approaches for modeling risk. Many of these ap-

[4] Software does not "fail" in the same way that a mechanical component fails. This is one limitation of the existing PHA approaches, which is discussed in the following Sub-section.

proaches use Bayesian statistical models and Monte Carlo simulations to approximate relative risk given certain initial conditions, or changes in risk due to the introduction of safety-related features. For example, Kochenderfer et al. [2010] use Bayesian networks and fast-time simulation to estimate the relative decrease in risk of near midair collision due to the introduction of collision avoidance algorithms on unmanned aircraft. Kuchar and Drumm [2007] use fault tree analysis to identify failure scenarios and then fast-time (Monte Carlo) simulation to assess sensitivity to changes in collision avoidance logic.

2.2.3 Limitations in Current Approaches to Preliminary Hazard Analysis

Many accidents in modern, software-intensive systems involve the interaction of components, the incorrect specification of software requirements, and unsafe human behavior due to a myriad of factors including human errors caused by confusion resulting from the automation design [Leveson, 2012]. Safety is an emergent property; that is, safety (or lack thereof) only emerges from the interactions of a system's components. The next section describes emergence in greater detail.

Much of the guidance for identifying hazard causes during PHA—for example documenting energy sources, hazardous materials, or faulted (failed) modes of mechanical components—is a necessary but insufficient aspect of hazard analysis. The underlying model of accident causation in the PHA literature is reductionist and assumes that if all component failure modes of a system have been identified, then so have all potential sources of hazardous behavior. Current PHA literature provides little to no guidance for how to identify hazardous interactions amongst components; incorrectly specified software requirements; or human operator errors due to poor design of procedures, computer interfaces, and underlying logic of automation and decision support tools. One *could* list "component interaction error", "software design error", or "operator mode confusion" in a hazard list like Table 1 on page 28. However, there is no guidance on how to identify *why* these factors might occur. That is, simply listing "software error" without further explanation does not add any useful information to the engineering process.

Consider again the example from Ericson [2005]. Table 4 on page 36 lists the potential failure modes for the missile rocket booster subsystem of a fictitious missile system. PHA-4 lists "erroneous initiate commands", with software faults and human error as potential causes. The recommended action is then to use multiple switches or conduct a fault tree analysis of the fuze design. Though the example PHA correctly

identifies software and human operators as potential sources of hazardous behavior, the method is limited in helping the analyst to reason about specific causal factors[5]. That is, *all* hazards are caused by hardware, software, or humans. Simply listing a generic set of factors is not very helpful, and PHA techniques suffer from a lack of guidance in identifying causal factors that lead to *specific* hazardous states that stakeholders wish to avoid.

Table 4: ACE Missile Example (adapted from [Ericson, 2005])

Subsystem: Missile Warhead Subsystem				
No.	Hazard	Causes	Effects	Recommended Action
PHA-4	Inadvertent W/H explosives initiation due to erroneous initiate commands	Erroneous commands from hardware faults; software faults; human error	Personnel death/injury	Use multiple independent switches in fuze design Conduct FTA of fuze design
PHA-5	Inadvertent W/H explosives initiation due to external environment	Bulletstrike, shrapnel, heat	Personnel death/injury	Use insensitive munitions Provide protective covering when possible
PHA-6	Failure of W/H explosives to initiate when commanded	Hardware faults; software faults	Dud missile; **not a safety concern**	

Finally, PHA requires the analysts to assign a probability of occurrence, which could be either a probability of a faulted mode, the probability of its affect on the system, or both. There are several problems with this approach. First, PHA occurs near the beginning of a project when very little design detail is available. Often the analyses rely on heritage data to extrapolate a probability for identical or similar components. However, in the case of software and human operators, heritage data does not, and cannot, exist. Even when software modules are reused, either the context in which they will operate or some details of the code will be modified. Of equal importance, human and software behavior is not stochastic.

Software behaves exactly as it is specified and coded [Leveson, 1995], and human behavior is highly dependent on context, both in the system design and due to en-

[5] This is not a critique the specific example in [Ericson, 2005] but is a critique of PHA methods in general.

vironmental factors [Dekker, 2005; Leveson, 2012; Klien et al., 2004; Reason, 2000; Vicente, 1999]. Even in later stages of system development, or during operations, software error probabilities cannot be calculated and human error probabilities can only be attained with limited applicability.

In summary, the types of causes identified by PHA techniques is limited to electromechanical faults or very generic causes related to human or software behavior. These generic types of causes are not particularly useful for guiding the design. In addition, when PHA requires probability estimates, the probabilities of many sources of hazardous behavior are difficult or impossible to validate. The probabilities are either highly uncertain (due to limited design detail at the beginning of a program), unknowable (due to human behavior in a complex environment), or irrelevant (because software behavior is strictly deterministic).

Limitations of Current Risk Assessment Methods

Risk assessment methods used in preliminary design are limited with respect to the input data, the internal mathematical foundation of the methods, and the interpretation of the results. The previous sub-section provided an argument about the limitations inherent in the data being input into risk assessment methods, which is the list of hazard causes and associated probabilities and recommendations. Chapter 3 proposes a different analysis technique that can identify more types of causes with limited design detail.

Proponent of existing approaches insist that the probabilities are merely "likelihoods" and are only meant to guide decision making, as opposed to representing a rigorous mathematical formulation. That is, the probabilities are intended to be qualitative in nature and should be treated as such. Ignoring arguments about the validity of probabilities (and ignoring the limitations with respect to comprehensiveness described above), there are other issues in the approaches used to assess risk during preliminary hazard analysis.

Consider a hypothetical case where the list of hazard causes is "complete", the associated probabilities are known with certainty, and the probabilities can be validated. For illustrative purposes, this hypothetical case has three hazard causes and the risk assessment approach uses a 5x4 risk assessment matrix prescribed in U.S. military programs (Table 5).

Table 5: Hypothetical Risk Assessment Matrix, 5x4

Probability	Severity	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
	Frequent (A)	High	High	**Cause 1** Serious	Medium
	Probable (B)	High	High	Serious	Medium
	Occasional (C)	High	**Cause 2** Serious	Medium	Low
	Remote (D)	**Cause 3** Serious	Medium	Medium	Low
	Improbable (E)	Medium	Medium	Medium	Low

When the axes are scaled linearly on the interval $[0,1]$, the risks for each cause are calculated as follows,

$$\text{Risk}_i = \text{Likelihood}_i \times \text{Severity}_i$$

$$\text{Risk}_1 = 1 \times 0.25 = 0.25$$

$$\text{Risk}_2 = 0.5 \times 0.7 = 0.35$$

$$\text{Risk}_3 = 0.25 \times 1 = 0.25$$

With a logarithmic probability axis on the interval $\left[10^{-10}, 10^{0}\right]$ and a logarithmic severity axis on $\left[10^{8}, 10^{0}\right]$, the risks are

$$\text{Risk}_1 = 10^{0} \times 10^{3} = 10^{3}$$

$$\text{Risk}_2 = 10^{-5} \times 10^{5} = 1$$

$$\text{Risk}_3 = 10^{-7} \times 10^{8} = 10$$

The risks receive the same ranking, "Serious"[6], and yet are both qualitatively and quantitatively different. The first risk will result in some relatively inconsequential loss with certainty, while the third risk has a relatively low chance of resulting in a

[6] According to the matrix prescribed in [US DoD, 2012].

catastrophic loss. The second risk is quantitatively greater than the other two using the linear axes and quantitatively the least when using logarithmic axes. Yet, all three risks are considered to be intermediate. Is a 50% chance of a critical loss actually worse than the certainty of a marginal loss? Or, what about a 25% chance of a catastrophic loss? Cox formalizes this critique, explaining that not only does the risk assessment matrix violate desirable mathematical properties (e.g. weak consistency and translation invariance), it can also provide inconsistent or misleading qualitative information [Cox Jr, 2008]. Roland and Moriarity [2009] employ a more sophisticated approach to combining likelihood and risk, but the approach is still limited by the uncertainty and validity of the probabilistic analysis as well as the following factors.

Not only do risk assessment matrices suffer from both qualitative and formal mathematical flaws, risk assessment matrices also do not explicitly consider the cost or effectiveness of mitigations. The ability to eliminate or effectively mitigate a hazard is a vital aspect of system safety [Dulac and Leveson, 2005], and this is neglected in classic risk assessment matrices. Some PHA tables do include suggested mitigations, however, and many even include the suggested mitigation as part of the overall assessment (ranking) of a particular hazard cause.

Cost estimates should be considered along with effectiveness of mitigation, but the cost estimates suffer the same limitations associated with existing PHA techniques. The focus on failures and faults implies only a certain class of mitigations. Put another way, the emphasis on reliability calls for redundancy, fault tolerance, or large design margins[7]. These kinds of design solutions may be necessary, but they are generally very expensive and are either irrelevant or ineffective for many kinds of problems in software- and human operator-intensive systems [Miller, 1988].

In addition to the inherent limitations described above, risk assessments are often presented (and thus interpreted) inappropriately. Probabilistic assessment has some degree of uncertainty, even when the artifact being assessed exhibits very predictable, stochastic behavior. This uncertainty must be declared, although often it is not [Downer, 2013]. Qualitative assessments of human and software error are perhaps best left out entirely, lest the entire space of the risk assessment matrix be filled with error bars. The statistically driven, simulation-based approaches used to assess relative risk are somewhat effective at providing some quantifiable measure of risk, but these methods are only appropriate when there are relatively high fidelity models

[7] Also called safety factors or factor of safety, which is the strategy of designing a component to withstand its predicted use case plus some margin.

of risk scenarios. Such models are only valid at later stages of systems engineering.

A different approach is needed for including safety in the early phases of systems engineering. A new approach to preliminary hazard analysis should do a better job of identifying potentially hazardous behavior due to the kinds of factors prevalent in modern, socio-technical systems. These factors include hazardous component interaction, inappropriate human behavior, incorrect specification of software requirements, confusion or complacency due to human interaction with automation, as well as hardware faults found in traditional systems. The safety-related activities should also guide in *developing* the system by helping engineers to identify safety-related requirements and to develop an architecture that eliminates or mitigates hazards.

2.3 Accident Causality and Hazard Analysis Techniques

There is an increasing prevalence of software and human-computer interaction, and thus a changing nature of accident causation in modern engineered systems. It is therefore important to review the accident causality models upon which hazard analysis techniques are based. There exist several hazard analysis techniques that have typically been applied once a design is complete but could be applied during preliminary hazard analysis. After a review of accident causality models, this section briefly describes several analytical techniques and their effectiveness during early phases of systems engineering.

2.3.1 Chain-of-Events Accident Models

A major difference between the methods introduced in Section 2.3.2 (and used this thesis) and traditional hazard analysis techniques is the underlying model of accident causality. Most current hazard analysis and safety assessment techniques are based on a chain-of-events model of causality, where the events represent component failures. Each failure event leads to the next one in the chain with a direct relationship between the two. An example chain-of-events model used in aviation is called the Swiss Cheese Model.

The Swiss Cheese model argues that accidents are caused by failures in four stages: organizational influences, unsafe supervision, preconditions for unsafe acts, and unsafe acts [Reason, 1990]. Each stage can be represented by a slice of Swiss cheese and the holes in the cheese represent a failed or absent defense in that layer. Figure 4 depicts this model, which has become increasingly popular in aviation [Nance, 2005].

The model assumes that the holes representing individual weaknesses are randomly varying in location and size. Eventually the holes come into alignment so that a path is possible through all slices, representing a sequence of failures throughout several layers of defense. Failures then propagate along the path through each defense barrier and cause an accident. The Swiss Cheese model not only assumes a linear chain of events structure but also assumes random behavior of components and independence between failures in each layer. Critics argue that these assumptions do not hold in practice, especially for safety-critical software and human behavior [e.g. Hollnagel et al., 2007; Dekker, 2013].

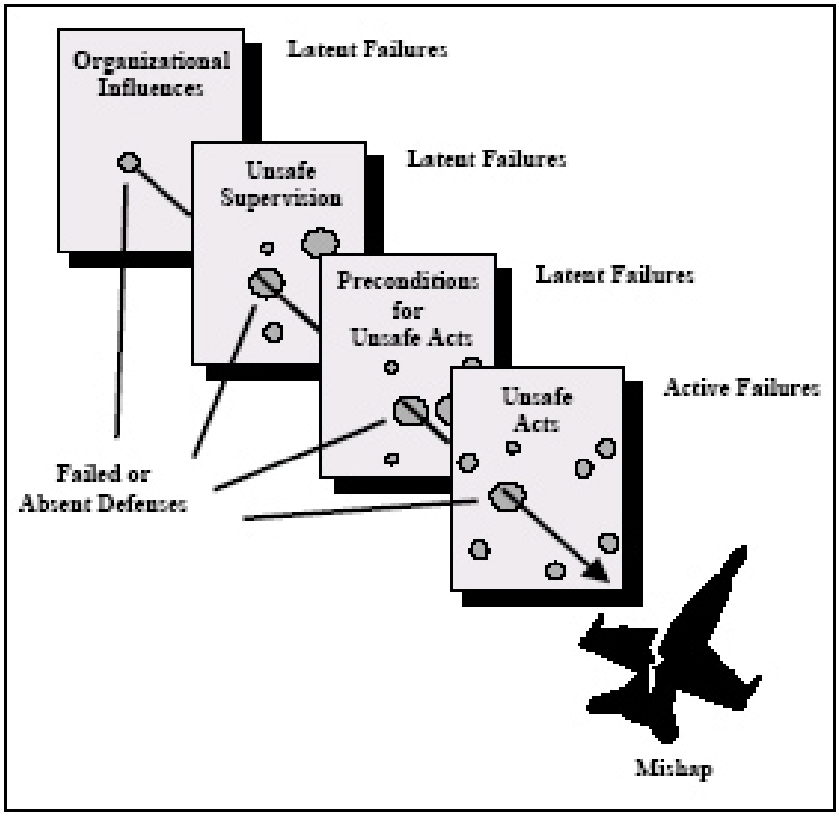


Figure 4: Swiss Cheese Accident Model [Reason, 1990]

Many of the hazard analysis techniques used for NextGen assume a linear, chain-of-events causality model. For example, "each [event sequence diagram] begins with a triggering event (primary system failure) and proceeds through a logic diagram to show how backup systems and procedures are used either in series or parallel to prevent the ultimate bad outcome from occurring." [JPDO, 2012, p.17]

Techniques based on Chain-of-Events Model

Fault Tree Analysis (FTA) was developed in the 1960s at Bell Laboratories [Watson, 1961] and expanded in the 1980s to formalize and standardize the rules of application [Vesely et al., 1981]. FTA is the primary technique for scenario modeling in Probabilistic Risk Assessment (PRA) [Modarres, 2006] and is perhaps the most popular hazard analysis technique. For aeronautics applications, FTA is recommended to be applied in between "Preliminary Design" and "Detailed Design" (as well as later phases) [SAE, 1996]. FTA is most effective when more design detail is available than what is typically available during concept development. It is therefore rarely used during concept development or system architecting.

Failure Modes and Effect Analysis (FMEA) was developed to systemically evaluate the effect of individual component failures on system performance. The United States Department of Defense developed FMEA in 1949 as part of its weapon systems program [USDoD, 1949], and the technique has been applied in many domains including aerospace [SAE, 1996]. Like fault tree analysis, FMEA is based on a chain-of-events model of accident causation. However, unlike FTA, FMEA is an inductive process that starts with a basic component failure or fault, and then the analyst reasons about this failure's effect on the overall system behavior. FMEA may be described as a "bottom up" approach, while FTA is a "top down" approach that begins with system-level events rather than component-level events.

Event Tree Analysis (ETA) was developed in 1974 during a nuclear power plan study [Rasmussen, 1975] and has been adopted in the aerospace domain [RTCA, 2008]. Like FTA and FMEA, ETA is based on the chain of events accident model, and in fact event trees were originally intended to be combined with fault trees as part of an overall Probabilistic Risk Assessment [Leveson, 1995]. The first step in ETA is to identify an initiating failure event such as a structural failure or loss of fuel. The next step is to list, in the anticipated sequence of operation, the set of barriers or protective functions intended to prevent the initiating event from leading to an accident. Last, a logical tree is constructed by tracing forward in time from the initiating event and inserting a binary branch at each barrier to represent the possible success or failure of that barrier.

ETA is similar to FMEA in that it is forward-searching, or "bottom up", and traces forward from a failure condition to a potential hazardous state. ETA is similar to FTA in its logical decomposition of conditions, and event trees are effective for quantitative

analysis if the probabilities of each barrier condition are known. Another similarity between ETA and FTA is that each barrier is assumed to operate independently, which means that the probability of each end state is simply the multiplication of each of the probabilities along the path to its end state.

Limitations of Chain-of-Events Models

Based on some chain-of-events structure such as the Swiss Cheese Model, traditional safety engineering techniques then focus on preventing or reducing the probability of component failure to prevent accidents. FTA and FMEA, at least as commonly used and as used in the TBO safety assessment discussed below, also assume that most of the component failure modes are independent. Human operators and software are treated as if they fail like mechanical hardware, and likelihood of error is assigned to them. Given the critical role that software and human decision making play in modern complex systems, these assumptions are unrealistic.

Accidents often arise due to unanticipated failures or due to unsafe interactions among components that have not failed. Starting a hazard analysis from failures puts the analysis at risk of identifying only a subset of the possible causes, as opposed to beginning with hazards and identifying the interactions that could possibly lead to hazardous states, including those not involving component failure (see Figure 5).

The systems approach used in this thesis recognizes that software does not ''fail,'' but merely performs the way it was designed: it can therefore be hazardous due to flawed requirements (or implementation) or unsafe interactions with the rest of the system. Most software-related accidents arise due to flaws in the software requirements [Leveson, 1995]; therefore, obtaining a complete and correct set of safety-related requirements and constraints on software behavior is key to preventing software-related accidents.

Human operators also do not fail in the sense that hardware does and most of their errors are not random. Instead, humans are influenced by the design and operation of the overall system, as well as the operational context, and can thus make unsafe decisions. These decisions may be due to the factors in Figure 6 on page 48, such as incorrect mental models of the process they are controlling, possibly due to missing or incorrect feedback.

The human error identification process in these traditional methods is incomplete, but the problems involve more than just incompleteness. In methods such as FTA or FMEA, human error is treated in exactly the same way as a physical failure, that is, as

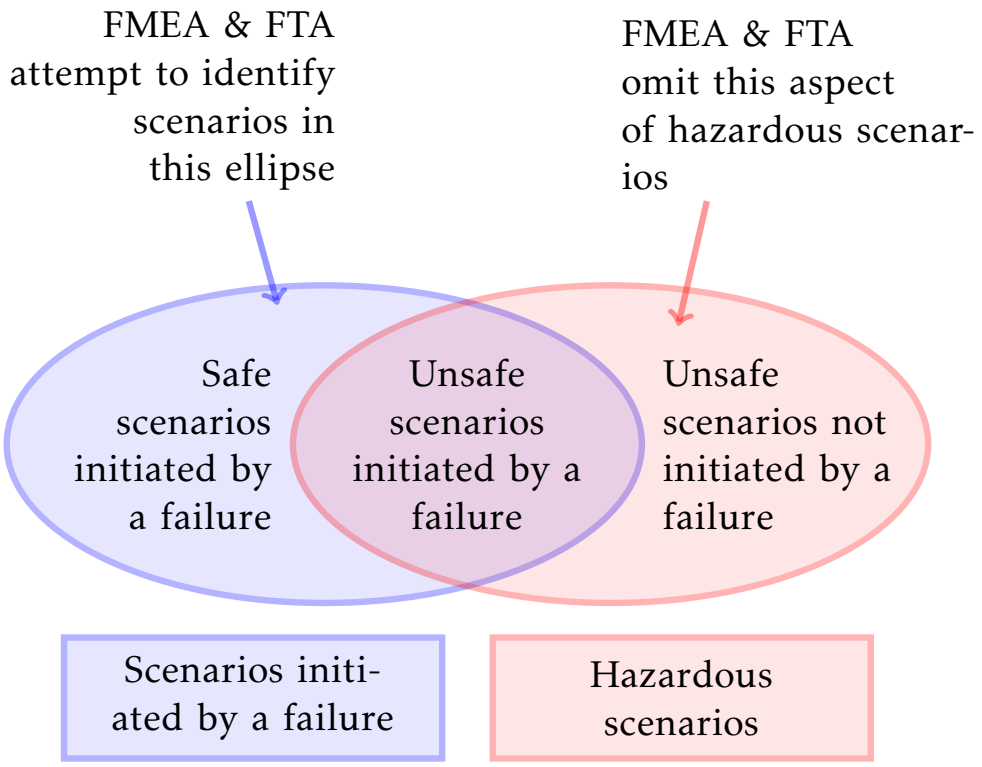


Figure 5: The consequences of equating safety and reliability

a deviation from a predefined behavior or procedure. Unfortunately, this treatment of human error oversimplifies it as a binary decision between right and wrong. Many of the most important situations involved in accidents are overlooked because they are difficult or impossible to model in this way, including when [Fleming et al., 2013]:

- The correct behavior is not predefined or not clear.
- The prescribed behavior is thought to be incorrect by the person responsible for following it.
- Procedures conflict with each other, or it is not clear which procedure applies.
- The person has multiple responsibilities or goals that may conflict.
- The information necessary to carry out a procedure is not available or is incorrect.
- Past experiences and current knowledge conflict with a procedure.
- The procedure is misunderstood or the responsibility for the procedure is unclear.
- The procedure is incorrect.

Although some accidents still occur today due to failures or combinations of failures,

many of them arise due to a complex combination of factors. As such, identifying a failure and its predicted effect is insufficient. Most accidents happen due to the interaction of components and dynamic behavior of the agents—which include human operators and software—and only sometimes is a failure even necessary. Examples of these complex factors abound in aviation, for example Air France 447 [Wise et al., 2011], Asiana 214 [Sherry and Mauro, 2014], and several runway overshoot accidents [e.g. CIAIAC, 2006; Hawkins et al., 2013].

Other Techniques

There are several methods based on Bayesian Belief Networks (BBN) and/or simulation that have been developed to model safety-related risk in the aerospace domain. Examples of BBN- or simulation-based approaches include Causal Models for Air Transport Safety (CATS) [Ale et al., 2008], Aviation Safety Risk Model (ASRM) [Luxhøj and Coit, 2006], Human Factors Analysis and Classification System (HFACS) [Wiegmann and Shappell, 2012], and Traffic Organization and Perturbation Analyzer (TOPAZ) [Stroeve et al., 2009]. These simulation-based methods have improved the ability to capture influences between agents relative to traditional hazard analysis techniques. However, these methods suffer from some of the same limitations as the chain-of-events based methods (FTA, FMEA, ETA), such as relying on the identification of component failures and their probabilities. In addition, these simulation-based methods are inappropriate for application during concept development because they require relatively complete designs [Harkleroad et al., 2013].

2.3.2 Systems Theoretic Accident Model and Process

System-Theoretic Accident Model and Processes (STAMP) was created to capture more types of accident causal factors including social and organizational structures, new kinds of human error, design and requirements flaws, and dysfunctional interactions among non-failed components [Leveson, 2004, 2012]. Rather than treating safety as a failure problem or simplifying accidents to a linear chain of events, STAMP treats safety as a control problem in which accidents arise from complex dynamic processes that may operate concurrently and interact to create unsafe situations.

Accidents can then be prevented by identifying and enforcing constraints on component interactions. This model captures accidents due to component failure, but also explains increasingly common component interaction accidents that occur in complex systems without any component failures. For example, software can create unsafe sit-

uations by behaving exactly as instructed or operators and automated controllers can individually perform as intended but together they may create unexpected or dangerous conditions.

STAMP is based on systems theory and control theory. In systems theory, emergent properties are those system properties that arise from the interactions among components. Safety is a type of emergent property. The emergent properties associated with a set of components are related to constraints upon the degrees of freedom of those components' behavior [Checkland, 1999]. There are always constraints or controls that exist on the interactions among components in any complex system. These behavioral controls may include physical laws, designed fail-safe mechanisms to handle component failures, policies, and procedures. Such controls must be designed such that the safety constraints are enforced on the potential interactions between the system components. In air traffic control, for example, the system is designed to prevent loss of separation among aircraft.

System safety can then be reformulated as a system control problem rather than a component reliability problem—accidents occur when component failures, external disturbances, and/or potentially unsafe interactions among system components are not handled adequately or controlled, leading to the violation of required safety constraints on component behavior (such as maintaining minimum separation). System controls may be managerial, organizational, physical, operational, or in manufacturing. In STAMP, the safety controls in a system are embodied in the *hierarchical* safety control structure. The next chapter describes hierarchy theory in greater detail.

Control processes operate throughout the hierarchy whereby commands or control actions are issued from higher levels to lower levels and feedback is provided from lower levels to higher levels (see Figure 8 in the next chapter). Accidents arise from inadequate enforcement of safety constraints, for example due to missing or incorrect feedback, inadequate control actions, component failure, uncontrolled disturbances, or other flaws. STAMP defines four types of unsafe control actions that must be eliminated or controlled to prevent accidents:

1. A control action required for safety is not provided or is not followed
2. An unsafe control action is provided that leads to a hazard
3. A potentially safe control action is provided too late, too early, or out of sequence
4. A safe control action is stopped too soon or applied too long

One potential cause of a hazardous control action in STAMP is an inadequate pro-

cess model used by human or automated controllers. A process model contains the controller's understanding of 1) the current state of the controlled process, 2) the desired state of the controlled process, and 3) the ways the process can change state. This model is used by the controller to determine what control actions are needed. In software, the process model is usually implemented in variables and embedded in the program algorithms. For humans, the process model is often called the "mental model" [Leveson, 2004]. Software and human errors frequently result from incorrect process models. Accidents like this can occur when an incorrect or incomplete process model causes a controller to provide control actions that are hazardous. While process model flaws are not the only cause of accidents in STAMP, it is a major contributor.

The generic control loop in Figure 6 shows other factors that may cause unsafe control actions. Consider an unsafe control action for an air traffic controller: a flight crew is instructed to increase altitude while another aircraft is flying through that new altitude. The control loop in Figure 6 would show that one potential cause of that action is an incorrect belief that the airspace above the aircraft is clear (an incorrect process model). The incorrect process model, in turn, may be the result of inadequate feedback provided by a failed sensor, the feedback may be delayed, the data may have been corrupted, etc. Alternatively, the system may have operated exactly as designed but the designers may have omitted a feedback signal or the feedback requirements may be insufficient.

Techniques Based on STAMP

Systems-Theoretic Process Analysis (STPA) is a hazard analysis technique that is based on the STAMP accident model. Causal Analysis based on STAMP (CAST) is used for accident analysis, and STPA-Sec is used for security analysis. STPA is significantly more powerful than failure-based techniques in the ability to capture a wider array of hazardous behaviors, including organizational aspects, requirements flaws, design errors, complex human behavior, and component failures [Leveson, 2012]. While many hazard analysis techniques stop once a sequence of events or failures has been identified, STPA helps explain the complex reasons *why* a sequence of events might occur, including underlying processes and control flaws that may exist without any component failure.

Although STPA is relatively new compared to traditional methods, it has been demonstrated successfully on a wide range of systems including aviation [Fleming et al., 2013], spacecraft [Ishimatsu et al., 2010; Nakao et al., 2012; Fleming et al.,

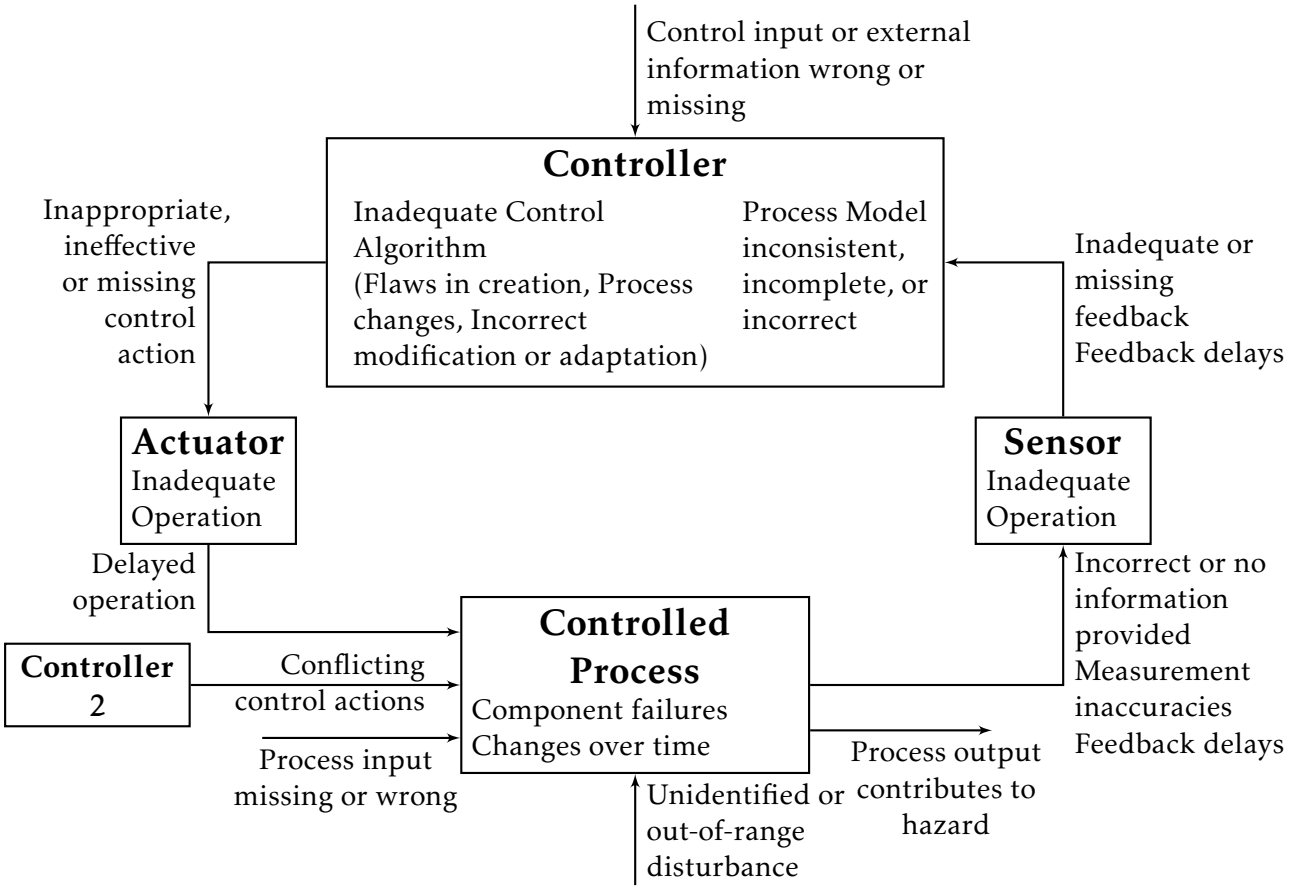


Figure 6: STPA Control Loop with Causal Factors

2012], missile defense systems [Pereira et al., 2006], civil infrastructure [Dong, 2012], and others.

STPA has only been applied to existing, operational systems or to projects with a significant amount of design detail, although Harkleroad et al. [2013] identified it as a potentially effective method during concept development. STPA has been most effectively applied when the actions available to a control agent are discrete or when an agent's available actions are pre-specified. In addition, STPA has not been used to compare architectures or design tradeoffs in terms of safety-related risk.

Figure 7 shows the analytical tools that are based on the STAMP accident causality model. In addition to STPA, CAST is an accident investigation tool based on STAMP, and STPA-Sec is a new technique used to identify and control vulnerabilities in the security domain [Young and Leveson, 2014]. These tools are especially adept at capturing behavior in modern complex human- and software-intensive systems where component interaction accidents (or security incidents) have become increasingly common

and traditional chain of events models are inadequate. STECA, the new concept development technique described in the next chapter, is also based on the STAMP accident causality model. These tools then support more general processes, such as systems engineering, management, operations, and regulation.

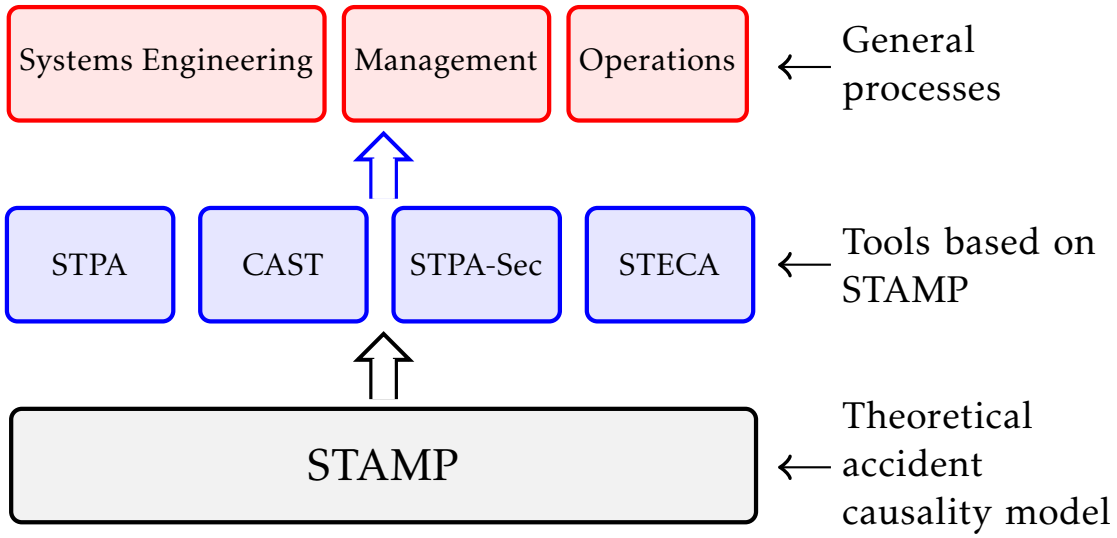


Figure 7: Techniques based on STAMP Accident Causality Model

2.3.3 General Assessment of Accident Models and Hazard Analysis Techniques

Tomorrow's aerospace systems will be highly coupled, involve nonlinear dynamics, and require changing roles among human operators and software systems. The hazard analysis must be able to handle these issues. Additionally, the motivation described in Chapter 1 calls for a method that can be applied early during concept development and system architecting, when little design detail is available and analytical techniques can help guide system development.

There are two general (and related) limitations to many of the traditional hazard analysis techniques, which include FTA, FMEA, and ETA. Although attempts have been made to extend some of these methods to include interaction and dependence among causal factors, the methods are still limited by their underlying accident causality models. This chain-of-events causality model assumes that accidents are caused by a linear succession of discrete, failure-based events and necessarily omits feedback and interaction elements of accident causality.

Moreover, these methods came along when engineered systems were much different than today. The use of software and computer systems has exploded since the

1960s, and although many attempts have been made to adapt these methods the original assumptions persist.

FMEA is a good method for analyzing component reliability, while FTA is a more efficient hazard analysis technique because it considers only those failures that lead to the higher level events. ETA is only effective when events proceed in a consistent, chronological order. None of the above techniques are particularly effective during concept development and tradespace exploration, when little design detail is available.

STPA has many advantages over the traditional techniques due to its underlying model of accident causation and the guidance it provides analysts in identifying hazardous scenarios. Because of its demonstrated ability to identify software design and requirements flaws; potentially hazardous, context-dependent human behavior; and hazardous interaction among components, STPA has the most potential for successful application to complex aerospace systems. In addition, STPA relies on a functional model of the system, as opposed to a physical model, which provides further rationale for its application in the early phases of development. STPA will need to be extended in order to handle the problems inherent in concept development and system architecting.

2.4 Summary

The systems engineering approaches during concept development and tradespace exploration do not explicitly consider and thereby do not provide a formal, systematic means of accounting for safety-related properties. The initial safety-related activity in most aerospace projects, the preliminary hazard analysis, is limited by an accident causality model that pre-dates the development of computer-intensive systems. The guidance given for conducting preliminary hazard analysis is restricted to consideration of mechanical failures or oversimplifies the role of human operators and software in modern, complex systems.

While progress has been made in generating safety-related requirements during certain systems engineering activities, little has been done to include safety in other activities such as concept generation and tradespace exploration. In fact, given the current body of tradespace exploration literature it may be inappropriate to include safety analytically (currently), due to the necessity of directly measurable metrics in many of the tradespace exploration frameworks.

A new approach is needed to overcome these difficulties. This thesis introduces a new process, based on the STAMP model, for analysis of a concept of operations to assist in safety-driven design from the early stages of system engineering.

[Page intentionally left blank]

Chapter 3
Systems-Theoretic Early Concept Analysis

> You think because you understand "one" you must understand "two" because "one and one" makes "two". But you forget you must also understand "and".
>
> –Rumi

Chapter 1 outlined two broad objectives for this research. The first objective is to develop rigorous, systematic tools for the analysis of future concepts in order to identify hazardous scenarios and undocumented assumptions. The second objective is to extend these tools to assist stakeholders in the development of concepts using a safety-driven approach. Both goals especially apply to systems where the tradespace includes human operation, automation or decision support tools, and the coordination of decision making agents.

In order to improve upon the existing state of practice and theory of including safety during concept development, a methodology should help analysts and stakeholders to *systematically*:

Objective 1) identify *missing information* or *undocumented assumptions* that will be required for effective operation of the system;

Objective 2) identify *inconsistent* or *conflicting* information within a concept that may lead to hazardous behavior;

Objective 3) identify where *more specific* operational concepts are required to understand safety- and functionally-related behavior of the system;

Objective 4) identify *requirements* or safety *constraints* for O1–O3; and

Objective 5) identify the *mitigation strategies* associated with factors identified in O1–O3

Because of this thesis' emphasis on introducing systematic methods into earlier phases of systems engineering, it is important to note a general distinction between the above objectives. The first three objectives (O1–O3) involve developing and applying

more sophisticated techniques in order to improve upon the current limitations of PHA, which lack systematic guidance and focus on component failure. The latter two objectives (O4–O5) extend to slightly different phases and goals of system engineering. Generating safety-related requirements and identifying mitigation strategies should explicitly be part of the early systems engineering effort and result in a process called *safety-guided design* and development.

To put these objectives in perspective, consider the Vee Model in Figure 2, on page24, and the following question: what is needed to move from a high-level Concept of Operations to later stages of systems engineering? By identifying hazardous scenarios in the ConOps and using this information to generate requirements, the approach can assist stakeholders and engineers in developing a system architecture.

Before describing the model-based approach to achieving these objectives, it is important to develop the theoretical underpinnings of the approach.

3.1 Theoretical Foundations

This thesis extends the System Theoretic Accident Model and Process (STAMP) and its associated hazard analysis technique (STPA – Systems-Theoretic Process Analysis) for analysis of an existing ConOps. The reasons for selecting this accident model and its associated analytical processes include: (1) during early system engineering activities, STAMP's focus on the functional behavior of a system makes it a strong candidate relative to other techniques that rely on analyzing physical hardware [Harkleroad et al., 2013]; (2) STAMP's ability to identify component interactions, software design flaws, and potential sources of hazardous human behavior, which are prevalent in many future systems under development in the aerospace industry; and (3) the systems- and control-theoretic underpinnings of the STAMP accident causality model can potentially be extended into more rigorous, formalized techniques that can guide early system engineering efforts.

The focus of this thesis requires further explanation of some of the general characteristics that are often present during concept development. The primary artifact of concept development, the ConOps (see Chapter 2), often consists of natural language text and/or low fidelity graphical depictions of work- and information-flows. ConOps should contain some reference to stakeholder goals and system-level requirements, but they rarely contain specific design requirements.

Because a ConOps is developed long before the system becomes operational, it typ-

ically includes (often implicit) assumptions about the future that are not necessarily true when the document is being developed. For example, a ConOps may contain assumptions about future technologies that do not yet exist. Finally, often in practice a ConOps is developed by committee, with disparate members who have different goals and perspectives [JPDO, 2011]. All of these characteristics make it difficult to systematically analyze a Concept of Operations and in particular to rigorously, systematically achieve Objectives 1-3 in the introduction to this chapter.

3.1.1 Systems Theory, STAMP, and STPA

STAMP and STPA are explained briefly in chapter 2, and the proposed extension rests on two related principles of the STAMP model of accident causality; (1) emergence and hierarchy and (2) communication and control. In fact, these two principles form the basis of general systems theory.

In systems theory, a level of complexity is characterized by properties that do not exist at lower levels [Checkland, 1999]. These properties are called *emergent*, and the study of any property that cannot be accounted for at lower levels of complexity is referred to as the *theory of emergence*. Safety is an emergent property, not a property of individual components.

For example, consider a pilot and an air traffic controller. One of the primary objectives of a pilot is to ensure that the aircraft follows a stable trajectory. Alternatively, from the perspective of an air traffic controller, who is tasked with managing (part of) the national airspace, the aircraft following a stable trajectory does not by itself constitute safety. From the level of the national airspace system, safety only *emerges* when the air traffic controller coordinates multiple aircraft trajectories and considers terrain, airspace restrictions, weather, aircraft capability, and other factors.

Hierarchy theory is concerned with the fundamental differences between one level of complexity and another. In the airspace example, at one level of complexity an air traffic controller must manage the trajectories of all the aircraft in his or her sector. At another level, a pilot simply cares about following his or her assigned trajectory. The pilot then must manage lower level characteristics such as thrust and control surface trim, which when combined will help in achieving that trajectory. Hierarchy theory provides an account of the relationships between different levels and how hierarchies are formed [Checkland, 1999]. That is, what separates the levels and what links them?

In a hierarchy of open[1] systems, such as the national airspace, maintaining the

[1] An open system is one that exchanges material, energy, and information with its environment.

hierarchy involves processes where there is *communication* of information in order to *control* lower levels. Checkland [1999] states that a system requires communication and control if it is "to survive the knocks administered by the systems' environment". Returning to the air traffic control example, if the air traffic controller is tasked with ensuring safe separation between aircraft, (s)he must have means to control or impose constraints on individual aircraft behavior. The air traffic controller must also be provided information about aircraft states, for example if aircraft have to change altitude due to turbulence. Therefore, *imposing constraints* (control) plays a fundamental role in the proposed approach in this thesis, as does *feedback* (communication).

Four conditions are required for process control [Ashby, 1957; Leveson, 2012]:

1. *Goal* condition: the controller must have a goal or goals
2. *Action* condition: the controller must be able to affect the state of the system, typically by means of an actuator or actuators.
3. *Model* condition: the controller must contain a model of the system
4. *Observability* condition: the controller must be able to ascertain the state of the system, typically by feedback from a sensor

Figure 8 depicts a hierarchical system, where the components interact with their environment, one level imposes constraints on the level below it, and feedback about its performance is transmitted back to the level above it. The proposed approach in this thesis recognizes that safety is an emergent property. System components interact with the environment and with each other, and safety is enforced by a set of control laws or goal conditions that constrain the behavior of these components. With safety viewed as a control problem, accidents occur when component failures, external disturbances, and dysfunctional interactions among components are not adequately controlled [Leveson, 2012]. This view of accident causality forms the basis of STAMP, and STPA is the hazard analysis approach based on the STAMP model.

3.1.2 Outline of Approach

These concepts—hierarchy and emergence, and communication and control—are fundamental to the model-based approach proposed in this thesis. For the purposes of analysis, these concepts should be used in the opposite order. Control-theoretic concepts are used first to construct a model of the system, and theories of hierarchy and emergence (in addition to control and communication) are then used to interrogate

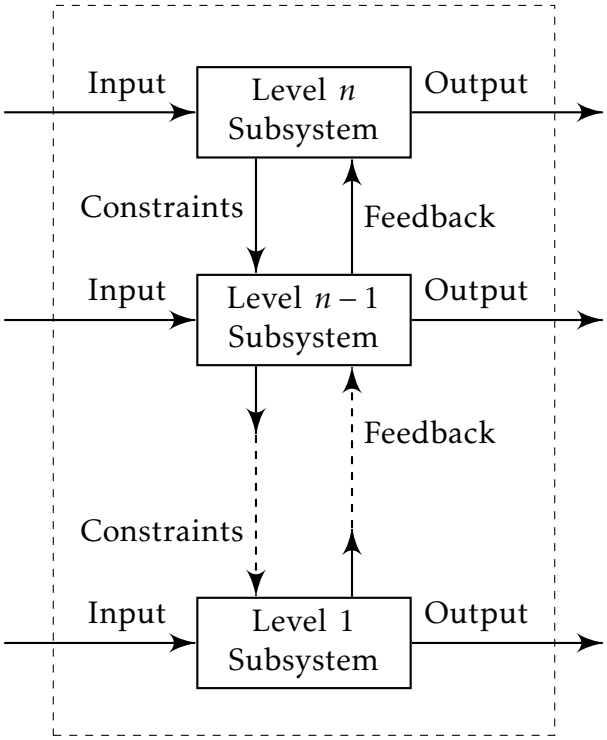


Figure 8: Basic Features of a Hierarchical System (adapted from [Mesarovic et al., 1970])

the model itself. The process is conducted according to Figure 9, where the main contributions from this extension are represented by the lower four boxes. The following sub-sections describe the theoretical development as well as provide a brief example for illustrative purposes.

Like a typical STPA hazard analysis, the systems-theoretic early concept analysis (STECA) begins with accidents and hazards, a high level decomposition of control functions, and then a set of high level safety responsibilities. These are basic system and safety engineering activities that should be done for any project (first box, Figure 9). Chapter 4 provides an example of how to identify a hierarchical list of safety responsibilities that is based on systems theory.

3.2 Systematic Control Model Development

Potential benefits of model-based systems engineering include the use of repeatable processes, promoting consistent views of the system, and formal application of modeling to support requirements generation, design, analysis, and verification [Friedenthal et al., 2007]. It is in this vein that this research seeks to develop ConOps in terms of models rather than informal documentation.

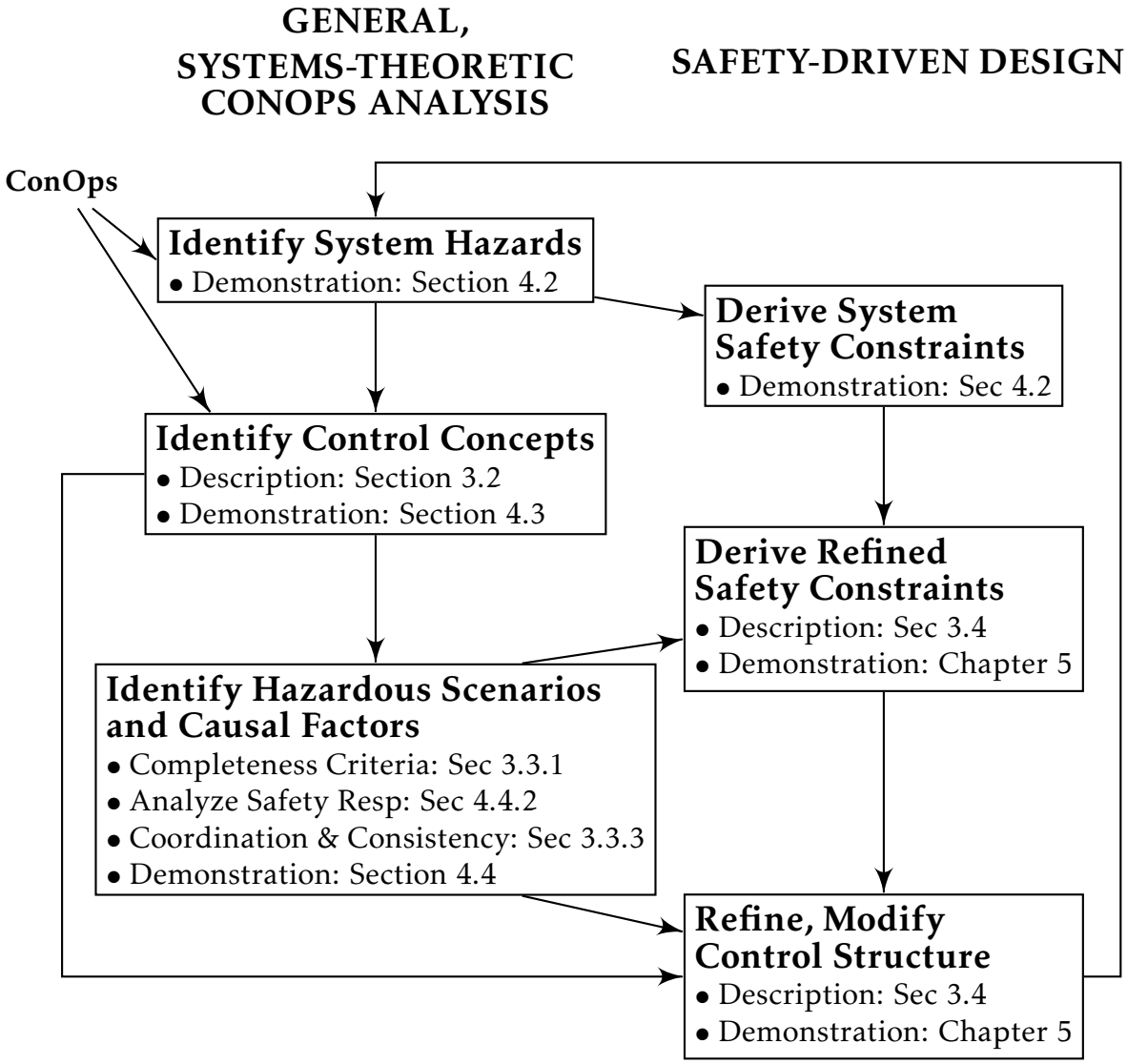


Figure 9: Proposed Methodology—STECA

Recall that the objectives of this research are to identify missing information or undocumented assumptions that will be required for effective operation of the system; identify inconsistent or conflicting information within a concept that may lead to hazardous behavior; and identify where more specific operational concepts are required to understand safety- and functionally-related behavior of the system. Consider also that a ConOps typically contains natural language text and graphical depictions of operating concepts, neither of which contain specifications nor rigorous, formal accounting of roles and responsibilities. The following modifications of STPA are needed to allow an analyst to rigorously and systematically develop a system model based on the descriptions contained in a ConOps.

Consider the STPA causal factors described in Chapter 2 and included again here

in Figure 10. These guide-words are very effective for analyzing complete or nearly complete designs or specifications, and for identifying potential *flaws* in a system. However, a control-theoretic approach can also be used for understanding and specifying how a system *should* behave. Recognizing the roles that components of the control loop play in enforcing safe behavior can help in developing and analyzing a concept, where detailed design information is unavailable and using the guide-words in Figure 10 may not yet be the most effective.

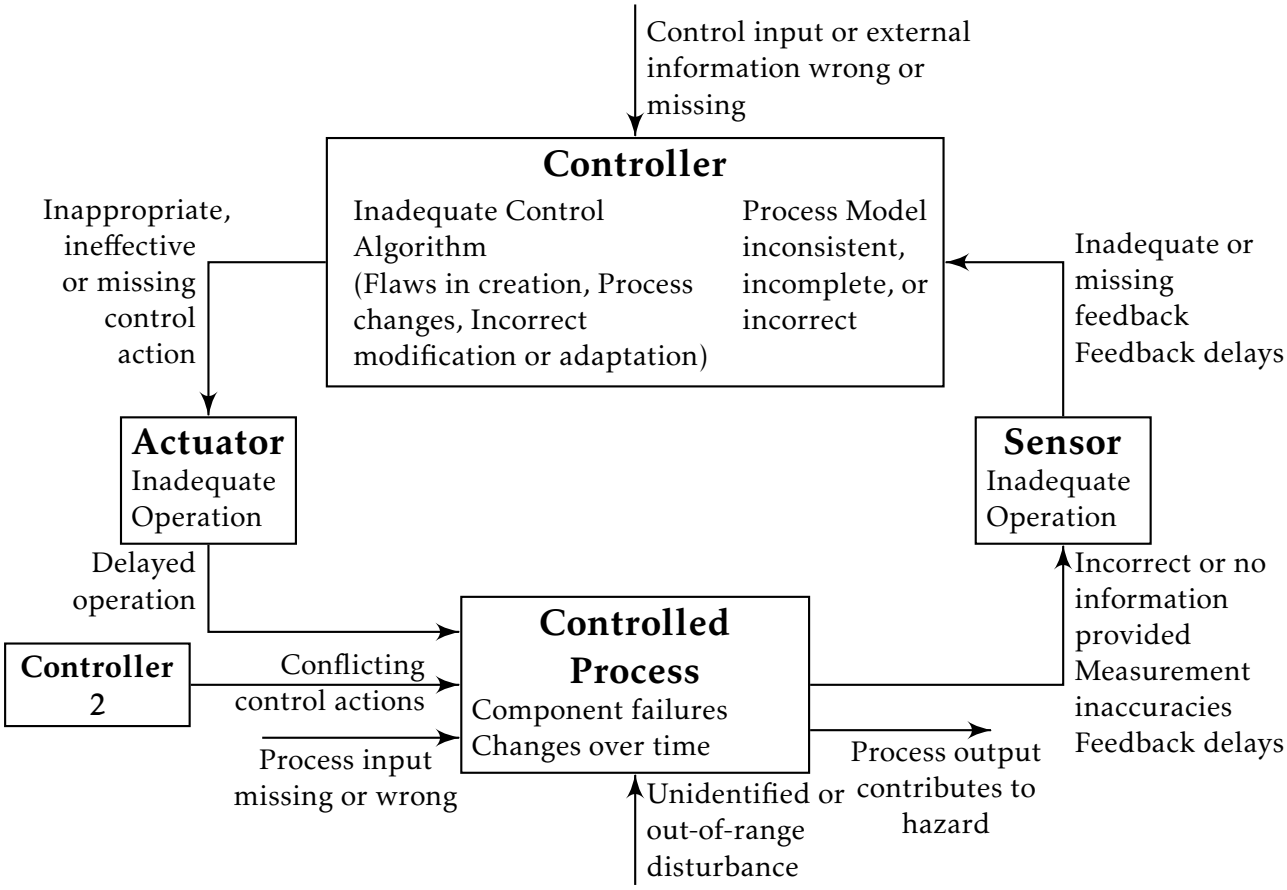


Figure 10: STPA Control Loop with Causal Factors

3.2.1 Operational Roles in a Control-Theoretic Framework

Rather than directly using the control flaws (causal factors) from Figure 10, first examine the basic functions of each entity in the control loop. That is, what is required of each entity in the control loop for effective, safe system behavior? What are the responsibilities of the controller, actuator, controlled process, and sensor? How do these entities interact with each other, with the environment, and with other control loops?

The Controller:

- creates, generates, or modifies control actions based on algorithm or procedure and perceived model of system
- processes inputs from sensors to form and update process model
- processes inputs from external sources to form and update process model
- transmits instructions or status to other controllers or entities in the system

The Actuator:

- translates controller-generated action into process-specific instruction, force, heat, torque, or other mechanism

The Controlled Process:

- interacts with environment via forces, heat transfer, chemical reactions, or other input
- translates higher level control actions into control actions directed at lower level processes (if it is not at the bottom of a control hierarchy)

The Sensor

- transmits continuous dynamic state measurements to controller (i.e. measures the behavior of controlled process via continuous or semi-continuous, digital data)
- transmits binary or discretized state data to controller (i.e. measures behavior of process relative to thresholds; For example, sensor has algorithm built-in to determine a threshold but has no control authority)
- synthesizes and integrates measurement data (e.g. takes location data from different types of sensors to create an estimate, like a Kalman filter)

This information can be built into a template that analysts and stakeholders use when developing, analyzing, and discussing a concept of operations. In fact, this information can be formalized into a formal, mathematical model that can be rigorously queried to ensure completeness and consistency. Such a formalization will be shown in the next sections.

The roles of the controller, actuator, controlled process, and sensor, and their interactions with the environment and other control loops can be summarized with 15 generic keywords or guide words. Figure 11 on the next page depicts these guide words in the familiar control loop format. With a proper accounting of these 15 items,

the control loop can achieve the four necessary conditions[2] of process control and adequately interact with its environment, other processes, and other controllers. In other words, these guide words are necessary to ensure that a control loop is controllable and coordinable with other controlled processes.

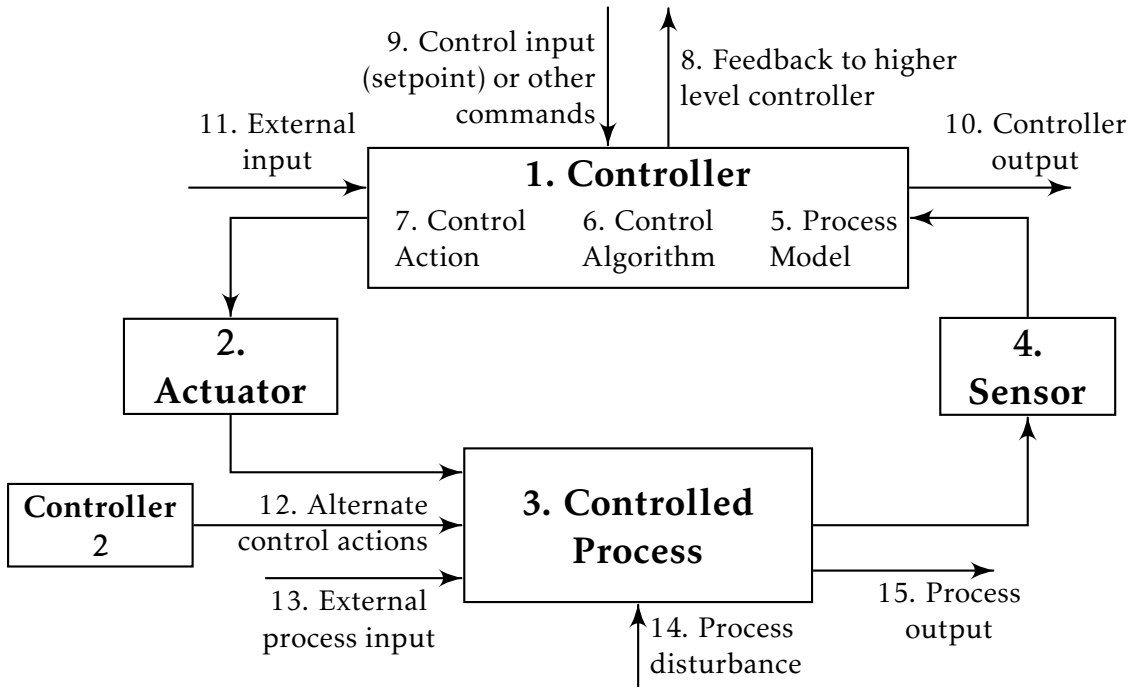


Figure 11: Control Loop with generic entities

The information in Figure 11 and the above lists (Controller, Actuator, Controlled Process, Sensor) can then be used to systematically parse and query the natural language description or graphical depiction in a concept of operations. The resulting model and subsequent database are easy to interrogate and visualize. These qualities help the analyst to check for internal inconsistencies and/or missing information that may result in unsatisfied control conditions, and also to check for inconsistencies across the system hierarchy.

Table 6 provides a series of prompts that an analyst can use when reading a text or graphic in a ConOps.

In order to obtain a "complete" model of the ConOps, this model development approach should be applied recursively over the entire ConOps document. The keywords, with associated questions and comments (Tables 6 and 7), can be applied to

[2] See page 56.

Table 6: Control-theoretic Analysis of Text

Source / Subject	What is the primary subject of the text? What is the primary source of action that the text (or graphic) is describing?
Role	Is the Source or Subject a Controller, Actuator, Controlled Process, or Sensor?
Behavior Type	For the given role, which type(s) of behavior does it exhibit? See the lists starting on page 60.
Context	Provide a justification for categorizing the text (or graphic) in the chosen manner.

individual sentences, paragraphs, and/or graphical depictions[3].

3.2.2 Formal Expression of Model Development

This section develops a mathematical formalism that is intended to achieve two ends. First, the formalism allows the analyst to achieve more rigor than using the text- and graphics-based descriptions in the previous section. That is, the formalism allows the analyst to apply the technique in a repeatable fashion and develop a model that is easier to query. Second, the formalism lends itself to tool development in future work. The model-based systems engineering paradigm is ultimately focused on developing both the theory and tools to assist in managing complexity and assisting in development and analysis. A control-theoretic formalism of a Concept of Operations is as follows.

Using the development from Mesarovic et al. [1970], the basic feature of a hierarchical system is that at one level a subsystem applies constraints to, and receives feedback from, a lower level subsystem (see Figure 8 on page 57). These features are now re-formulated in control-theoretic terms. A controller, actuator, process, and sensor at some level i will be denoted as $\mathcal{C}^i$, $\mathcal{K}^i$, $\mathcal{P}^i$, and $\mathcal{L}^i$, respectively. What is important to note in hierarchy theory is that the *controlled process* at one level can actually be a *controller* at the level below. That is

$$\forall i \neq 1, \quad \mathcal{P}^i = \mathcal{C}^{i-1} \tag{4}$$

where the controller $\mathcal{C}^{i-1}$ has its own actuators, controlled processes, and sensors $\left(\mathcal{K}^{i-1}, \mathcal{P}^{i-1}, \mathcal{L}^{i-1}\right)$. Without loss of generality, the following development drops the superscript notation, except where necessary.

[3] For example, a graphical depiction of information flows in a ConOps could provide the information necessary for model development

Table 7: Database Version of Control Model

See Fig. 11		**Description**
1.	Controller	Which controller is being described in the text?
2.	Actuator	What mechanism(s) does the control have in order to affect the process?
3.	Cntl'd Process	What process does the controller have control over?
4.	Sensor	What type of feedback does the controller receive about the process it controls?
5.	Process Model	What states and variables does the controller know about the process it controls?
6.	Cntl Algorithm	Does the controller use an algorithm or procedure to generate action?
7.	Control Actions	What types of action can the controller generate?
8.	Controller Status	Does the controller provide feedback to higher level controllers?
9.	Control Input	Does the controller receive set points or other types of commands?
10.	Controller Output	Does the controller have output other than through the actuator? This often includes transmission of information to other controllers.
11.	External Input	Does the controller receive external input, either in terms of other system information or other controller action(s), or other (e.g. a power source)?
12.	Alt Controller	Does the process receive action from controllers other than in item 1, 2?
13.	Process Input	Does the process require external input to function? Examples include pressure, power, and heat.
14.	Proc Disturbance	What environmental factors does the process interact with?
15.	Process Output	Does the system require that the process output something to other components? (e.g. power, pressure)

The entire ConOps document is denoted, $\mathbb{C}$, and the document consists of a structured group of elements that contain information about how the concept should behave. These elements could be sentences, paragraphs, or graphical objects such as figures that depict information flows or sequences of events. These "information ele-

ments" are denoted $\mathcal{I}$. The ConOps is then the set:

$$\mathbb{C} = \left\{ \bigcup_i \mathcal{I}_i \;\middle|\; (\mathcal{I}_i \in \mathbb{I}) \wedge \left(\mathcal{I}_i \cap \mathcal{I}_j = \emptyset\right),\ \forall i \neq j;\ i, j \in N \right\} \tag{5}$$

where $\mathbb{I}$ is any element that conveys information about operational concepts in terms of prose or graphics, and N is the total number of elements in a document. Equation 5 ensures that the model generation process does not use duplicate or overlapping information. The process is repeated recursively over each element $\mathcal{I}$, resulting in completeness while avoiding duplication and potential inconsistency. Decomposing a document into a coherent, mutually exclusive set of information elements requires some basic understanding of grammar[4].

Identifying the Components of the Model

The following formalism provides guidance for how to identify the elements necessary to generate a model of the concept. Each information element $\mathcal{I}$ is defined as a tuple $(\mathcal{S}, \mathcal{R}, \mathcal{B}, \mathcal{A})$ where:

- $\mathcal{S}$ is the source or subject of the information object $\mathcal{I}$. Any complete sentence in English should have a subject and predicate. Identifying $\mathcal{S}$ is often similar to identifying the subject of a sentence in grammar, while identifying the predicate of the sentence yields the rest of the information in the tuple. The information object, $\mathcal{I}$, could be a set of sentences or paragraphs that share the same subject. Identifying the subject of a graphical object may not be as straightforward as in natural language text, and graphics may contain multiple subjects or sources of responsibility[5]. Much of the information of a graphical object can be inferred using the associated text that refers to the graphic or by using the model or data that underlies the graphics.

- $\mathcal{R}$ is the responsibility of the subject in control-theoretic terms.

$$\mathcal{R} \in \{\mathcal{C}, \mathcal{K}, \mathcal{P}, \mathcal{L}\} \tag{6}$$

[4] To a certain extent, decomposing a document also depends on the competency of the original authors.

[5] In fact, this is one of the benefits of using graphical depictions and of model-based systems engineering in general. That is, graphical depictions allow for the storage of many different kinds of information in one concise space.

where

$$\begin{aligned}\mathcal{C} &:= \text{ controller,}\\ \mathcal{K} &:= \text{ actuator,}\\ \mathcal{P} &:= \text{ process being controlled,}\\ \mathcal{L} &:= \text{ sensor.}\end{aligned}$$

- $\mathcal{B}$ is the type of behavior prescribed to the source, and $\mathcal{B}_{\mathcal{R}}$ represents the possible behaviors ascribed to an arbitrary responsibility $\mathcal{R}$. The available behavior types are

$$\mathcal{B} \in \left\{\mathcal{B}_c, \mathcal{B}_{\mathcal{K}}, \mathcal{B}_p, \mathcal{B}_{\mathcal{L}}\right\}. \tag{7}$$

  – The controller ($\mathcal{B}_c$) represents a transformation $\mathcal{F}_c$ from input signals, $\mathcal{I}_c$, to output signals, $\mathcal{O}_c$. The input-output model for the controller is

$$\mathcal{B}_c = \left\{(\mathcal{I}_c, \mathcal{O}_c) \mid \mathcal{O}_c = \mathcal{F}_c \times \mathcal{I}_c,\ \mathcal{F}_c = f\,(G, \rho)\right\} \tag{8}$$

  where the set of controller inputs, $\mathcal{I}_c$, consists of feedback information, $\mathcal{I}_s$, communications from other controllers, $\mathcal{I}_o$, and higher level commands or set points, $\mathcal{R}$. The set of controller outputs,$\mathcal{O}_c$, consists of the available control actions, $\mathcal{O}_a$ and information transmission to other controllers, $\mathcal{O}_o$. The control function, transformation $\mathcal{F}_c$, is a function of the algorithm (or procedure, decision-making process, or policy), G, and process model, ρ. Furthermore, formation and maintenance of the process model is performed via feedback and external information sources, ρ_s and ρ_e, respectively. Therefore, in terms of operational concepts, a controller is comprised of a sub-set of behaviors related to processing inputs and generating outputs:

$$\begin{aligned}\mathcal{B}_c \in &\left\{\mathcal{B}_{\mathcal{I}_c}, \mathcal{B}_{\mathcal{O}_c}\right\},\\ &\mathcal{B}_{\mathcal{I}_c} : \mathcal{I}_c \to \rho,\\ &\mathcal{B}_{\mathcal{O}_c} : \rho \times G \to \mathcal{O}_c.\end{aligned} \tag{9}$$

  – Actuator behavior ($\mathcal{B}_{\mathcal{K}}$) is a signal mapping from controller commands, $\mathcal{O}_c$, to manipulated process variables, $\mathcal{V}_c$. The behavioral model of the actuator

is

$$\mathcal{B}_{\mathcal{K}} = \{v \mid \mathcal{O}_c \rightarrow \mathcal{V}_c\} \tag{10}$$

– Process behavior $\left(\mathcal{B}_p\right)$ is a transformation $\mathcal{F}_p$ from input signals, $\mathcal{I}_p$, to output signals, $\mathcal{O}_p$.

$$\mathcal{B}_p = \left\{ \left(\mathcal{I}_p, \mathcal{O}_p\right) \,\middle|\, \mathcal{O}_p = \mathcal{F}_p \times \mathcal{I}_p,\ \mathcal{F}_p = f\left(\mathcal{V}\right) \right\} \tag{11}$$

where the set of process inputs, $\mathcal{I}_p$, consists of an actuator signal intended to manipulate certain variables of the process, $\mathcal{V}_m$, actions from other controllers, $\mathcal{V}_A$, external process inputs, $\mathcal{I}_e$, and disturbances, $\mathcal{D}$. $\mathcal{F}_p$ is the process dynamics of the system. The set of process outputs, $\mathcal{O}_p$, consists of external outputs, $\mathcal{O}_e$ and signals related to variables under control, $\mathcal{V}_c$. A controlled process exhibits two behavior types:

$$\begin{aligned} \mathcal{B}_p \in & \left\{ \mathcal{B}_{\mathcal{C}_p}, \mathcal{B}_{\mathcal{D}_p} \right\}, \\ & \mathcal{B}_{\mathcal{C}_p} : \mathcal{I}_c^i \rightarrow \mathcal{I}_c^{i-1}, \\ & \mathcal{B}_{\mathcal{D}_p} : \mathcal{V}_m \times \mathcal{V}_A \times \mathcal{I}_e \times \mathcal{D} \rightarrow \mathcal{O}_e \times \mathcal{V}_c. \end{aligned} \tag{12}$$

Note again that for any controller, $\mathcal{C}^i$ (re-introducing the superscript notation), the role of its controlled process is also that of a controller of a lower level process when $i \neq 1$. In other words, process behavior of type $\mathcal{B}_{\mathcal{C}_p}$ should automatically trigger a recursion, where a lower level analysis produces another set of actuators, processes, and sensors. That is,

$$\mathcal{B}_{\mathcal{C}_p} \implies \mathcal{P}^i = \mathcal{C}^{i-1} \tag{13}$$

subject to the rule in equation (4). Rather than exhibiting the dynamic behavior typically associated with process control, the transformation performed by the process is equivalent to the mapping performed by a controller $\left(\mathcal{F}_p^i \equiv \mathcal{F}_c^{i-1}\right)$. Alternatively, at the bottom level,

$$i = 1 \iff \mathcal{B}_p \equiv \mathcal{B}_{\mathcal{D}_p}. \tag{14}$$

Thus, if the process is at the lowest level of the system hierarchy, the transformation $\mathcal{F}_p$ represents the system dynamics. In the controls literature this

mapping is often represented in state space as a dynamic, feedback control system of the form,

$$\dot{x}(t) = f(x(t), u(t)) \tag{15}$$
$$y(t) = g(x(t), u(t))$$

where $x(t)$ represents the dynamics of the system subject to control input $u(t)$. The second equation, $y(t)$, represents the feedback terms. In most engineered systems, the hazardous states that should be eliminated or mitigated can be approximated by a vector of continuous or semi-continuous functions, as in equation (15)[6].

– Sensor behavior ($\mathcal{B}_{\mathcal{L}}$) represents a signal mapping from measured variables, $\mathcal{V}_m$, to controller inputs, $\mathcal{I}_s$. The behavioral model of the actuator is

$$\mathcal{L} = \{\mathrm{i} \mid \mathcal{V}_m \rightarrow \mathcal{I}_s\} \tag{16}$$

The type of feedback controller input defines the behavior of the sensor, which consist of the set

$$\mathcal{B}_{\mathcal{L}} \in \left\{\mathcal{B}_{\mathcal{L}_c}, \mathcal{B}_{\mathcal{L}_d}, \mathcal{B}_{\mathcal{L}_s}\right\} \tag{17}$$
$$\mathcal{B}_{\mathcal{L}_c} : \mathcal{V}_m \rightarrow \mathcal{I}_{s,c}$$
$$\mathcal{B}_{\mathcal{L}_d} : \mathcal{V}_m \rightarrow \mathcal{I}_{s,d}$$
$$\mathcal{B}_{\mathcal{L}_s} : \mathcal{V}_m \rightarrow \mathcal{I}_{s,s}$$

where $\mathcal{I}_{s,c}$ is a continuous signal (or semi-continuous digital signal) representing the evolution of the process, $\mathcal{I}_{s,d}$ is discrete data representing a state transition of the process $\mathcal{P}$, and $\mathcal{I}_{s,s}$ is a synthesis of measured process variables into a lower-dimensional data stream ($\dim(\mathcal{I}_{s,s}) < \dim(\mathcal{V}_m)$).

- $\mathcal{A}$ is the context or set of assumptions, which provides the analysts' justification for assigning the first three elements to the quadruple $\mathcal{I}$. Context can simply be a textual rationale, reference to other parts of the ConOps, reference to other documents, or other means of inference. $\mathcal{A}$ is ultimately not a part of the control-

[6] In other systems that have very real hazards, such as investment banking, such dynamic equations may not exist. However, a mapping $\mathcal{F}_p$ for inputs to outputs should still exist.

theoretic model development, but it is an important aspect of making explicit the often undocumented assumptions that are present during concept development. Each information element, $\mathcal{I}$, can easily be stored in a database consisting of the source or subject, its responsibility within the system, its behavior, and the assumptions or context used to define, identify, or classify the information element. Not only can the model that results from the aggregation of tuples be analyzed (see next section), but also the model is easily traceable back to the original ConOps.

Because this thesis advocates for a systems approach to developing a concept, it is insufficient to simply identify and store all the model information according to the tuple $(\mathcal{S}, \mathcal{R}, \mathcal{B}, \mathcal{A})$ and then analyze their individual behavior. The analyst must also identify the relationships between all of the elements and then analyze both component behavior and interactions between those components.

Synthesizing Information into Hierarchical System Model

The previous development involves parsing the ConOps by mapping information elements $\mathcal{I}$ into control-theoretic constituents defined by the tuple $(\mathcal{S}, \mathcal{R}, \mathcal{B}, \mathcal{A})$. The resulting tuples do not, however, by themselves represent a model of the entire concept. The above analysis should result in a set of controllers, each with its own actuators, processes, and sensors. Section 3.1 describes the importance of emergence and hierarchy in systems theory, but it is important to develop that theory further here. In systems theory it is inappropriate to analyze individual control loops and then make a determination about the overall behavior of the system. Furthermore, it is inappropriate to analyze individual components like sensors, actuators, or controllers.

Rather, the behavior of the system can only be determined in the context of all the components and their interactions. Instead of focusing solely on understanding the behavior of each component, the relevant issue here relates to how the individual control elements relate to each other. This section develops a formalism, based on hierarchy theory, to determine the relationships between control components.

Though the following section presents a formalism for checking the consistency across the hierarchy, this section presents heuristics for *identifying* or constructing the hierarchy based on information contained in the ConOps. This research proposes the use of several abstractions that can be used to determine the "vertical" and "horizontal" relationships between control components. This section also introduces the formal mathematical notation to be used later.

Several different (but related) notions of hierarchy, or abstraction, may be used to

generate a system control model from the individual control loops generated from—and stored in—the tuples $(\mathcal{S}, \mathcal{R}, \mathcal{B}, \mathcal{A})$. For example, Mesarovic and Takahara [1975] describe three types of hierarchies: strata or levels of description, layers or levels of decision complexity, and echelons or organizational decomposition. In the controls literature the hierarchy is typically layered in terms of time scale, for example, scheduling (weeks), system-wide optimization (days); local optimization (hours); supervisory, predictive, or advanced control (minutes); and regulatory control (seconds) [Skogestad, 2004]. The echelon hierarchy—described by Mesarovic and used in the controls literature [e.g. Morari et al., 1980]—is often used to decompose a system using the notion of decision-making authority. That is, some decision-making units are influenced or controlled by others.

A specific characteristic of the echelon hierarchy is that there are many elements within a given level, which implies another dimension of organization. Intent Specifications [Leveson, 2000b] organize system information according to three types of hierarchy: level of intent, part-whole abstractions, and refinement. Part-whole abstraction provide another horizontal decomposition of the system. That is, while decision complexity, time scale, or authority defines a hierarchy *vertically*, part-whole abstractions describe the organization and relationships *horizontally* within a given level.

Hierarchical Control Structure

Any level of a system can be represented as $S : X \to Y$, a mapping from a set of outside stimuli, X to a set of responses, Y.

$$X = X_1 \times X_2 \times \cdots \times X_n, \quad \text{and} \quad Y = Y_1 \times Y_2 \times \cdots \times Y_n. \tag{18}$$

The i^{th} level of the system is then the mapping

$$S_i : X_i \to Y_i \tag{19}$$

The previous step should have identified a set of stimuli and responses, in particular in the controller and process behaviors, $\mathcal{B}_c$ and $\mathcal{B}_p$, respectively. The i^{th} mapping is then comprised of the controller inputs and outputs. From equation (8), the mapping

consists of:

$$X_i = \left\{\mathcal{I}_o^i, \mathcal{I}_S^i, \mathcal{R}^i\right\} \tag{20}$$

$$Y_i = \left\{\mathcal{O}_a^i, \mathcal{O}_o^i, \mathcal{F}^i\right\} \tag{21}$$

Building on the concept introduced in equation (4),

$$\begin{aligned}
&\text{(i)} \quad S_i: \ \mathcal{I}_o^i \times \mathcal{I}_S^i \to \mathcal{O}_a^i \times \mathcal{O}_o^i \text{ if } i = n, \\
&\text{(ii)} \quad S_i: \ \mathcal{I}_o^i \times \mathcal{I}_S^i \times \mathcal{R}^i \to \mathcal{O}_a^i \times \mathcal{O}_o^i \times \mathcal{F}^i \text{ if } 1 < i < n, \\
&\text{(iii)} \quad S_i: \ \mathcal{I}_o^i \times \mathcal{R}^i \to \mathcal{O}_o^i \times \mathcal{F}^i \text{ if } i = 1.
\end{aligned} \tag{22}$$

The set of systems S_i, $1 \leq i \leq n$, is a hierarchical control structure if there exist two families of mappings $h_i : Y_i \to \mathcal{F}_{i+1}, 1 \leq i \leq n$ and $c_i : Y_i \to \mathcal{R}_{i-1}, 1 \leq i \leq n$, such that for each x in X and $y = S(x)$:

$$\begin{aligned}
&\text{(i)} \quad y_n = \ S_n(x_n, h_{n-1}(y_{n-1})), \\
&\text{(ii)} \quad y_i = \ S_i(x_i, c_{i+1}(y_{i+1}), h_{i-1}(y_{i-1})), \ 1 < i < n, \\
&\text{(iii)} \quad y_1 = \ S_1(x_1, c_2(y_2)).
\end{aligned} \tag{23}$$

Safety and Authority within Control Structure

The structure described in equations (19)–(23) represents a general description of input-output systems and their hierarchical relationships. Each sub-system, S_i, could simply contain a feedback control algorithm, for example PID[7]. However, in systems with sufficient complexity there often exists a set of decisions, and the control agent must select among alternatives. That is, the mapping $S_i : \mathcal{R}^i \to \mathcal{R}^{i-1}$ is a decision-making unit. In this type of hierarchy, there exists a family of decision problems $\mathscr{D}_i(\gamma_i)$, $\gamma_i \in \mathcal{R}^i$ and a transformation $\mathcal{K}_i$ such that for any input γ_i the output $\gamma_{i-1} = S_i(\gamma_i)$ is given as $\gamma_{i-1} = \mathcal{K}_i(x_i)$ where x_i is a solution of the decision problem $\mathscr{D}_i(\gamma_i)$. The inputs $\gamma_i \in \mathcal{R}^i$ from the decision-making unit immediately above act as a parameter in the decision problem of sub-system S_i. Alternatively, the outputs γ_{i-1} obtained from the transformation $\mathcal{K}_i$ are parameters for the lower level decision units. Such a decision-making hierarchy exists when S_i has "priority of action" or "control authority" over S_j.

[7] PID≡Proportional-Integral-Derivative, a form of feedback control widely and successfully used in industrial systems.

In the management literature, the hierarchical structure of an organization is described in terms of general decision making. For example, a decision making problem under conditions of uncertainty may be specified as a satisfaction problem by the 4-tuple $\left(g, \tau, X^f, \Omega\right)$: find a solution x in the feasible set X^f such that for all uncertainties ω in Ω:

$$g\left(X, \omega\right) \leq \tau\left(\omega\right) \tag{24}$$

where $\leq$ is a given relation, τ is a tolerance function, and g is an objective function.

In safety-driven design, the organization need not be defined in terms of *general* decisions. Rather, the safety-driven approach focuses on the identification and prevention of *unsafe* decisions; in the paradigm of hierarchical control, these unsafe decisions are defined more precisely as unsafe control actions. Building upon the formalism of unsafe control developed by Thomas [2013], an unsafe control action in the STAMP accident model can be expressed formally as a 4-tuple $\left(\mathcal{C}^i, T, \mathscr{C}, Co\right)$ where:

- $\mathcal{C}^i$ is the source controller that can issue control actions in the system. The controller may be automated or human.
- T is the type of control action. There are two possible types: Provided describes a control action that is issued by the controller while Not Provided describes a control action that is not issued.
- $\mathscr{C}$ is the control action (i.e. command) that is output by the controller.
- Co is the context in which the control action is or is not provided.

Section 3.3 further develops this formalism; the tuple provides a rigorous way to trace controller actions to hazards and vice versa. For the purpose of merely developing the model, an important omission[8] from the above 4-tuple is the *destination* of the control action. Thus, the definition is extended to include a destination$\mathscr{C}_i^j$, $i, j \in N$, which represents a control action from the j^{th} source to the i^{th} destination.

Without loss of generality, a three-layer "decision" hierarchy is then:

$$S_1 : \mathscr{W}_1 \times \mathscr{C}_1^3 \times \mathscr{C}_1^2 \rightarrow \mathscr{C}_1^1, \tag{25}$$

where $\mathscr{W}_1$ is feedback information from the *controlled process*. $\mathscr{C}_1^2$ and $\mathscr{C}_1^3$ provide constraints on the set of actions available to controller $\mathcal{C}^1$. Depending on the architecture of the control system, this level might receive constraints only from the immedi-

[8] An omission that Thomas acknowledges, and can be extended.

ate level above it. That is, $\mathscr{C}_i^j = \emptyset$ for $j - i \geq 2$ for completely stratified systems.

The second layer is represented by a mapping

$$S_2 : \mathscr{W}_2 \times \mathscr{C}_2^3 \rightarrow \mathscr{C}_1^2, \tag{26}$$

and $\mathscr{W}_2$ is feedback available to the second-level controller. At this level, the controller may not have direct access to hazard variable states $\mathcal{V}\left(\mathcal{C}^1\right)$ but may have additional information about the environment that the lower level controller does not have access to. The final layer of this three-layer hierarchy is

$$S_3 : \mathscr{W}_3 \rightarrow \mathscr{C}_1^3 \times \mathscr{C}_2^3. \tag{27}$$

At some point in the hierarchy, the system must have access to information about hazardous states. That is, $\mathscr{W}_i$ should provide information about the variables relevant to system hazards present in the controlled process of controller $\mathcal{C}^1$.

Consider a very simplified example in aircraft guidance and navigation. The flight crew (S_3) is informed of convective weather ($\mathscr{W}_3$) and inputs a new series of waypoints into the Flight Management System (S_2). The Flight Management System then sends commands to the aileron hydraulics (S_1). The local aileron control system uses sensor input about its position ($\mathscr{W}_1$) to adjust pressure in the hydraulics. The Flight Management System uses aircraft position data ($\mathscr{W}_2$) to update commands to the local aileron controller as well as to send position information to the Flight Crew. In this case $\mathscr{W}_2 \subset \mathscr{W}_1$.

In the above aircraft example it is apparent that many more components[9] are required to direct the aircraft in the proper direction. These lower-level control components are highly coupled, and the guidance of an aircraft depends on the simultaneous manipulation of many variables $\mathcal{V}_m$. This relationship suggests another dimension of decomposition that must be accounted for when identifying and synthesizing a control structure model.

Other Vertical Relationships

The level of a given controller may not be obvious at this early stage of concept development. That is, decision-making priority may not be evident, or may not have even been defined yet, in an early concept of operations. Other notions of vertical

[9] Ailerons, elevators, rudders, spoilers, thrust, and many others are required to control an aircraft trajectory.

decomposition are described here in order to guide this process of identifying the appropriate level of a control agent.

Decision Complexity represents another type of decomposition in control hierarchies. Increasingly complex decisions tend to lack well-defined and complete specification of uncertainties, input conditions, problem constraints, and processes involved in transforming input conditions into desired output states [Simon, 1977]. Such complex decisions often require selection among multiple alternatives and involve interrelated factors with time dependence and nonlinearities such as feedback lag, delayed effects, singularities, tipping points [Forrester, 1987; Sterman, 1994]. Decision complexity in systems theory often implies that a decision at one level involves processing, understanding, and coordinating decisions at a lower level (see equation 23).

Time scale constitutes another form of vertical decomposition. For example, an aileron control (sub)system in an aircraft measures and adjusts commands on the order of fractions of seconds; a pilot (in a highly automated civilian aircraft) might be adjusting flight plans or profiles on the order of minutes or hours. In this case there is already a built-in priority of action in most aircraft systems, but these timing differences serve as another indication of a hierarchical decomposition. A heuristic for vertical decomposition of control agents is then

$$\left\{ S_i = f\left(t_i\right), S_j = f\left(t_j\right) \middle| \, t_i \gg t_j \right\} \implies i > j \tag{28}$$

where $S = f(t)$ represents a control output as a function of time interval, t. The following notions describe additional aspects of control hierarchy.

1. Higher level units are concerned with larger portions of the system, which can be modeled using the STAMP notion of a process model, along with the concepts of aggregation and set theory.

$$\left\{ \exists \rho^i \in \mathcal{C}^i, \rho^j \in \mathcal{C}^j \middle| \left(\rho^i \supset \rho^j\right) \vee \left(\rho^j \in \rho^i\right) \right\} \implies i > j \tag{29}$$

   That is, if a process model of one controller (i) either is a superset of, or contains, the process model of another controller (i) then controller i is supremal to controller j. If aspects of ρ^i are in ρ^j and vice versa, but the superset condition does not hold, then the relationship is most likely horizontal and not vertical. The next sub-section describes horizontal decomposition.

2. Higher level units are not only concerned with the slower aspects of the systems'

operation (see equation 28) but also: the exchange with the environment takes place at a lower frequency, the dynamics of concern is slower, and the period between decision time is longer.

$$\left.\begin{array}{c} \lambda_i\left(f\left(x_i\right)\right) \ll \lambda_j\left(f\left(x_j\right)\right) \\ \lambda_i\left(\mathcal{D}_i\right) \ll \lambda_j\left(\mathcal{D}_j\right) \\ T\left(\mathscr{C}^i\right) \gg T\left(\mathscr{C}^j\right) \end{array}\right\} \Longrightarrow i > j \tag{30}$$

where $\lambda_k\left(f\left(x_k\right)\right)$ is a fundamental mode of the dynamics of subsystem S_k, $\lambda_k\left(\mathcal{D}_k\right)$ is a fundamental mode of the disturbances to subsystem S_k, and $T\left(\mathscr{C}^k\right)$ is the time between control actions associated with S_k.

3. *Abstraction hierarchies* [Rasmussen, 1986] decompose the system in terms of level of description. A control agent may be concerned with functional purpose, abstract function, generalized function, physical function, or physical form. The Functional Purpose level describes the goals and purposes of the system, and systems typically include more than one system goal such that the goals conflict or complement each other. The relationships between the goals indicate potential trade-offs and constraints within the work domain of the system. For example, the goals of a flight planner might be to achieve a desired route while trading off between flight time versus fuel consumption.

   The Abstract Function level describes the underlying laws and principles that govern the goals of the system. These are typically empirical or theoretical laws in an engineered system, but economic or judicial principles underlie a social system. Aircraft flight is governed by laws related to thrust, lift, and drag.

   The Generalized Function level explains the processes involved in the laws and principles found at the Abstract Function level, i.e. how each abstract function is achieved. Causal relationships exist between the elements found at the Generalized Function level. To generate thrust, a turbofan uses fuel injection and intake air, which has implications at the Functional Purpose level.

   The Physical Function level reveals the physical components or equipment associated with the processes identified at the Generalized Function level. The capabilities and limitations of the components such as maximum capacity have implications all the way up the hierarchy.

The Physical Form level describes the condition, location, and physical appearance of the components shown at the PFn level. In the aircraft example, the wings, turbofans, and fuselage are arranged in a specific manner, basically illustrating the location of the components.

Horizontal Decomposition

Developing a systems-theoretic model of a concept lies primarily in identifying the vertical relationships described above. However, within any level of control, S_i, there may exist a number of individual control agents, controlled processes, and other entities. The starting point is to recognize vertical position of the units according decision-making priority, abstraction, or other types of vertical relationships. Decomposition within one level of a hierarchy can then be done in terms of part-whole abstraction [Rasmussen, 1986; Leveson, 2000b] or echelons [Mesarovic et al., 1970].

Part-whole abstractions involve refinement and its opposite, aggregation. An intuitive description of aggregation is as follows. Suppose that ρ_1 is a mathematical description of a physical system using a given set of variables, and ρ_2 is a consistent description of the same system using a smaller set of variables. Then ρ_2 is termed an aggregate model for ρ_1, and the variables of the system ρ_2, are termed aggregate variables. Any of the variables within the refined model ρ_2 can then be said to have *horizontal* relationships in the control hierarchy. The following section explores some consistency properties related to aggregation and horizontal decomposition.

Another way to reason about horizontal relationships is to consider span of control. In an organizational hierarchy (also called a "Multi-echelon hierarchy"), any agents under the same span of control will have horizontal relationships. Building on the previous formalization of hierarchical control structures, and using the notation of Mesarovic and Takahara [1975], the following development formalizes the horizontal relationships among control agents. If C is a (finite) family of sub-systems S_i, $i \in N$, where N is a finite set, and if $>$ is a strict partial ordering of N, then $(C, >)$ is a hierarchy of systems. If $(C, >)$ is a hierarchy of control systems, and the ordering $>$ is such that $i > j$ iff S_i has priority of action over S_j, then $(C, >)$ is a control structure hierarchy.

Echelons in a control structure hierarchy $(C, >)$ are recognized in terms of the ordering $>$, representing priority of action. The first echelon units are the minimal units of C; the family

$$C^1 = \{S_i \mid_i \in N_1\}$$

is the first echelon, where

$$N_1 = \{i \mid i \text{ is a minimal element of } N\}.$$

The i^{th} echelon units are the minimal units of C when all lower echelons are omitted; the family

$$C^i = \{S_k \mid k \in N_i\}$$

is the i^{th} echelon, where

$$N_i = \{k \mid k \text{ is a minimal element of } N - [N_1 \cup N_2 \cup \cdots \cup N_{i-1}]\}.$$

Finally, define multi-echelon control structures as a subclass of general control hierarchies. A hierarchy of control systems $(C, >)$ is a multi-echelon control structure if, $\forall i, j \in N$, there is at most a unique $k \in N$ such that $\forall l \in N$,

$$l > i \quad \text{and} \quad l > j \quad \Longrightarrow \quad l > k.$$

This condition implies that any member of C has at most one unit of the immediately higher echelon which has control authority over it. In other words, a strict multi-echelon control structure is a "pyramid" structure, which is rare in real systems. That is, there is almost always overlap in control authority, often for good reason. In fact, there is often a trade-off between the simplicity of having the strict multi-echelon property and the availability of multiple controllers. The following section describes methods for assigning control responsibility and for identifying issues with coordination and consistency among multiple controllers. Regardless of this trade-off, systematically identifying the span of control provides another way of identifying horizontal relationships. If multiple entities (control agents) respond to the action of the same C^i (that is, they are under the span of control of the same controller) then they are on the same level of the hierarchy and have horizontal relationships.

Continuing the aircraft example, the FMS tries to coordinate between many processes in order to achieve some flight path objective. These lower-level processes and associated control systems are dynamically coupled and operate in parallel, for example adjusting thrust while manipulating (multiple) control surfaces to achieve a smooth turn or climb. In the case of an aircraft, the control structure is not a strict multi-echelon hierarchy, because lower level control systems must respond to either

FMS signals, manual pilot inputs, or both.

3.3 Systems-Theoretic Analysis of Model

Much of the control- and systems-theoretical foundation used to develop the model of the concept is also used to guide the analysis of the model. However, the focus shifts from modeling to identifying potential causal factors and invalid assumptions. According to Leveson [2012], there are several fundamental vulnerabilities in a hierarchical system. "At each level of the hierarchical control structure, inadequate control may result from missing constraints (unassigned responsibility for safety), inadequate safety control commands, commands that were not executed correctly at a lower level, or inadequately communicated or processed feedback about constraint enforcement" [p.81].

The control-theoretic approach emphasizes the importance of process models in enforcing adequate control: a process model must contain "the required relationship among the system variables (the control laws), the current state (the current values of the system variables), and the ways the process can change state" [Leveson, 2012, p.87], or the dynamics of the process. The four fundamental requirements of process control (see 1.a-d below) described in the previous sub-section must also be satisfied.

Once the control model of the ConOps has been built (previous sub-section), the hazardous scenarios and causal factors can be identified using these systems-theoretic views of accident causality. Specifically, the analysts, engineers, and stakeholders should ask:

1. Are the control loops complete? That is, does each control loop satisfy a Goal Condition, Action Condition, Model Condition, and Observability Condition?

    (a) Goal Condition – what are the goal conditions? How can the goals violate safety constraints and safety responsibilities?

    (b) Action Condition – how does the controller affect the state of the system? Are the actuators adequate or appropriate given the process dynamics?

    (c) Model Condition – what states of the process must the controller ascertain? How are those states related or coupled dynamically? How does the process evolve?

    (d) Observability Condition – how does the controller ascertain the state of the

system? Are the sensors adequate or appropriate given the process dynamics?

2. Are the system-level safety responsibilities accounted for?
3. Do control agent responsibilities conflict with safety responsibilities?
4. Do multiple control agents have the same safety responsibility(ies)?
5. Do multiple control agents have or require process model(s) of the same process(es)?
6. Is a control agent responsible for multiple processes? If so, how are the process dynamics (de)coupled?

As in the previous section, these questions can be formalized, and further description is provided with the following formalization. Question 1 relates to completeness of the individual control loops, questions 2-3 relate to assigning safety-related responsibilities to various control agents, and questions 4-6 relate to coordination of multiple control agents. The analysis therefore proceeds through three basic areas, which are explored in the following subsections and depicted in the bottom left of Figure 12.

3.3.1 Completeness Criteria for Individual Control Loops

Completeness criteria for process control systems have been developed elsewhere [e.g. Leveson, 2000a]. While existing specification languages[10] are formal and executable, it is often not desirable—and perhaps not possible—to specify an entire system formally. The purpose of the formalism here, during concept development, is not necessarily intended to support simulation but rather to provide a rigorous means for identifying gaps in the control loops. Because this thesis is intended to support safety-driven design and development, completeness criteria are explicitly linked to system hazards.

In addition to the original descriptions in Figure 11 on page 61, the control loop in Figure 13 assigns the variables and mappings presented in the formalism of the previous section.

Every system has a set of hazards, which are undesirable states of the system[11]. Define the set of states $\mathcal{V} \subseteq \mathcal{X}$, where $\mathcal{X}$ is the entire set of system state variables and

[10] There are many examples, including SpecTRM-RL [Leveson, 2000a], Statecharts [Harel, 1987], Prototype Verification System [Owre et al., 1996], AsmL [Barnett and Schulte, 2001], etc.

[11] Section 3.4 has a more complete definition of a hazard, but the above definition suffices for the current development.

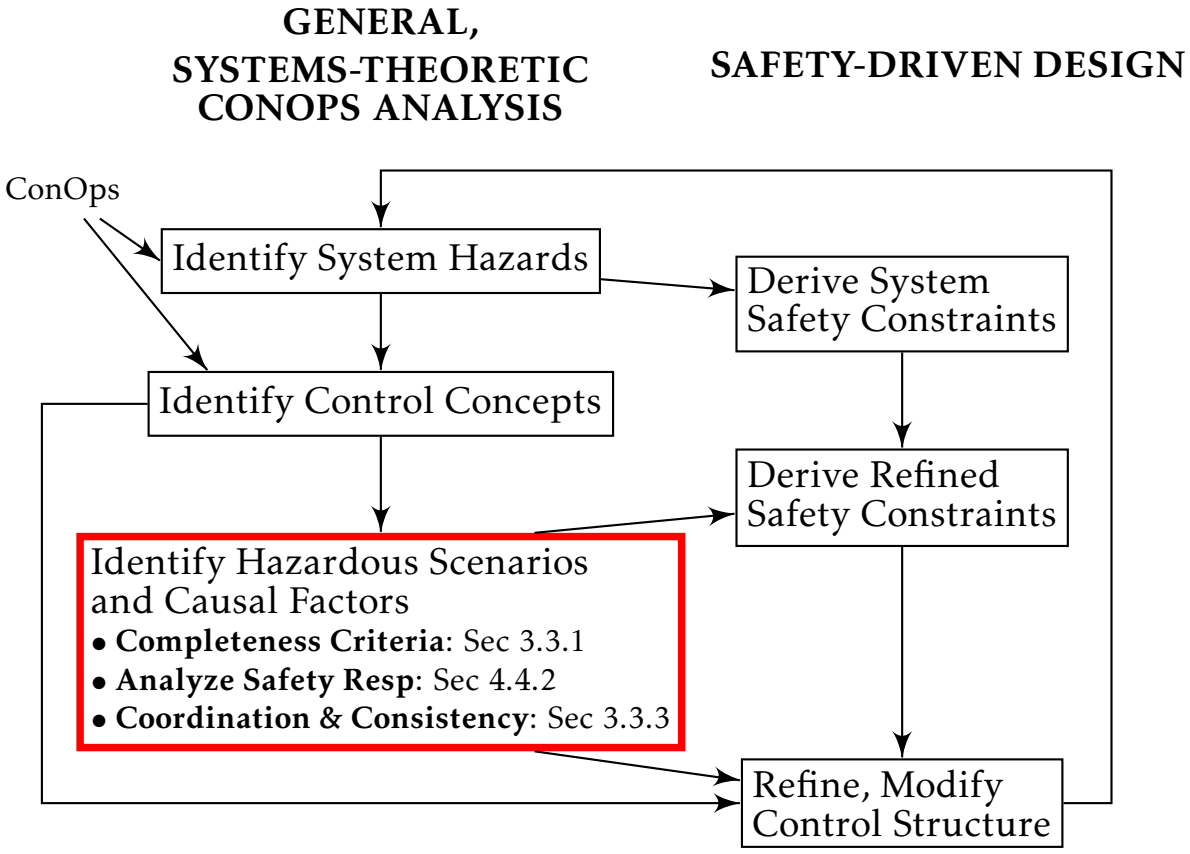


Figure 12: Proposed Methodology—Analysis

$\mathcal{V}$ is a set of variables associated with hazard $\mathcal{H}$. For example, loss of separation (LOS) between aircraft occurs when they violate some minimum separation distance. Thus, $\mathcal{H}_{\text{LOS}}$ consists of the states, $\mathcal{V} = \{x_1, y_1, h_1, x_2, y_2, h_2\}$, which are the current position of aircraft$_1$ and aircraft$_2$ in three dimensions each.

The control-theoretic approach to safety also assumes that processes are dynamic and can evolve over time or abruptly change. State dynamics typically take the form $\dot{\mathcal{X}} = f(\mathcal{X}, \mathcal{U}, \mathcal{D})$ or $\mathcal{X}_{k+1} = f(\mathcal{X}_k, \mathcal{U}_k, \mathcal{D}_k)$. Recall that, for higher level control agents, the controlled process itself could be a control agent and these dynamics may not be continuous. In the following formalism, process dynamics take the form $\mathcal{X} \times \mathcal{U} \times \mathcal{D} \mapsto \mathcal{X}$.

1. *Goal Condition*—the goal condition relates to preventing, and recovering from, hazardous states. In safety-driven design, the goal condition should seek, in order of decreasing priority, to (1) eliminate hazards, (2) avoid hazards, and (3) recover from hazards. That is,

$$\forall \mathcal{V} \in \mathcal{H}, \exists G, \mathcal{O}_a [G \rightarrow \mathcal{O}_a \wedge \mathcal{O}_a \implies \neg \mathcal{V}]$$

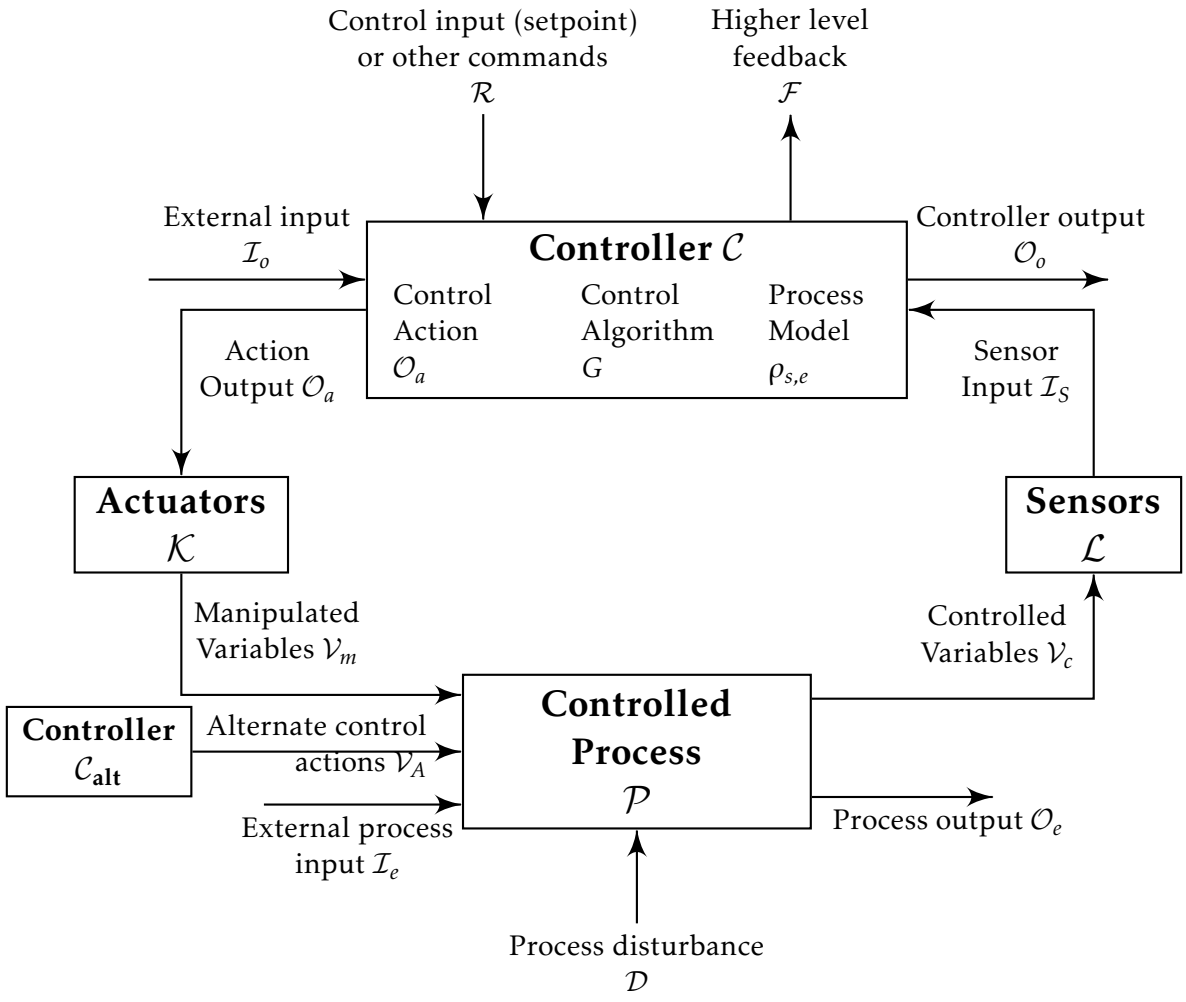


Figure 13: Generic Process Control Loop

and the negation of $\mathcal{V}$ can take several forms. Hazard elimination implies that the hazard does not exist ($\neg\mathcal{V} \implies \mathcal{H} = \emptyset$), hazard reduction or avoidance implies that the system state will never become a hazardous state($\neg\mathcal{V} \implies \mathcal{O}_a \times x \mapsto x' | x, x' \notin \mathcal{H}$), and hazard control or recovery seeks to minimize the amount of exposure to the hazard ($\forall x \in \mathcal{H}, \exists \mathcal{O}_a [\mathcal{O}_a \times x \mapsto x', x' \notin \mathcal{H}]$, that minimizes the time from $x \to x'$).

2. *Action Condition*—for every hazardous state variable, there is a signal that can manipulate that state, causing it to change. Furthermore, the actuator maps the controller output into the signal that can manipulate (hazardous) process states.

$$\forall \mathcal{V} \in \mathcal{H}, \exists \mathcal{K} [\mathcal{O}_a \times \mathcal{K} \mapsto \mathcal{V}_\mathcal{K} \wedge \mathcal{V}_\mathcal{K} \times \mathcal{V} \mapsto \mathcal{V}'], \tag{31}$$

where $\mathcal{X}'$ represents some change in state from $\mathcal{X}$.

3. *Model Condition*—the model condition asserts that the control agent must have a model for how the system evolves $\rho = [\mathcal{I}, \mathcal{V} \,|\mathcal{I} \times \mathcal{V} \mapsto \mathcal{V}']$, where $\mathcal{I} = \{\mathcal{C}, \mathcal{D}, \mathcal{E}\}$ are the control inputs, environmental disturbances, and other inputs from within the system, respectively. $\mathcal{V}'$ represents the evolution of the potentially hazardous state variables due to inputs and internal dynamics. Rather than consider every state in the system, in safety driven design emphasis is given to $\mathcal{V} \in \mathcal{H}$, the evolution of process variables that may lead to a hazard.

4. *Observability Condition*— there is a signal that measures the states v, and the signal can also help discern the evolution from V to V′ due to action A

$$\forall \mathcal{V} \in \mathcal{H}, \exists \mathcal{L}\,[\mathcal{O}_a \times \mathcal{V} \times \mathcal{L} \mapsto \mathcal{I}_{\mathcal{L}} \vee \mathcal{V} \mapsto \mathcal{V}' \implies \mathcal{I}_{\mathcal{L}} \times \rho \mapsto \rho'],$$

where ρ' represents some update of the model ρ. The above definition of observability not only asserts that the signal must be updated for every change in the (hazardous) process states, the signal must also explicitly discern the change (or lack of change) due to action, $\mathcal{O}_a$.

3.3.2 Analyzing Safety-Related Responsibilities

This part of the analysis is intended to ensure that all hazards and safety constraints are accounted for in the hierarchical control structure and to identify goals and responsibilities that conflict with safety constraints.

Given a set of system hazards, $H_i \in \mathcal{H}$, then a safety constraint represents control over system behavior that prevents the hazardous states from occurring. That is,

$$\exists \sigma_i \in \Sigma, \sigma_i \implies \neg H_i. \tag{32}$$

where Σ is a set of constraints, and σ_i is a *safety constraint*. There are two fundamental hazardous scenarios associated with system hazards and safety constraints. The first is that a safety constraint is unaccounted for. The second basic type of scenario is when the enforcement of a particular safety constraint (or general system goal) can cause a different hazard. In the framework of a hierarchical control structure, safety constraints and system goals are enforced via control actions, $\mathscr{C}$.

Let $\mathsf{P}(x, \mathbb{S})$ be defined for all pairs $(x, \mathbb{S})$, where

$$\mathsf{P}(x, \mathbb{S}) \equiv x \text{ results in } \mathbb{S}. \tag{33}$$

The predicate $\mathsf{P}(\mathsf{x}, \mathbb{S})$ is true, iff $\mathbb{S}$ is a defined condition or system state, and x is an action resulting in that condition.

A control structure that enforces every safety constraint is one that has available actions that result in equation (32). That is,

$$\forall \sigma_i \in \Sigma, \exists \mathsf{c} \in \mathscr{C} \left[\mathsf{P}(\mathsf{c}, \sigma_i)\right], \tag{34}$$

which states that, for every safety constraint there exists at least one available control action that causes the constraint to be true. A control structure with gaps is defined as a system that does not satisfy equation (34) and does not have appropriate control actions available to the various control agents.

Alternatively, there may be actions that conflict with safety constraints and actually cause the undesired hazard.

$$(\forall H_i \in \mathcal{H})(\neg \exists \mathsf{c} \in \mathscr{C})\left[\mathsf{P}(\mathsf{c}, H_i) \wedge \mathsf{P}(\mathsf{c}, \mathcal{G})\right] \tag{35}$$

where $\mathcal{G}$ is a system goal state or system hazard such that $\mathcal{G} = H_j, i \neq j$. A system that does satisfy equation (35) is not necessarily a bad design or inherently unsafe. Rather, the identification of gaps (systems that do *not* satisfy equation 34) and conflicts (systems that do *not* satisfy equation 35) is intended to simply flag potential hazardous scenarios and causal factors.

These scenarios identify areas in the concept where architectural decisions, future design decisions, and refinement of safety-related and non-safety related requirements should be considered with great care. In fact, this analysis brings to bear design decisions and requirements that previously have not been identified as safety-related, as the example in Chapter 4 demonstrates. Thomas [2013] has developed a formal definition for hazardous control actions and a means for identifying them using STPA. Refer to Appendix C for the formal definition of unsafe control actions and hazards. The preceding development assumes that hazardous control actions, which represent violations of safety constraints, can be identified using STPA. Identifying hazardous control actions becomes increasingly powerful as more design detail becomes available.

3.3.3 Coordination and Consistency

It is important to ensure that all safety responsibilities are accounted for, and often the system design results in either (a) one entity being responsible for enforcing multiple safety constraints or (b) multiple entities being responsible for enforcing the same safety constraints. Combinations of both do exist, especially in sufficiently complex systems.

Much theoretical work in systems theory has been dedicated to hierarchical control strategies related to problem (a) above. That is, much of the work attempts to ensure that there exists some policy guaranteeing that lower level control agents will achieve their individual objectives simultaneously with the higher level objectives [Mesarovic and Takahara, 1975]. The process control literature and other fields related to control theory have developed both the practice and theory of decomposing systems so that some coordination principles exist between the relatively de-coupled processes [e.g. Acar and Ozguner, 1989; Morari and Stephanopoulos, 1980; Zheng et al., 1999; Skogestad, 2004; Tatjewski, 2008].

This work neglects the latter problem—problem (b) above—where control decisions come from multiple sources and actuate on the same process variables. Often in process and chemical control, control structure designers are able to decouple the system sufficiently so that there is no overlap in responsibility. This is often not possible or not desirable in many complex socio-technical systems and/or systems with a high degree of dynamic coupling.

Safety-driven design is concerned not only with ensuring that coordination principles exist within the control structure but also in coordinating scenarios where multiple control agents have responsibility over the same process(es). Cowlagi and Saleh [2013] have suggested an approach and research direction for hazard analysis that builds upon some of the systems-theoretic concepts developed byMesarovic [1970], as well as the approach to hazard analysis used here and originally proposed by Leveson [2004]. This thesis builds on that work but also includes the so-called "Multiple Controller" problem [Ishimatsu et al., 2010] in a formal way.

The first principle of safety-driven design relates to coordination of multiple controllers, which asserts that there must be some priority of action, or "leader", if processes (or control agents) can be manipulated by more than one source.

Define the priority of action function $\mathscr{P}(\cdot,\cdot)$, as follows:

$$\mathscr{P}(c,d) \iff c > d, \tag{36}$$

where the inequality implies that the action of c takes priority over action d. Conversely, if d takes priority over c or if there is no priority, then $\mathscr{P}$ evaluates to false. That is

$$\neg\mathscr{P}(c,d) \iff d > c \vee c = d. \tag{37}$$

Priority of action must also satisfy a transitive property,

$$\mathscr{P}(a,b), \mathscr{P}(b,c) \implies \mathscr{P}(a,c). \tag{38}$$

Now define the action process predicate, $\boldsymbol{A}(\mathrm{c},\mathcal{V})$ iff $\mathrm{c} \times \mathcal{V} \to \mathcal{V}'$. That is, $\boldsymbol{A}$ is true whenever the action c can cause a change in state(s) from $\mathcal{V}$ to $\mathcal{V}'$. The coordination principle for two controllers (without loss of generality as long as transitive closure holds) is then,

$$(\forall \mathrm{c} \in \mathscr{C}_i)\Big(\forall \mathrm{d} \in \mathscr{C}_j\Big)\exists\,(\mathscr{P}(\mathrm{c},\mathrm{d}) \vee \mathscr{P}(\mathrm{d},\mathrm{c}))\Big[\mathsf{A}\Big(\mathrm{c},\mathcal{V}_p\Big) \wedge \mathsf{A}\Big(\mathrm{d},\mathcal{V}_p\Big)\Big], \tag{39}$$

where $\mathcal{V}_p$ is some system state, and actions c and d are generated from the i and j control agents, respectively. Equation (39) also enforces a control hierarchy. While the development and modeling effort may have identified control agents within the same level of the hierarchy, priority of action must be given if these control agents have overlapping control authority.

The second principle of safety-driven design is related to consistency of action. Any two controllers with safety-responsibilities related to the same process variables must ensure certain consistency characteristics. These controllers must have a consistent understanding or model of the current state, other actions that can affect that state, and how the state will evolve from those actions. The consistency principle for two controllers is:

$$\Big(\forall v \in \mathcal{V}, \forall \mathrm{c} \in \mathscr{C}_i, \forall \mathrm{d} \in \mathscr{C}_j \,\Big|\, \mathsf{A}(\mathrm{c},v) \wedge \mathsf{A}(\mathrm{d},v)\Big)\Big[\rho_i(a,v) \equiv \rho_j(a,v) \wedge G_i \equiv G_j\Big], \tag{40}$$

where $\rho_k(a,v)$ is the k^{th} control agent's model of state dynamics, v, subject to inputs, $a = \{\mathrm{c},\mathrm{d},\epsilon\}$. The latter aspect of this set, ϵ, represents "other" inputs to the process.

See equations (11) and (12) on page 66 for more detailed description of process inputs, which include but are not limited to environmental disturbances. G_k defines the goal condition of the k^{th} controller. The equivalence properties in equation (40) can be described using a property of aggregation called *dynamic exactness*. Suppose that the model ρ_1 is described by the state equation

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) \tag{41}$$

and the model ρ_2 is described by

$$\dot{\hat{\boldsymbol{x}}}(t) = M\hat{\boldsymbol{x}}(t) + N\boldsymbol{u}(t) \tag{42}$$

In order for ρ_2 to be an aggregated model of ρ_1, it is required that

$$\hat{\boldsymbol{x}}(t) = C\boldsymbol{x}(t) \tag{43}$$

for all t. This requirement is termed dynamic exactness. It is easy to see that dynamic exactness is achieved iff the matrix equations,

$$MC = CA \tag{44}$$

$$N = CB, \tag{45}$$

are satisfied.

The equivalence property in equation (40) is relatively straightforward for many computer systems, where input-output behavior can be simulated and all relevant inputs are known. For human control agents, assuring—or even understanding—the consistency of mental models is more difficult, as is anticipating all possible input sequences for a complex computer system. As will be demonstrated in the following chapter, the coordination and consistency principles are intended to help the analysts identify potential scenarios that could arise due to independent sources of information, independent mathematical models, different goals or policies, and other factors. Such characteristics could be qualitative (such as for human operators or autonomous systems which are immature in terms of design detail) but could become increasingly quantitative (such as black box input-output models) as the concept matures.

As was the case with Analyzing Safety-Related Responsibilities (sub-section beginning on page 81), the system is not necessarily unsafe if conditions in equation (39)

and (40) do not hold. Rather, these conditions represent potentially hazardous aspects of a ConOps that require further inquiry and may need to be refined or modified in order to avoid potential conflicts.

The above description and formalism makes no distinction between deterministic or probabilistic signals or information, for example cases where a control or feedback signal only has some probability of reaching its intended destination. Because this analysis focuses on worst-case conditions and assumptions (see definitions in Table 9 on page 89), the analyst must reason about system behavior if/when the signal does not reach its destination. Such reasoning is with respect to (a) control conditions, (b) fulfillment of safety-related responsibilities, and (c) coordination and consistency of control agents.

The framework presented in this section is based on control- and systems theory. It allows an analyst to systematically, rigorously interpret and decipher a natural-language description of a concept using guidewords and a series of generic roles and questions. Then, using a systems-theoretic view of accident causality, the framework asks a series of questions about the model to ensure completeness and consistency and to identify areas where further investigation is necessary. Results using this approach show that it identifies many more types of scenarios and factors than traditional PHA approaches (see Chapters 4 and 6). Additionally, the approach helps analysts, engineers, and other stakeholders to identify and document more explicit and implicit assumptions about the concept.

3.4 Using STECA in Early Systems Engineering

The early phases of systems engineering involve identifying system objectives and criteria, defining top-level requirements, defining a system-level architecture, and then performing trade studies that ultimately lead to a design (see, for example, [Leveson, 2000b; de Weck et al., 2011; Kapurch, 2010; INCOSE, 2011] and Figure 2 on page 24). As Chapter 1 argued, safety engineering should be integrated into these activities, and thus what follows is a brief explanation of those specific safety-related activities and their resulting artifacts. Table 8 depicts the relationships between safety-driven design activities and their counterparts in general systems engineering.

Once the model and scenarios are identified, this information can be used to guide the system design. The methodology presented in Section 3.2 results in the definition of a control structure. This control structure should be representative of the descrip-

Table 8: General Systems Engineering and Safety-driven Design

General Systems Engineering	⇄	Safety-Driven Design
Identify System Objectives, Criteria	⇄	Identify Accidents and Hazards
Define Top-level Requirements	⇄	Define Top-Level Safety Constraints
Define a System Architecture	⇄	Define a Functional Control Structure

tions contained in the ConOps, and this control structure could help define part of a baseline architecture for the system.

However, the analysis in Section 3.3 will identify potentially hazardous scenarios and associated causal factors. These scenarios drive the development of safety-related requirements and constraints (as well as functional requirements). The hazardous scenarios will also result in a refined or modified control structure. Alternative control structures should attempt to eliminate or mitigate against the hazardous scenarios found using the techniques and theory in Section 3.3.

Figure 14 shows the basic aspects of systems-theoretic early concept analysis. The figure captures the fact that the control model generated from the ConOps, and the subsequent analysis, form the inputs to STECA.

What is particularly noteworthy about the process is that modifying the control structure may fundamentally change some of the assumptions about the ConOps. For this reason, there is an upward arrow shown in Figure 14. This arrow depicts the fact that safety-driven design is, like any design effort, much more iterative and nonlinear. Changing the control structure should eliminate or mitigate certain hazards, but it may introduce new hazards. The figure also references the relevant sections later in this thesis.

The two processes in the bottom right of Figure 14, then, form the basis of safety-driven design: (1) derivation of refined safety constraints and requirements and (2) generation of a control structure and potential alternatives. Again, the control-theoretic modeling effort and subsequent analysis serve as inputs to these two activities.

Table 9 contains the definitions used throughout the rest of the thesis.

The first safety-related activity involves identifying a set of accidents and high level hazards. While system-level objectives define what the system *should* do, system-level hazards define what the system should *not* do (see Table 9). An accident is simply a loss that stakeholders must avoid, and a hazard is defined as *"a system state or set of*

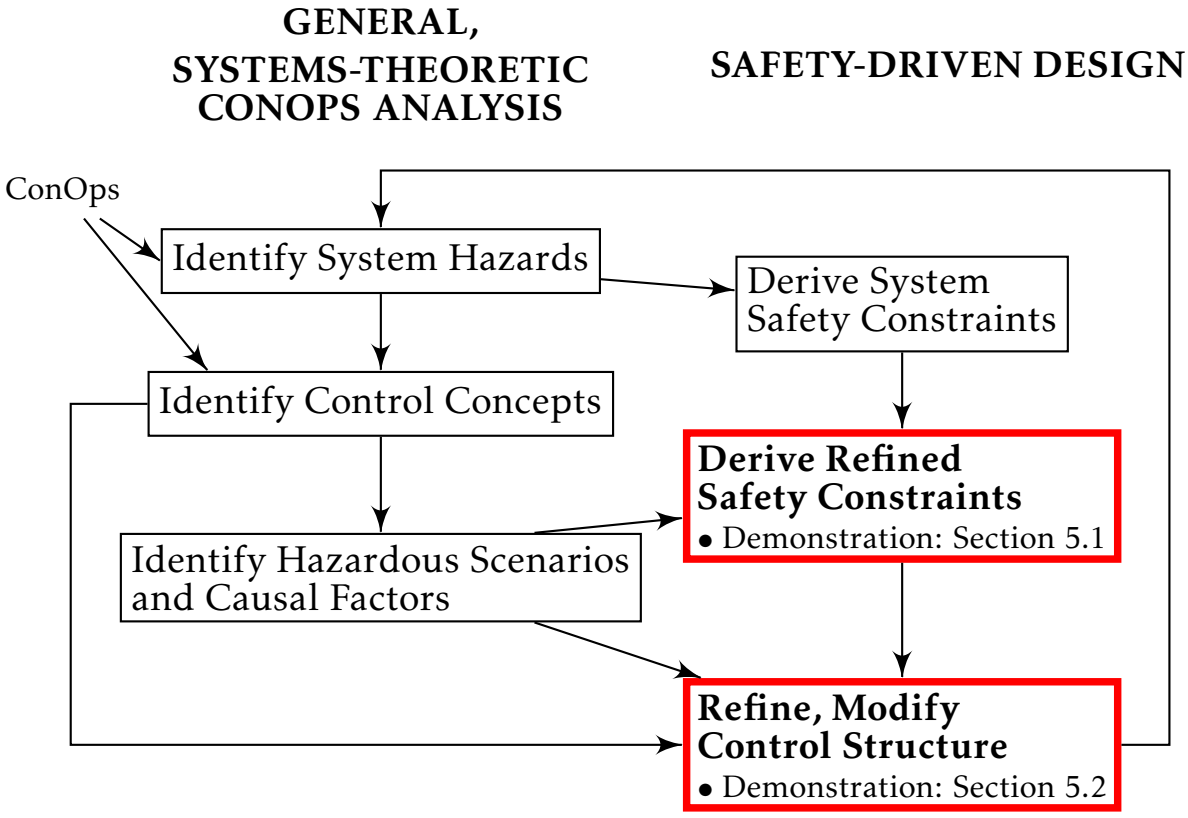


Figure 14: Proposed Methodology—STECA

conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident (loss)" [Leveson, 2012].

For example, accidents for a train system would include loss of human life or injury. Hazards for a simple automated train door include:

[H-1] Doors close on a person in the doorway

[H-2] Doors open when the train is moving or not in a station

[H-3] Passengers or staff are unable to exit during an emergency

Identifying Safety Constraints

In safety-driven design, requirements typically take the form of a safety constraint. These constraints allow, control, or restrict the behavior of components as well as their interactions.

For the train door hazards listed above, the associated safety constraints would be:

[SC-1] Doors must not close when a person is in the doorway

[SC-2] Doors must not open when the train is moving or not in a station

[SC-3] Train must allow passengers/staff to exit during an emergency

Table 9: Definition of Terms in Safety-Driven Design

Term	Definition
Accidents	undesired event that results in a loss, including human life or injury, damage to property, environmental pollution, etc.
Hazard	a system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident (loss)
Safety Constraint	limitation or restriction on system behavior. In safety-driven design the constraint should prevent hazardous behavior from occurring
Scenario	set of events, actions, or behaviors that can lead to a hazardous state (includes requirements flaws, inappropriate human behavior, dysfunctional interactions among components, and component failures)
Causal Factors	lower level or refined events, actions, or behaviors that contribute to a scenario
Control Structure	hierarchical structure where each level imposes constraints on the activity of the level beneath it
STECA	systems-theoretic early concept analysis, an analytical tool based on the STAMP accident model that can be used in the early systems engineering process

The safety constraints are simply written in such a way that prevents the associated hazard from occurring. Much like the system-level safety constraints depend on identifying the appropriate system-level hazards, identifying lower-level safety constraints depends on identifying lower-level hazardous scenarios and causal factors.

That is, just as the system-level safety constraints are written to prevent the occurrence of a hazard, so are the lower level constraints written to prevent the occurrence of hazardous scenarios and associated causal factors. The safety-related constraints and requirements will only be complete to the extent that the analysis used to identify causal factors is complete. Developing refined requirements and constraints, then, is made possible by rigorously defining the control structure (via methods described in Section 3.2) and then identifying potentially hazardous scenarios (via methods described in Section 3.3).

For example, a hazardous scenario might arise if the system does not satisfy Completeness Criteria for Individual Control Loops (section 3.3.1). The train door may close on a passenger if it cannot sense the presence of passengers in the doorway. In this case, the Observability Condition is unsatisfied, and refined requirements related to the first hazard involve sensing and detecting the presence of passengers in the

doorway.

Another issue might arise if the train door can be controlled by both the train conductor and automated systems. The conductor and automation might issue conflicting "Close Door" commands and violate Coordination and Consistency postulates (section 3.3.3). Refined requirements would then constrain the behavior of the door whenever multiple agents issue simultaneous commands.

The following chapters present an example of the approach used to identify scenarios, applied to an important concept being developed for the next generation of air traffic management. Chapter 5 then uses those results to systematically identify requirements and constraints that will prevent these scenarios from occurring.

Generating the Control Structure

Developing the control structure takes the following steps. The first step is developing a control model from the ConOps, using the theory and process described in Section 3.2. That is, the theory and process developed in Section 3.2 results in a model that describes the *structure* of the system and the *relationships* between components.

These types of information are consistent with a general definition of architecture given in Section 2.1. However, a control structure provides additional information about the relationships between components. These relationships are based on systems theory, where the control structure is represented by a hierarchical structure and each level imposes constraints on the activity of the level beneath it. Therefore, a control structure contains not only general information exchanges but also hierarchical and authority relations, control roles, and safety-related responsibilities.

For example, a general[12] architectural relationship might state that Component X exchanges data Y with Component Z. However, in a control structure there is additional information, which might state that Component X enforces constraints on Component Z via actuator signal Y. Component X is therefore at a higher level in the system hierarchy.

The theory in Section 3.2 and demonstration in the following chapter represent a process for developing the *initial* system control structure. However, this initial control structure is based on an informal ConOps that may have flaws or missing information, and thus the control structure itself may have flaws. That is, because the analysis in Chapter 4 focuses on identifying causal factors and missing information, the control models also contain missing or potentially hazardous relationships. As will

[12]See the definitions of a general architecture in Section 2.1, page 23.

be shown shortly, these causal factors can be used to further develop and/or modify the system control structure.

The resulting control model is then refined using the scenarios and causal factors identifying using the theory in Section 3.3. These scenarios and causal factors can be used in at least two ways to refine or modify the system control model.

The first way to refine the control structure is by developing requirements and additional constraints on component behavior. Just as the system safety constraints are worded to prevent the occurrence of system-level hazards, the refined safety constraints are intended to prevent the occurrence of lower level scenarios and causal factors.

The previous section described safety requirements and constraints, which is important in and of itself. The constraints also help define the system control structure. For example, if the analysis identifies a missing feedback link from a lower level component to a higher level component, then an additional requirement would add the feedback. This additional requirement (or constraint) represents an update to the control structure.

The second way to use, and hopefully prevent, the hazardous scenarios and causal factors is to change the control structure itself. Recall that a hierarchical control system consists of control agents, controlled processes, constraints going down the hierarchy in the form of control actions, and information going up the hierarchy in the form of feedback (see Figure 8 on page 57). A modification of the control structure, then, consists of modifying the order of the hierarchy itself (i.e. the authority that one component has over another), the available control actions, and the available feedback.

Consider again the train door example, where the train door can be controlled by both the train conductor and automated door controller. The previous section described a potential safety constraint regarding order of priority if there is a conflict in commands between the two door control agents.

That example, and the ensuing requirements, assume some existing control structure (that both the conductor and autonomous system can open or close the door). Based on this control structure, there exist several possible requirements to deal with the case of conflicting commands. Each of these requirements has a tradeoff.

An alternative approach is to *modify the control structure itself*. In the train example, an alternative structure might disallow simultaneous door commands altogether. One control structure would eliminate manual control via the train conductor entirely and

use only the automated system (or vice versa).

Clearly, modifying the control structure also has tradeoffs. It is often the case, however, that a change in the control structure is simpler and more effective than the alternative, which is attempting to derive detailed safety requirements based on an existing structure (and its associated hazardous scenarios). Chapter 5 presents such an example in the air traffic control domain.

Summary

This chapter has described the theoretical underpinnings of a new approach to concept development and safety-driven design. Following this theoretical development, the chapter proceeds with the proposed approach.

The process consists of identifying the control concepts within a ConOps, generating an initial control model, and then interrogating that model in order to identify hazardous scenarios. Once the initial control concepts and hazardous scenarios are identified, this information can then be used to generate safety-related requirements and constraints and to modify or refine the system control structure.

The following chapters describe the process, applied to an example in the air traffic control domain.

Chapter 4

Application of STECA Approach

> *At the heart of this vision* (NextGen) *is the concept of Trajectory-Based Operations.* [Toner, 2011]

Recall the goals of this research from the previous chapters. The first goal is to develop rigorous, systematic tools for the analysis of future concepts in order to identify hazardous scenarios and undocumented assumptions. The second goal is to extend these tools to assist stakeholders in the development of concepts using a safety-driven approach Both goals especially apply to systems where the tradespace includes human operation, automation or decision support tools, and the coordination of decision making agents.

Systems-theoretic early concept analysis (STECA) has been applied to the Trajectory-Based Operations (TBO) concept being developed by the United States Federal Aviation Administration as part of the NextGen air traffic management modernization program. After a brief description of this concept, the following sections present the application to the case study, showing how the objectives can be achieved in a systematic, rigorous fashion.

4.1 Trajectory-Based Operations

The United States air transportation is already under stress, and demand in aircraft operations is expected to increase significantly in the next decade and beyond [FAA, 2013]. In addition, there are growing concerns about air transportation's effect on the environment and national security. It is assumed in the aerospace community that current technologies and procedures in the national airspace cannot meet these increasing demands; therefore, the United States is creating the Next Generation Air Transportation System (NextGen) air traffic management modernization program. The goals of NextGen are to expand capacity, ensure safety, protect the environment, and grant flexibility and equity to airspace users.

The TBO Concept of Operations proves to be a useful case study for this thesis. There are several reasons it is compelling as a case study, in addition to the technical

reasons to be described momentarily and their relationship to the literature gap identified in Chapter 2. First, TBO is an important real-world problem, and its successful implementation will have a significant impact on tomorrow's airspace. Past attempts at modernizing the national airspace have failed, in part due to the ineffectiveness or lack of tools necessary to develop new technologies and procedures [Cone, 2002; Office, 1986].

Second, a professional working group has conducted a preliminary hazard analysis of the TBO concept, and they used the TBO ConOps as the primary source of information for their analysis [JPDO, 2012]. The existence of results developed using the traditional approach provides a basis on which to compare the results of this research.

Trajectory-Based Operations (TBO) is a shift from the current Air Traffic Management (ATM) and control strategy of clearance-based operations, intended primarily to increase capacity and improve efficiency. Today's operations rely on relatively little automation, in comparison to the TBO framework where aircraft will follow four dimensional paths, called trajectories, which are computed by autonomous systems and decision support tools (DST) [JPDO, 2010]. When fully realized, these trajectories will represent an aircraft's gate-to-gate movement and will be the basis for Air Traffic Control (ATC) and Air Traffic Management (ATM) that focuses on traffic flow and airspace use and autonomy of individual aircraft. The primary themes of TBO are: moving from clearance-based to trajectory-based airspace management, increasing reliance on automation and decision support tools, and distributing traffic management responsibilities throughout the system.

A key term in TBO is the four dimensional trajectory, or 4DT, which defines the aircraft in 3-dimensional space and time and is described in the list below. TBO uses the 4DT "to both strategically manage and tactically control surface and airborne operations" [JPDO, 2011]. The 4DT represents not only the aircraft's current state but also its intent, or where it will be in the future in both space and time. There are several other key terms that are listed here for reference:

- 4DT — Four dimensional trajectory, defined laterally and longitudinally by latitude and longitude, vertically by altitude and with time. Surface movement is a 3DT—lateral, longitudinal, and time [JPDO, 2011].
- RNP — Required Navigation Performance, describes an aircraft's ability to follow a ground track and/or vertical profile to within some specified tolerance in nautical miles or feet (±TBD NM, ±TBD ')

- RTP — Required Time Performance, describes an aircraft's ability to reach a way-point within some specified window of time
- Conformance — Monitoring of the aircraft's position, altitude, and time performance against the agreed-upon 4DT. Monitoring is against performance requirements for the flight maneuver or surface movement. Conformance monitoring occurs both in the air and within ground automation. Alerts are generated if the aircraft is not meeting its 4DT performance [JPDO, 2011].

These definitions allude to the fact that TBO relies not only on accurate state information but also on accurate prediction and navigation capability. In addition to these important considerations, note again that one of the goals of TBO[1] is to grant increased autonomy, flexibility, and equity to both individual users and operation centers. All of these factors represent a major shift in (1) the types of information gathered and exchanged, (2) technologies used to generate and transmit this information, (3) the roles and authority of the various actors.

In addition to the major technological and administrative challenges facing the development and implementation of NextGen, there are general research objectives that this thesis is intended to address. Section 2.1 (p. 23) describes the informality and lack of rigor often present during concept generation, and Section 3.1 (p. 54) describes some of the challenges specific to generating and analyzing a Concept of Operations. The TBO Concept of Operations [JPDO, 2011] shares many of the problems typically found in a ConOps: lack of a specification or requirements, prevalence of undocumented or implicit assumptions, and description of the concept using informal text or graphics.

4.2 Analysis of TBO

The basic steps of the proposed approach described in Chapter 3 are shown again in Figure 15. This section demonstrates the two highlighted boxes in Figure 15, namely, the identification of hazards and safety constraints. This section also develops the general air traffic management control hierarchy, using concepts and theory from the previous chapter.

[1] This goal of TBO is also an important goal of NextGen in general.

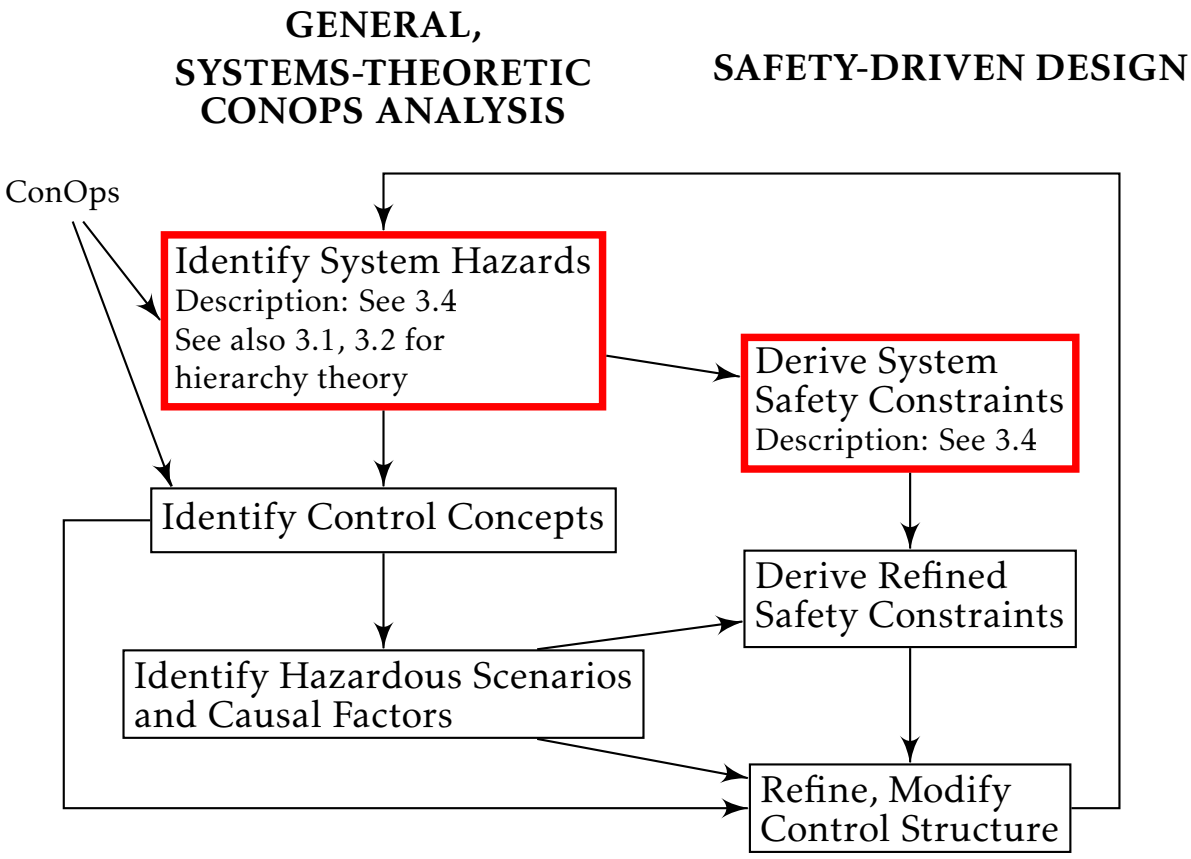


Figure 15: Methodology—Top-Level Systems Engineering

Top Level System and Safety Engineering

Table 8 on page 87 depicts the relationships between safety-driven design activities and their counterparts in general systems engineering. The first safety-related activity involves identifying a set of accidents and high level hazards (see Table 9 for definitions).

For example, in air traffic management, the accidents would be aircraft loss, equipment damage, and injury or loss of life. A few example hazards that could lead to these accidents are listed below. This list represents a sub-set of hazards used for the certification of TCAS [Leveson and Reese, 1994] and the analysis of several aeronautical applications [Fleming et al., 2013].

[H-1] Aircraft violate minimum separation (LOS or loss of separation, NMAC or Near midair collision)

[H-2] Aircraft enters uncontrolled state

[H-3] Aircraft performs controlled maneuver into ground (CFIT, controlled flight into terrain)

Using these hazards, then define a set of safety-related requirements (constraints) and responsibilities. The system-level safety constraints relate directly to the hazards; conforming to the safety constraints should prevent the associated hazard from occurring. The following constraints are based on the three hazards above.

[SC-1] Aircraft must remain at least TBD nautical miles apart en route[2] ↑ [H-1]

[SC-2] Aircraft position, velocity must remain within airframe manufacturer defined flight envelope ↑ [H-2]

[SC-3] Aircraft must maintain positive clearance with all terrain (This constraint does not include runways and taxiways) ↑ [H-3]

The next step involves defining a high-level system control structure. As described in Chapter 3, in STAMP the system is represented by a hierarchical, functional control structure. In a *Level of Decision Complexity* hierarchy, higher level responsibilities are concerned with slower aspects of the system operation, the disturbances take place at a lower frequency, and the dynamics of concern are slower.

Among the hazards and constraints listed above, the lowest level function involves aircraft attitude control ([H-2],[SC-2]). That is, control of the aircraft's speed, heading, and altitude occurs on the (relatively) shortest time scale, involves the least amount of uncertainty, and requires the least amount of information. This level of control occurs via changes in thrust and manipulation of aircraft control surfaces. The next level involves avoiding terrain ([H-3],[SC-3])—the aircraft must use its control systems, but there is added complexity associated with identifying both the terrain and potential changes in trajectory in order to avoid those obstacles. Not only are the control decisions more complex, but also this function occurs on a slower time scale than the manipulation of aircraft thrust and control surfaces. Finally, the top level of the hierarchy involves the separation of two or more aircraft ([H-1],[SC-1]). Separation of aircraft represents the highest level of complexity among the listed hazards. Separation assurance necessitates not only knowledge and prediction of multiple aircraft states but also the identification of, and selection among, multiple actions required to avoid a conflict.

The basic functions required to safely manage the airspace and prevent the above

[2] Similar constraints could be developed for other phases of flight.

hazards from occurring, then, are route planning[3] and piloting[4]. The route planning function must provide conflict-free trajectories and sequence the flow of aircraft, while the piloting function must navigate the aircraft along its assigned path. From a hierarchical, control-theoretic perspective, this example system takes the structure shown in Figure 16. The safety-related responsibilities are also stated in that figure. A hierarchy of route planning, guidance, and control is a typical decomposition in engineered systems (see for example Leonard et al. 2007).

These functions and responsibilities are intentionally *solution neutral* and are applicable to any air traffic management concept. That is, these functions are not (yet) assigned to a human or computer, aircraft or air traffic controller, or any combination of these and other potential solutions. While the TBO ConOps has made some assumptions about these assignments, the level of generality, neutrality, and abstraction in Figure 16 is important in systems engineering, particularly during the architecting phases.

These activities provide a reference and context upon which one can develop a model of the system using control- and systems-theoretic processes. In safety-driven design, the development of the model and subsequent analysis of the model should ultimately be traceable to system hazards and high level safety-related responsibilities and functions. Again, these activities and artifacts closely parallel general systems engineering.

In fact, identifying hazards and safety-related responsibilities should be done in conjunction with identifying system goals and requirements. This chapter, along with Chapter 3, describe the safety-driven identification of a system control structure that is based on control and systems theory. There is nothing preventing the use of a similar approach for generating a control structure that satisfies other system properties, in addition to safety.

At this early stage, the hierarchy should be (a) sufficiently general, in order to avoid overly constraining potential design solutions and (b) account for the system-level

[3] "Route Planning" is intended to be a very high level, abstract term. This is the general term for the entities responsible for managing air traffic, making ultimate decisions on aircraft routes, identifying and solving conflicts, etc. Traditionally this has been the responsibility of ATC and associated decision support tools; in the future this responsibility may be delegated to operators, flight crews, and/or autonomous systems.

[4] "Piloting" is also intended to be a very high level, abstract term. This is the general term for the entities responsible for navigating and controlling aircraft. Traditionally this has been the responsibility of the flight crew, in conjunction with flight management systems. In the future, the 'pilot' could be any combination of flight crew and airborne automation, or even ground-based crews or automation.

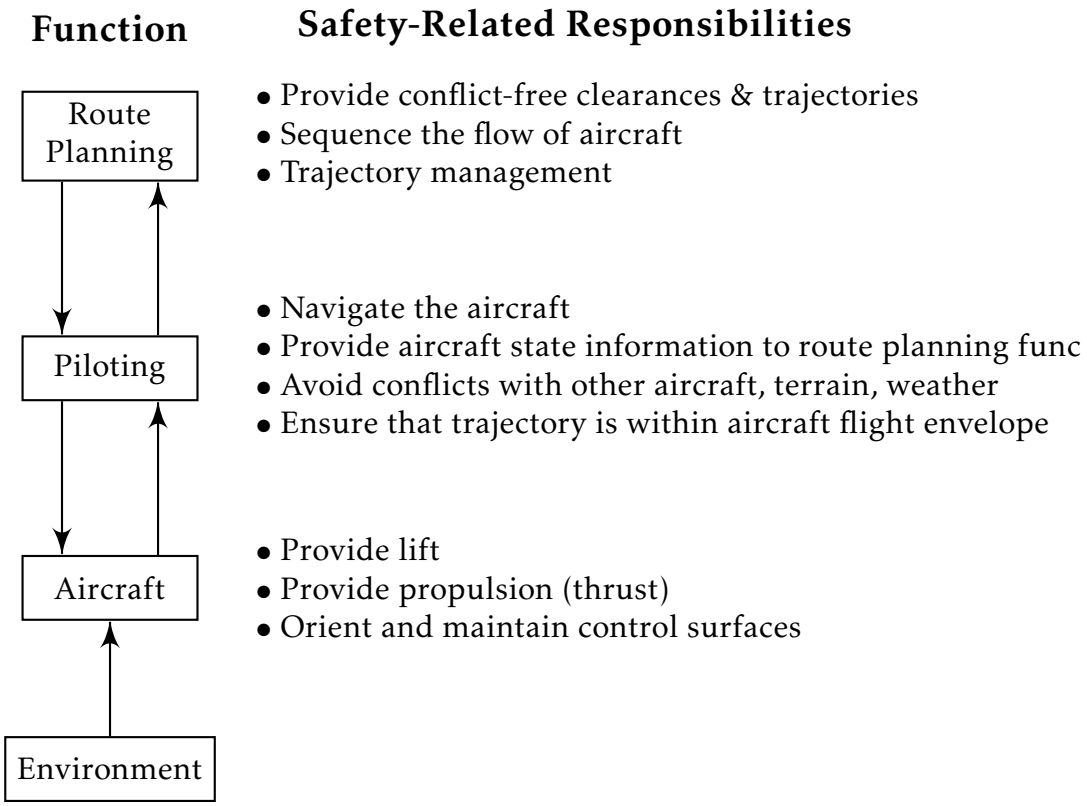


Figure 16: High Level Control Structure & Responsibilities

losses that stakeholders want to avoid, hazards associated with those losses, and constraints that will prevent those hazards. That is, the hierarchy should be relatively simple and concise, and yet these steps are important because they provide the framework and context for developing and analyzing the model. The following sections describe model generation (4.3) and analysis (4.4) for TBO Conformance Monitoring, one important aspect of the overall TBO concept.

4.3 Model Generation

Recall from Chapter 3, (see also Figure 17) the description and theory used to generate a model from textual or graphical descriptions of a concept. That theory is now applied to one chapter of the TBO ConOps, which describes the Conformance Monitoring function, while Appendix A contains further example analyses of TBO.

In the TBO ConOps [JPDO, 2011], there is a chapter dedicated to conformance monitoring, which is the degree to which an aircraft follows its agreed-upon trajectory. This example is intended to show how these control-theoretic concepts can be used to (1) query a certain aspect of a concept—it could be a sentence, paragraph, or figure—

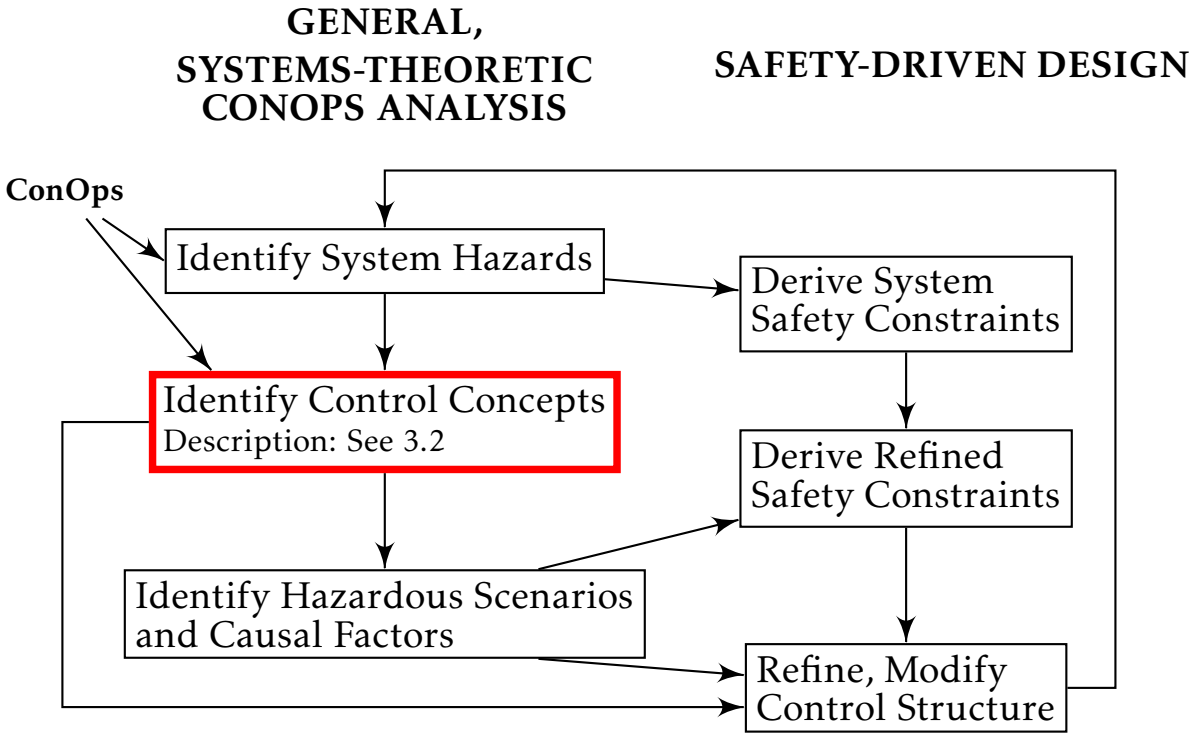


Figure 17: Methodology—Identifying Control Concepts

and then (2) to use the resulting information to build a control-theoretic model of the system. The following quote is one of the first sentences dedicated to conformance monitoring:

> $\mathcal{I}$–1 *TBO conformance is monitored both in the aircraft and on the ground against the agreed-upon [trajectory]. In the air, this monitoring (and alerting) includes lateral deviations...(actual lateral position compared to intended position), longitudinal based on flight progress in the FMS [aircraft software], vertical based on altimetry, and time from the FMS [aircraft software] or other "time to go" aids.* [JPDO, 2011]

To begin, the analyst must ask: What is the primary source, subject, or actor in the text, and in what way does this source relate to control theory? The quoted text describes *conformance*, or conformance monitoring.

Next, what is the source's role in control theory? Conformance monitoring acts as a *sensor*, and in this text there appear to be two versions of the sensor: one in the aircraft and another on the ground. Of the three generic roles that a sensor can take in the proposed framework, the conformance monitoring sensor provides two. See Table 10 on the following page for a summary of this brief example analysis, and note that a separate model will be developed for the ground conformance monitor shortly.

Table 10: Example Analysis of Text—TBO Conformance Monitoring

Subject	Conformance monitoring, Air automation
Role	Sensor
Behavior Type	Transmits binary or discretized state data to controller (i.e. measures behavior of process relative to thresholds; has algorithm built-in but no control authority)
	Synthesizes and integrates measurement data
Context	This is a decision support tool that contains algorithms to synthesize information and provide alerting based on some criteria.

With this high-level picture of conformance monitoring and its role in the context of control theory, the rest of the information from the quoted text can be used to fill out further details about the control model. From the text we can identify that conformance monitoring has directly to do with identifying an aircraft's position (latitude, longitude, altitude, at a given time). The text also suggests that this position will be compared with the aircraft's *intended* position. A sensor is used to inform a controller's process model. Therefore, it can be assumed that these position variables are related to a process model.

Next, the conformance monitor uses the FMS (aircraft software) and associated equipment to measure these process model variables. Finally, from the high level control structure and responsibilities, shown in Figure 16 on page 99, it can be inferred from the text that the airborne conformance monitor is related to the control function of piloting the aircraft.

Table 11 shows these results in tabular form, relating to each control loop element necessary for ensuring control and coordination. Notice that the table is incomplete. Although much of the missing information can be inferred, particularly by an expert on this system, the table cannot and should not be completed without further investigation into the rest of the ConOps or other NextGen documentation. Also, it is important to bear in mind that this incompleteness does not imply that the ConOps is "incomplete". Rather, this is simply the first step in the systematic, recursive process and completeness can only be considered when this process has been applied over the entire concept, or system boundary.

Table 11: Preliminary Control Model of Conformance Monitor Example

See Fig. 11		**Description**
1.	Controller	Piloting function

See Fig. 11		**Description**
2.	Actuator	
3.	Cntl'd Process	Aircraft
4.	Sensor	Altimeter, FMS, Aircraft conformance monitor
5.	Process Model	Intended latitude, longitude, altitude, time; Actual latitude, longitude, altitude, time
6.	Cntl Algorithm	
7.	Control Actions	
8.	Controller Status	
9.	Control Input	
10.	Controller Output	
11.	External Input	
12.	Alt Controller	
13.	Process Input	
14.	Proc Disturbance	
15.	Process Output	

Figure 18 (page 106 at the end of this sub-section) graphically depicts how the natural language text can be mapped onto a control loop. While a database format of Tables 10 and 11 assists in data storage, analysis, and retrieval, a graphical representation improves a user's ability to visualize the control-theoretic elements of the concept. Both formats—databases and graphical control loops—lend themselves to a formalism that allows for automatic checking of consistency and completeness.

Finally, quote $\mathcal{I}$–1 on page 100 also mentions conformance monitoring on the ground. Though its role is notionally the same, ground monitoring represents a different source of information. The model should reflect this difference. Because the rest of the quoted text describes conformance monitoring in the air, little can be inferred about the ground version of conformance monitoring and thus the analyst must look for additional detail elsewhere in the ConOps. The initial model for ground-based conformance monitoring has very little detail and is shown in Table 12.

Table 12: Initial Control Model of Ground Conformance Model

Subject	Conformance monitoring, Ground automation
Role	

Behavior Type	
Context	This is a decision support tool that contains algorithms to synthesize information and provide alerting based on some criteria.

⇓

1.	Controller	
2.	Actuator	
3.	Cntl'd Process	Airspace
4.	Sensor	
5.	Process Model	
6.	Cntl Algorithm	
7.	Control Actions	
8.	Controller Status	
9.	Control Input	
10.	Controller Output	
11.	External Input	
12.	Alt Controller	
13.	Process Input	
14.	Proc Disturbance	
15.	Process Output	

Consider the next sentence in the TBO ConOps:

$\mathcal{I}$–2 *Independent of the aircraft, the ANSP uses ADS-B position reporting for lateral and longitudinal progress, altitude reporting for vertical, and tools that measure the time progression for the flight track. Data link provides aircraft intent information. Combined, this position and timing information is then compared to a performance requirement for the airspace and the operation. ...precision needed...will vary based on the density of traffic and the nature of the operations.* [JPDO, 2011]

Prior to the above text, the analysis thus far only reveals that there is a ground-based conformance monitor, it performs the role of sensor, and the controlled process is the aircraft. The quote reveals several additional things about ground-based conformance monitoring: the behavior type, the type of sensing used by the ANSP, the pro-

cess model states to be used, as well as additional sources of information used by the ground (ANSP) entity. See Table 13 for a summary of this brief example analysis, and note that a separate model is developed for the ground conformance monitor.

Table 13: ANSP/Ground—TBO Conformance Monitoring

Subject	Ground automation
Role	Sensor
Behavior Type	Synthesizes and integrates measurement data
Context	This is a decision support tool that contains algorithms to synthesize information and provide alerting based on some criteria.

The information in Table 13 on page 104 is similar to that in Table 10. However, the information contained in this particular quoted text is much different than that in $\mathcal{I}$–1 on page 100, and the resultant control model is much different. Table 14 shows that the ground-based conformance monitor may be using a different set of tools to monitor conformance than airborne conformance monitoring, and the ground-based monitor also has a different process model and controlled process.

Figure 19 on page 107 represents the graphical form of the same model.

Table 14: Preliminary Control Model of Ground Conformance Monitor

See Fig. 11		**Description**
1.	Controller	Ground ≡ ANSP
2.	Actuator	
3.	Cntl'd Process	Airspace
4.	Sensor	ADS-B, altitude reporting, "tools", Data-link - trajectory intent information
5.	Process Model	All (all aircraft under ANSP jurisdiction) Intended lat, long, alt, time; All Actual lat, long, alt, time; traffic density; operation type; performance requirement
6.	Cntl Algorithm	
7.	Control Actions	
8.	Controller Status	
9.	Control Input	
10.	Controller Output	
11.	External Input	

See Fig. 11		**Description**
12.	Alt Controller	
13.	Process Input	
14.	Proc Disturbance	
15.	Process Output	

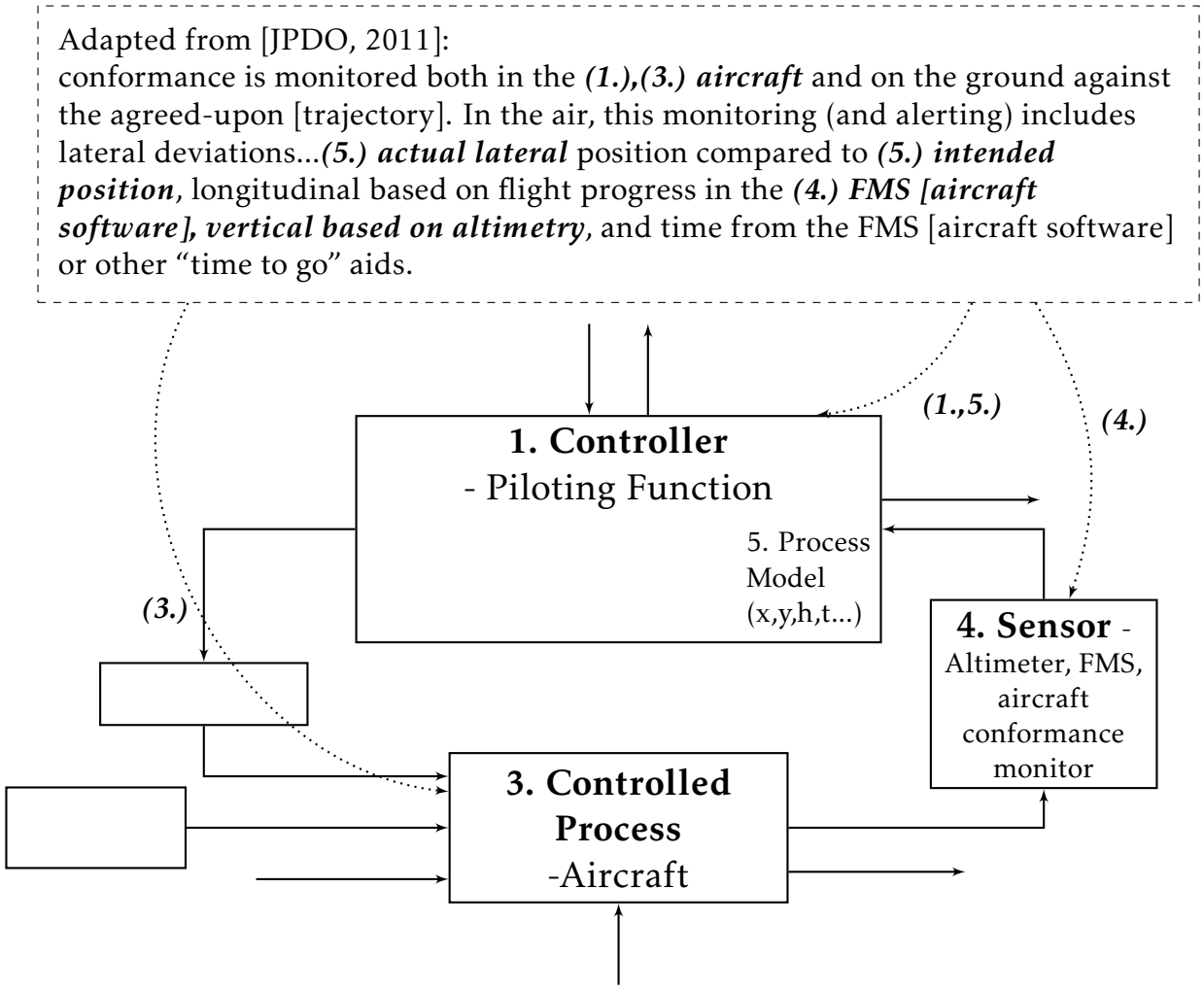


Figure 18: Graphical Control Model of Airborne Conformance Monitor

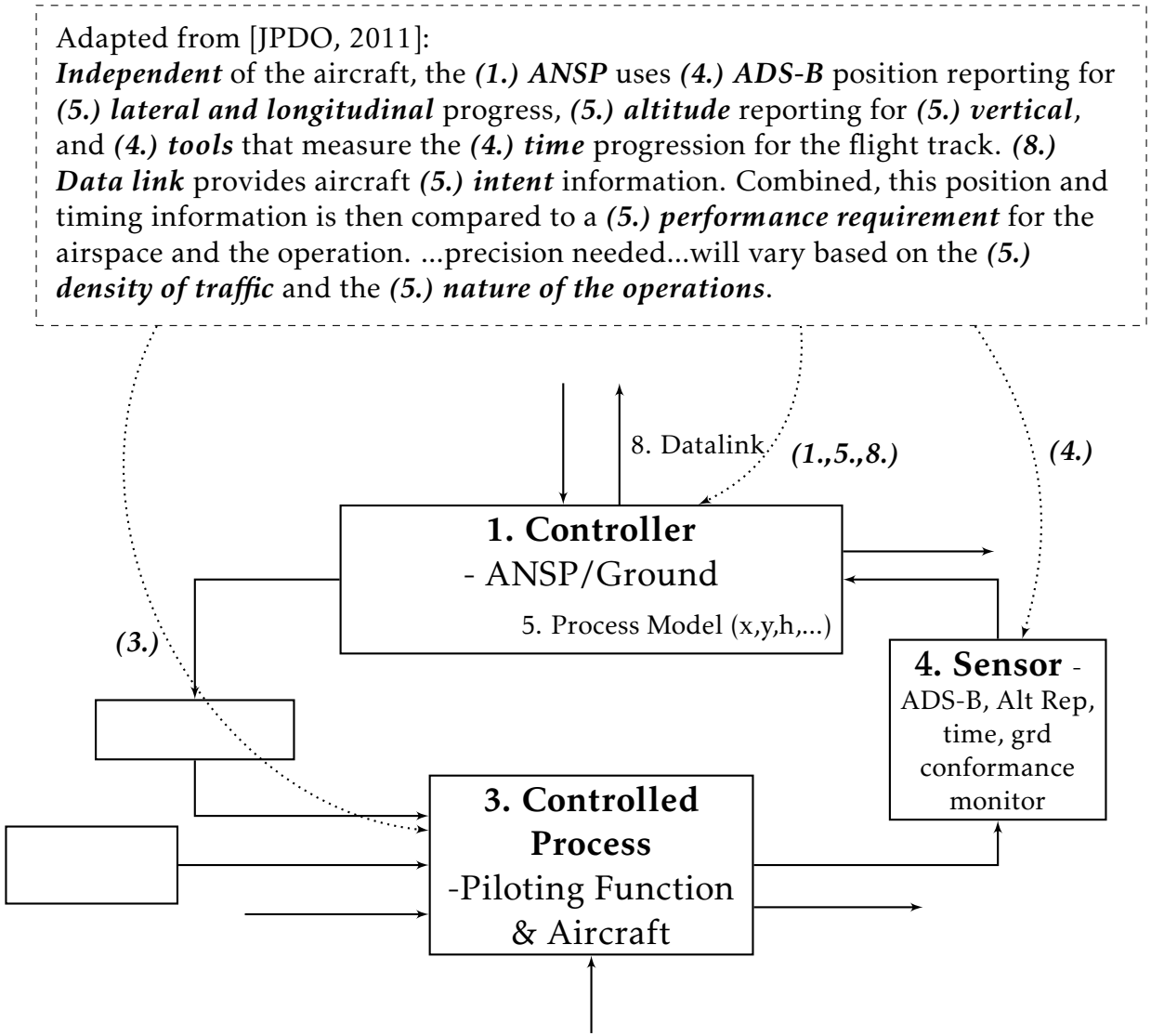


Figure 19: Graphical Control Model of Ground Conformance Monitor

Recursive Application of Method—Conformance Monitoring

The following presents a recursive application of the method over the remainder of the chapter in the TBO ConOps dedicated to conformance monitoring. After each iteration the control model is updated, and the updated aspects of the model are depicted as **bold and underlined**, while the parts of the model from previous iterations are depicted using standard font.

The following quote adds more detail about performance requirements and their relation to conformance monitoring. The updated control model is shown for the ground automation only in Table 15, although similar results hold for the airborne automation.

> $\mathcal{I}$–3 *In framing the required performance..., TBO recognizes that traffic density drives needed performance. There may be departures where a lateral precision of RNP 0.3 is required close in to the airport, and where time is measured in seconds. RTP is used as a tool to separate crossing traffic, and where vertical altitude restrictions are necessary. All of these factors must be considered in defining the parameters for conformance monitoring.* [JPDO, 2011]

Table 15: Updated Control Model for $\mathcal{I}$–3

See $\rightarrow \mathcal{I}$–3 of $\mathbb{C}_{\text{TBO}}$	
Subject	Conformance monitoring, Ground automation
Role	Sensor
Behavior Type	Synthesizes and integrates measurement data
Context	This is a decision support tool that contains algorithms to synthesize information and provide alerting based on some criteria.

⇓

1.	Controller	Ground ≡ ANSP
2.	Actuator	
3.	Cntl'd Process	Airspace
4.	Sensor	ADS-B, altitude reporting, "tools", Data-link

See → $\mathcal{I}$–3 of $\mathbb{C}_{\text{TBO}}$		
5.	Process Model	All (all aircraft under ANSP jurisdiction) Intended lat, long, alt, time; All Actual lat, long, alt, time; traffic density; operation type; performance requirement **RNP, RTP**
6.	Cntl Algorithm	**Crossing vs flow traffic**
7.	Control Actions	
8.	Controller Status	
9.	Control Input	
10.	Controller Output	
11.	External Input	
12.	Alt Controller	
13.	Process Input	
14.	Proc Disturbance	
15.	Process Output	

The following information ($\mathcal{I}$–4) from the TBO ConOps describes the predicted aircraft state information in slightly different terms than the previous descriptions. Therefore Table (16) includes the new text along with the text it replaced.

$\mathcal{I}$–4 *Conformance monitoring has an expected ground track, climb performance (based on known aircraft type, weight, and preferred profile), and time performance. In conformance monitoring, the aircraft is on a closed trajectory.* [JPDO, 2011]

Table 16: Updated Control Model for $\mathcal{I}$–4

See → $\mathcal{I}$–4 of $\mathbb{C}_{\text{TBO}}$	
Subject	Conformance monitoring, Ground automation
Role	Sensor
Behavior Type	Synthesizes and integrates measurement data
Context	This is a decision support tool that contains algorithms to synthesize information and provide alerting based on some criteria.

See $\rightarrow \mathcal{I}$–4 of $\mathbb{C}_{\text{TBO}}$		
1.	Controller	Ground ≡ ANSP
2.	Actuator	
3.	Cntl'd Process	Airspace
4.	Sensor	ADS-B, altitude reporting, "tools", Datalink
5.	Process Model	All (all aircraft under ANSP jurisdiction) Intended **ground track**, **climb performance**, **time performance** ~~lat~~, ~~long~~, ~~time~~; All Actual **ground track**, **climb performance**, **time performance** ~~lat~~, ~~long~~, ~~time~~; traffic density; operation type; performance requirement RNP, RTP; aircraft type, weight, performance **profile** ~~requirement~~
6.	Cntl Algorithm	Crossing vs flow traffic
7.	Control Actions	
8.	Controller Status	
9.	Control Input	
10.	Controller Output	
11.	External Input	
12.	Alt Controller	
13.	Process Input	
14.	Proc Disturbance	
15.	Process Output	

The next quote from the TBO ConOps ($\mathcal{I}$–5) describes separation in terms of the overlap between the protected airspace of two or more aircraft. The protected airspace is described as an elliptical shape, and the size of these ellipses is dependent on the certainty of surveillance and navigation information. Importantly for TBO, this characterization of protected airspace is much different than the current paradigm of protected airspace, where every aircraft is laterally separated by a fixed distance or vertical separation, depending on the operation and type of surveillance [Ray, 2014].

In the traditional approach, protected airspace is a disc with a radius equal to the lateral separation and height equal to the vertical separation. Table 17 includes updated information about potentially unsafe actions for both the general functions in air traffic management as well as updated information for the process model and con-

trol algorithm. Table 17 describes only the ground automation portion of the system, but similar updates are included for airborne automation.

$\mathcal{I}$–5 *As the aircraft approaches level-off and cruise, the shape of the protected airspace morphs into more of an elliptical 3-D shape, where the aircraft is positioned in the narrow end of the elliptical shape, with the wake vortex "tail" as its aft bound and vertical, lateral, and longitudinal uncertainty defining the flexible airspace. No two elliptical shapes can overlap if separation is to be assured. In this case, Aircraft A and Aircraft B have crossing trajectories. Aircraft A's protected space is smaller because it has less uncertainty than Aircraft B. The trailing area of protection may reflect wake turbulence requirements. The lateral protection is the uncertainty in navigation performance, while the leading distance along the flight path represents the time uncertainty. In level flight, the vertical altitude dimension is quite small.* [JPDO, 2011]

Table 17: Updated Control Model for $\mathcal{I}$–5

Subject	Conformance monitoring, Ground automation
Role	Sensor
Behavior Type	Synthesizes and integrates measurement data
Context	This is a decision support tool that contains algorithms to synthesize information and provide alerting based on some criteria. **Potential unsafe control action for trajectory generation function: Approving a 4DT that will lead to LOS or not modifying a 4DT that will overlap** **Potential unsafe control action for piloting function: Aircraft is following a 4DT that will lead to LOS**

1.	Controller	
2.	Actuator	
3.	Cntl'd Process	Piloting function and aircraft, Airspace
4.	Sensor	ADS-B, altitude reporting, "tools", Datalink

See → $\mathcal{I}$–5 of $\mathbb{C}_{\text{TBO}}$		
5.	Process Model	All (all aircraft under ANSP jurisdiction) Intended ground track, climb performance, time performance; All Actual ground track, climb performance, time performance; traffic density; operation type; performance requirement RNP, RTP; aircraft type, weight, performance profile; **Wake turbulence**; **Ellipse, uncertainty** (shape of conformance tolerance)
6.	Cntl Algorithm	Crossing vs flow traffic; **Overlap of 2 or more 4DTs (Ellipses)**
7.	Control Actions	
8.	Controller Status	
9.	Control Input	
10.	Controller Output	
11.	External Input	
12.	Alt Controller	
13.	Process Input	
14.	Proc Disturbance	
15.	Process Output	

The TBO ConOps also describes an alerting function for the conformance monitoring automation for both the ground controller ($\mathcal{I}$–6) and flight crew ($\mathcal{I}$–7). The basic idea of the alerting function is to allow the user—either ATC or flight crews, independently—to set alerts for measuring an aircraft's progress against its assigned 4DT. below). Table 18 contains the updated model information for the ground controller, developed from quote $\mathcal{I}$–6.

> $\mathcal{I}$–6 *Alerting is triggered by automation and alerts the controller to transgression from the conformance airspace, and may be set as alerts for measuring progress. By setting progress alerts, the controller has an aid to measure progress in meeting the 4DT.* [JPDO, 2011]

Table 18: Updated Control Model for $\mathcal{I}$–6

Subject	ATC

See $\rightarrow \mathcal{I}$–6 of $\mathbb{C}_{\text{TBO}}$	
Role	Controller
Behavior Type	Processes inputs from sensors to form and update process model
Context	ATC measures progress via conformance monitoring automation Potential unsafe control action for trajectory generation function: Approving a 4DT that will lead to LOS or not modifying a 4DT that will overlap

1.	Controller	**ATC (controller)**
2.	Actuator	
3.	Cntl'd Process	Piloting function and aircraft, Airspace
4.	Sensor	ADS-B, altitude reporting, "tools", Datalink, **alerting automation**
5.	Process Model	All (all aircraft under ANSP jurisdiction) Intended ground track, climb performance, time performance; All Actual ground track, climb performance, time performance; traffic density; operation type; performance requirement RNP, RTP; aircraft type, weight, performance profile; Wake turbulence; Ellipse, uncertainty
6.	Cntl Algorithm	**Generate or compute conformance monitoring airspace volume [automation]; compare actual position with monitoring volume [automation]**
7.	Control Actions	
8.	Controller Status	
9.	Control Input	
10.	Controller Output	
11.	External Input	
12.	Alt Controller	

See → $\mathcal{I}$–6 of $\mathbb{C}_{\text{TBO}}$		
13.	Process Input	
14.	Proc Disturbance	
15.	Process Output	

Table 19 contains the updated model information for the ground controller, re-framing quote $\mathcal{I}$–7 in control-theoretic terms.

$\mathcal{I}$–7 *From the cockpit, the pilot can monitor performance, as well. Most of the tools are already used. Altitude alerts exist. RNP can be monitored, and the progress can be provided by the FMS. What is needed is the cockpit display of traffic information (CDTI) with tools for merging, spacing, and separation. These tools will help the pilot monitor other traffic as well as progress in meeting the 4DT. The pilot sets the alerting parameters in the respective automation.* [JPDO, 2011]

Table 19: Updated Control Model for $\mathcal{I}$–7

Subject	Flight crew
Role	Controller
Behavior Type	Creates, generates, or modifies control actions based on algorithm or procedure and perceived model of system Processes inputs from sensors to form and update process model Processes inputs from external sources to form and update process model
Context	Crew measures progress via conformance monitoring automation crew makes decisions for merging, spacing, separation Potential unsafe control action for piloting function: Aircraft is following a 4DT that will lead to LOS

1.	Controller	**Flight crew**
2.	Actuator	Not specified; assume FMS and manual control
3.	Cntl'd Process	Implied; Aircraft control surfaces, thrust

See $\rightarrow \mathcal{I}$–7 of $\mathbb{C}_{\text{TBO}}$		
4.	Sensor	**CDTI, FMS, Altitude alerts**, conformance monitoring automation
5.	Process Model	Ownship Intended ground track, climb performance, time performance; Ownship Actual ground track, climb performance, time performance; traffic density; operation type {climb,cruise,arrival}; performance requirement RNP, RTP; aircraft type, weight, performance profile; flight object
6.	Cntl Algorithm	**Implied: Other aircraft (relevant, all?)** Intended ground track, climb performance, time performance; Ownship Actual ground track, climb performance, time performance; traffic density; operation type {climb,cruise,arrival}; performance requirement RNP, RTP; aircraft type, weight, performance profile; flight object
7.	Control Actions	
8.	Controller Status	
9.	Control Input	**Not specified; related to merging, spacing, sequencing, and assuring conformance**
10.	Controller Output	
11.	External Input	
12.	Alt Controller	**ATC (ANSP)**
13.	Process Input	
14.	Proc Disturbance	
15.	Process Output	

The final text ($\mathcal{I}$–8) regarding conformance monitoring in the TBO ConOps is concerned with open versus closed trajectories. This is the essence of what conformance monitoring is intended to achieve: the determination of whether or not an aircraft is following its assigned 4DT. That is, conformance monitoring determines whether an aircraft is on an *open* or *closed* trajectory.

The language used in $\mathcal{I}$–8 is slightly different than the previous conformance-

related text ($\mathcal{I}$–1 through $\mathcal{I}$–7). First, the text simply says "Automation", and it is referring to the more general TBO-related automation beyond the conformance-related automation. Next, the text mentions downstream conflicts and not just immediate overlaps of different trajectories; earlier quotes from the text mention only the overlap of elliptical airspace ($\mathcal{I}$–5 on page 111). Finally, the text states that open trajectories can "even lead to a conflict requiring intervention".

Though the model developed in Table 20 does not include such explicit terms, the emphasis on "intervention" for only open trajectories implies that (1) TBO automation will generate conflict-free, closed trajectories; (2) the automation cannot handle open trajectories (stated in weaker terms in $\mathcal{I}$–8); and (3) that intervention should be rare. Furthermore, from elsewhere in the text it is inferred that this intervention will come from ground controllers, but the possibility exists that on-board collision avoidance systems will also provide this intervention function.

$\mathcal{I}$–8 *It is difficult for automation to deal with open trajectories. The uncertainties that open trajectories introduce affect more than just the aircraft in question and may impact downstream flows, and even lead to a conflict requiring intervention to assure safety.* [JPDO, 2011]

Table 20: Updated Control Model for $\mathcal{I}$–8

Subject	Automation
Role	Controller
Behavior Type	Creates, generates, or modifies control actions based on algorithm or procedure and perceived model of system Processes inputs from external sources to form and update process model
Context	$\mathcal{I}$–8 does not specify whether there is a human user of this automation, or the user and location (e.g. ground or flight deck). It does imply, however, that it is the general "Trajectory Generation Function" that is in control It is assumed that controller / ANSP "intervention" is intended to be rare

1.	Controller	**ANSP or Flight Crew**

See $\rightarrow \mathcal{I}$–8 of $\mathbb{C}_{\text{TBO}}$		
2.	Actuator	
3.	Cntl'd Process	Piloting function and aircraft, Airspace
4.	Sensor	ADS-B, altitude reporting, "tools", Datalink, alerting automation, **TBO Trajectory automation**
5.	Process Model	All (all aircraft under ANSP jurisdiction) Intended ground track, climb performance, time performance; All Actual ground track, climb performance, time performance; traffic density; operation type; performance requirement RNP, RTP; aircraft type, weight, performance profile; Wake turbulence; Ellipse, uncertainty; **Downstream flows (flow model, in addition to "All Intended trajectory" model)**
6.	Cntl Algorithm	Generate or compute conformance monitoring airspace volume [automation]; compare actual position with monitoring volume [automation]; **Decision about aircraft conformance vs non-conformance [automation]**
7.	Control Actions	
8.	Controller Status	
9.	Control Input	
10.	Controller Output	
11.	External Input	
12.	Alt Controller	
13.	Process Input	
14.	Proc Disturbance	
15.	Process Output	

It is important to again emphasize that the model development process should be limited, to the extent possible, to the individual information element that is currently being analyzed. Because the proposed method is easily repeatable and should be applied to the entire set of information elements, $\mathcal{I} \in \mathbb{C}$, the control model represented

by Table 20 contains only the assumptions that can be gleaned from $\mathcal{I}$–8, while also documenting additional assumptions or inferences in the "Context" row.

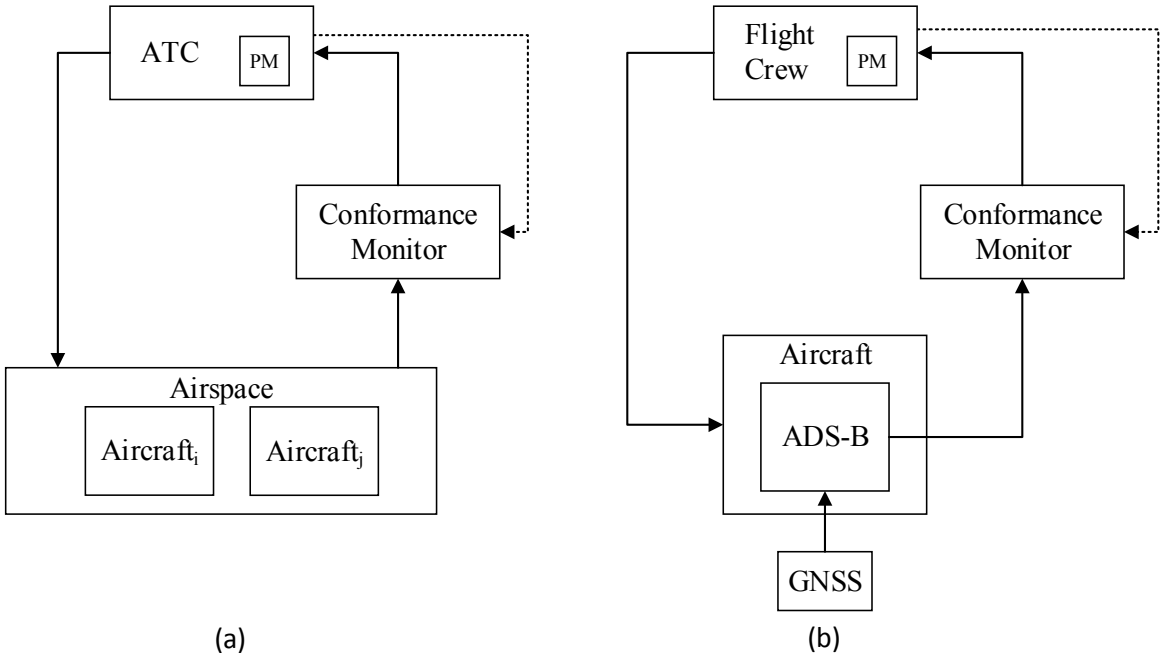


Figure 20: Individual Control Loops derived via Analysis

At this point there are two separate control models under development—one of the aircraft as a control entity and another of the ground as a control entity. Again, any distinction among pilots or aircraft avionics (e.g. FMS) as having control of the "airborne" function is unnecessary at this point in the analysis. However, the TBO ConOps development of conformance monitoring implies that pilots and air traffic controllers will still have control authority in mid- and long-term TBO systems. Figure 20 provides a high-level depiction of these separate feedback loops.

Because this research is concerned with identifying behaviors that emerge due to the interactions between and among various components, it is important to determine if and how these individual control loops relate to each other. Some of this work has already been done, as Section 4.2 on page 95 laid the groundwork via a general, hierarchical decomposition of the functions required to maintain safe airspace.

These distinct control models, then, can be synthesized via a mapping to the functional control structure and hierarchical safety responsibilities derived earlier. In particular, the TBO ConOps states elsewhere that the "ANSP's authority over the airspace and the flight crew's authority over the aircraft's trajectory do not change" [JPDO, 2011]. It is therefore appropriate and intuitive to map the ground-based conformance

monitoring loop to the 'Trajectory Management Function' and the airborne loop to the 'Piloting Function' of Figure 16 on page 99.

Figure 21 depicts the control structure that results from analyzing only the chapter dedicated to conformance monitoring in the TBO ConOps, with the rest of details identified in Tables 11–20. Appendix A contains the rest of the model generation results for other aspects of the TBO Concept of Operations.

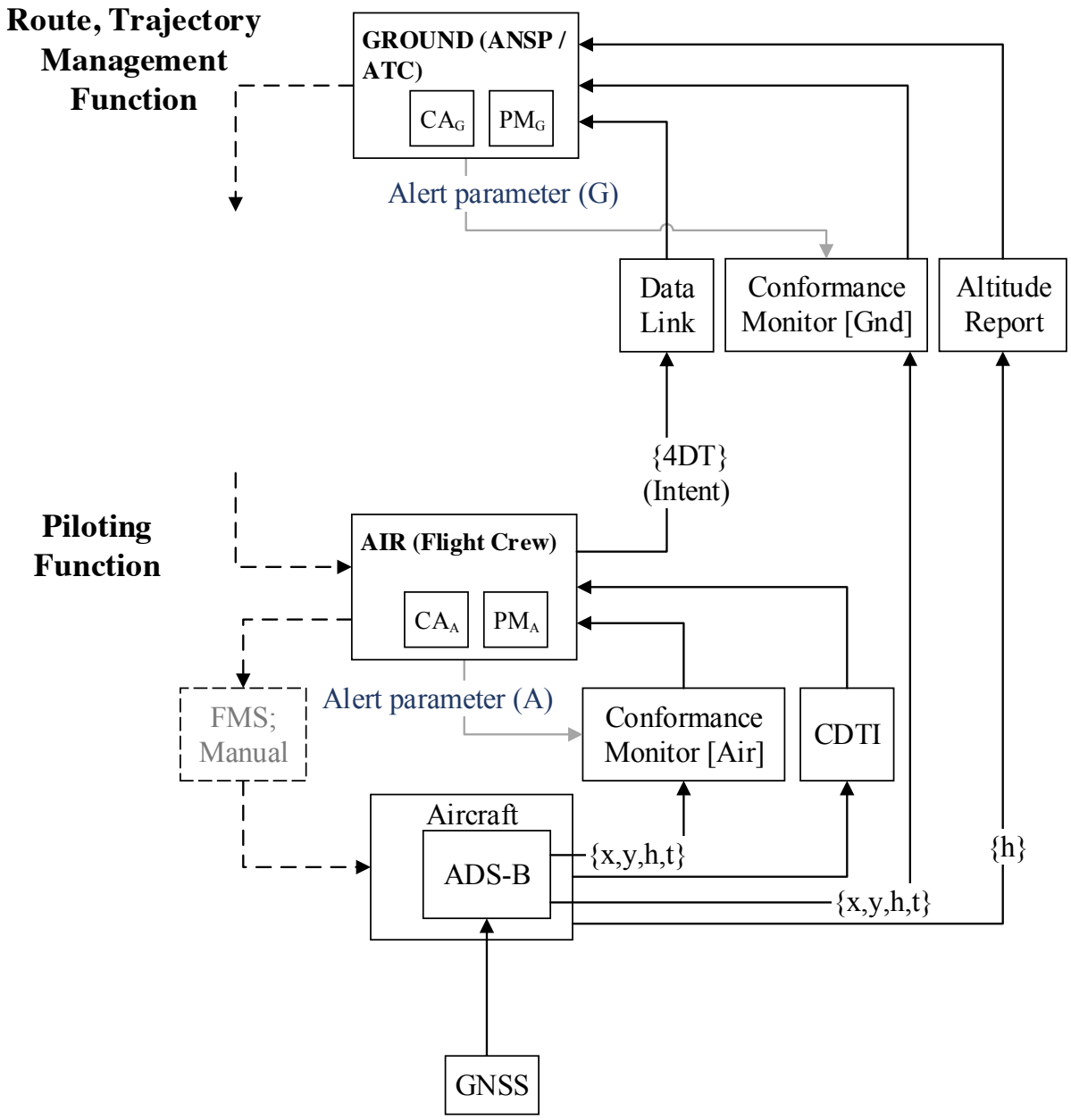


Figure 21: TBO Conformance Monitoring Control Structure

Following is a description of each element in the synthesized model (Figure 21). GNSS, ADS-B, CDTI, and FMS are not included in the development here because they

have already been developed thoroughly elsewhere.

Though conformance monitoring functions clearly have a model of the process and an algorithm for determining if an aircraft conforms to its prescribed trajectory, conformance monitoring is *not* a controller. Rather, conformance monitoring is a lower level function that maps refined state information into more abstract state information. That is, conformance monitoring takes detailed surveillance information (among other things), and in its simplest form outputs a binary datum to a higher level control function. The control functions, which could be other forms of automation or human operators, ultimately have to decide whether an aircraft conforms and then determine an action based on this understanding of aircraft conformance. It is for these reasons, as well as the preceding analyses (Tables 10–20), that the conformance monitors are depicted as sensors in Figure 21.

Because the conformance monitors are modeled as sensors, the control functions in Figure 21 may be modeled with relatively high level, abstract process models and algorithms.

Ground-based Conformance Monitoring Model

Qualitatively, the conformance monitoring consists of a comparison between the measured state of an aircraft in 4 dimensions and its intended state, also in 4 dimensions (three positions and time). If the aircraft deviates from its intended state by more than some tolerance, the conformance monitor will alert the ground controller. The ground-based monitor should have a model of every TBO-enabled aircraft within its jurisdiction.

The TBO ConOps recognizes that navigation performance will never be perfect, so there is a tolerance bound, within which the aircraft can still be considered to "conform". Therefore, in addition to 4D states, the ground-based conformance monitoring model is a function of traffic density, the operation type, and required navigation performance. The tolerance bounds will also be a function of a wake turbulence model, and predicted downstream traffic flows. Finally, the ground control agent may set an "alert parameter". Table 21 summarizes the content of the model.

Formal Conformance Monitoring Model

Following is a more rigorous, mathematical formulation that can be used later in the systems engineering process for more detailed analysis, to develop a formal specification, or to generate and integrate into MBSE tools.

Table 21: Conformance Monitoring Model Variables

Conf. Monitoring	Measured latitude, longitude, altitude for all aircraft
	Intended latitude, longitude, lltitude for all aircraft
	Measured Traffic Density
	Type of Operation (e.g. cruise, descent)
	Navigation Performance (RNP, RTP)
	Wake Turbulence Prediction, Measurement
	Elliptical Conformance Volume
	Downstream, Predicted Traffic Flow
	Alert Parameter

The process model for the ground-based control agent (i.e. the ANSP) is simply,

$$\mathrm{PM_G} := \{\mathrm{Conf}_i, \dot{x}_i\} \tag{46}$$

where Conf_i represents whether each aircraft, i, in the ANSP jurisdiction is conforming with its prescribed trajectory, and $\dot{x}_i$ represents some prediction of the future states of each aircraft. The ground-based control actions are unspecified in the TBO ConOps' chapter regarding conformance monitoring, but it can still be inferred that the ANSP control algorithm is a function of conformance and that the ANSP will generate an action that enforces conformance. This analysis is for demonstration purposes, and additional analysis of the TBO ConOps reveals other details about ground-based control algorithms, available actions, and models of the airspace. Appendix A contains further analysis regarding ground-based control agents.

Similar to a control agent, conformance monitoring develops a model of the system and contains an algorithm to determine whether certain criteria are met. However, conformance monitoring functions as a sensor and does not have control authority based on what is proposed in the TBO ConOps. The sensor behavior is a mapping from measured variables to controller inputs (see equation. 17). For the ground-based conformance monitor, this mapping is

$$\mathcal{B}_{\mathcal{L}G} := \mathcal{V}_{mG} \rightarrow \mathcal{I}_{Gc}. \tag{47}$$

$\mathcal{I}_{Gc}$, is the signal going to the ground control agent regarding conformance, and the

functional behavior of the sensor is

$$\mathcal{V}_{mG} = \mathcal{L}_G \times D_{c,i} \tag{48}$$

where $\mathcal{L}_G$ is a model of the airspace state and $D_{c,i}$ is the decision criteria regarding conformance of aircraft i. The "ground", (or ANSP) conformance model is defined as the set of dynamic variables,

$$\mathcal{L}_{\mathrm{G}} := \{z_{\mathrm{int},i}, z_{\mathrm{act},i}, \rho, \tau, P_r, W, E_{cm}, F_D; i \in \mathbb{G}\} \tag{49}$$

where

$$\begin{aligned}
z_{\mathrm{int},i} &:= \{G, C, t\}_{\mathrm{int},i} \\
z_{\mathrm{act},i} &:= \{G, C, t\}_{\mathrm{act},i} \\
\rho &:= \text{Traffic density} \\
\tau &:= \text{Operation type} \\
P_r &:= \{\mathrm{RNP}, \mathrm{RTP}\} \\
W &:= \text{Wake turbulence model} \\
E_{cm} &:= \text{Elliptical conformance model} \\
F_D &:= \{F, z_{\mathrm{int},i}\}
\end{aligned}$$

and G is the ground track, C is the climb performance, and t is time of arrival, constituting the aircraft state, z. The $_{\mathrm{int}}$ and $_{\mathrm{act}}$ subscripts represent intended and actual performance, respectively, for the i^{th} aircraft within ANSP jurisdiction, which is the set $\mathbb{G}$. RNP and RTP represent standard definitions of navigation and time performance, and F_D is a downstream flow model consisting of a general flow model (F) of a particular airspace and a prediction of aircraft arrivals into that space.

Criteria for determining whether an aircraft conforms with its assigned trajectory are,

$$D_{c,i} = \left\{ z_{\mathrm{act},i} \middle| z_{\mathrm{act},i} \notin \bar{z}_i(z_{\mathrm{int},i}, E_{cm}, a_{\mathrm{G}}), \quad \forall i \in \mathbb{G} \right\} \tag{50}$$

where $\bar{z}_i$ is an allowed volume for aircraft i, as a function of the intended aircraft state in time, the elliptical conformance model at a given time (E_{cm}), an alert parameter set by the operator (a_{G}), and $\mathbb{G}$ is the set of aircraft under ANSP jurisdiction. Evaluation of Equation (50) to True indicates that aircraft i does not conform to its assigned

trajectory. This is a relatively primitive form of conformance monitoring that simply alerts the user—either a ground controller, flight crew, and/or other software functions that require information about conformance—whether the aircraft is following the assigned trajectory. More advanced forms of conformance monitoring could be developed, but the definition in equation (50) is consistent with the TBO ConOps.

Returning to the ground-based control agent (ANSP) in Figure 21, its control algorithm consists of—at a minimum—use of *some* decision on conformance alerting. That is,

$$\mathrm{CA_G} \supseteq \mathrm{Conf}_i \tag{51}$$

where Conf_i is obtained, according to equation 9 on page 65, via signal processing of the input signal $\mathcal{I}_{Gc}$.

Airborne Conformance Monitoring Model

The qualitative description of the airborne model is similar to the ground-based model. The significant differences are that the airborne monitor is responsible only for monitoring its own conformance, as opposed to the entire airspace. Conformance alerting is a function of flight crew input, rather than air traffic controller input. See Appendix A.1 for a similar, formal model of the airborne component.

Summary of Conformance Monitoring Models

In addition to the differences between ground and airborne models, conformance monitoring also assumes that other sources of information such as traffic density, operation type, performance requirements, and other factors are derived from the *same sources* as the ground-based conformance monitor. Further refinement of the process model variables reveals other common sources of information. For example, the TBO ConOps and NextGen in general prescribe ADS-B (Automatic Dependent Surveillance—Broadcast) and GNSS (Global Navigation Satellite System) as the primary sources of aircraft state and intent information for both ground and airborne avionics.

Before performing an analysis of the above model, a few observations can be made about the conformance monitor, its models, and its algorithms. With respect to the conformance model, the TBO ConOps provides a relatively detailed accounting of some of the variables, but little about others. For example, it is well documented that both actual and intended aircraft states will be derived primarily from ADS-B

surveillance data. Actual position is measured via satellite surveillance (GNSS), but the intended trajectory is ill-defined in the referenced chapter of the ConOps[5].

Traffic density is an important factor in determining whether to alert the relevant agents about non-conformance. The conformance monitoring concepts lacks a description of how traffic density is calculated, updated, and which agent(s) has responsibility to do so. It is beyond the scope of this research to determine this parameter, but the point here is that this method of modeling provides a rigorous way to catalog all the components, identify the relationships between components, and compare the relative fidelity of definition of these components and relationships.

The preceding model development made as few assumptions as possible regarding the allocation of control tasks, mechanisms for information exchange, authority, and other factors. The model development method has used only the information documented explicitly in the TBO ConOps section on conformance monitoring, and this information has been parsed in order to refine the functional control hierarchy presented at the beginning of this chapter (Figure 16 on page 99).

The following section describes the analytical process, but that analysis requires a more complete model in order to represent a valid demonstration. Refer to Appendix A, which applies this control-theoretic modeling approach to the larger TBO concept. The recursive application of this approach results in the allocation of roles and hierarchy among the various control agents, including further definition of the roles of human operators and automation. The modeling effort in Appendix A also captures additional information exchanges, available control actions, and data sources beyond conformance monitoring.

Upon completion of a hierarchical control model, the approach proceeds with a systems- and control-theoretic analytical process.

4.4 Analysis of the Model

This section applies the principles and theory developed in Section 3.3 to the concept of conformance monitoring in TBO. Hazardous scenarios and causal factors are identified according to three general categories—completeness of the control loops, analysis of safety-related responsibilities, and coordination and consistency. Figure 22 depicts this part of the effort, relative to the overall process.

[5] Other parts of the ConOps describe how the trajectory—and thus the intended aircraft states—will be negotiated and agreed-to, although the actual definition of a 4DT is left relatively vague. Other references define a 4DT with more detail, e.g. [Ballin et al., 2008; Jackson, 2010].

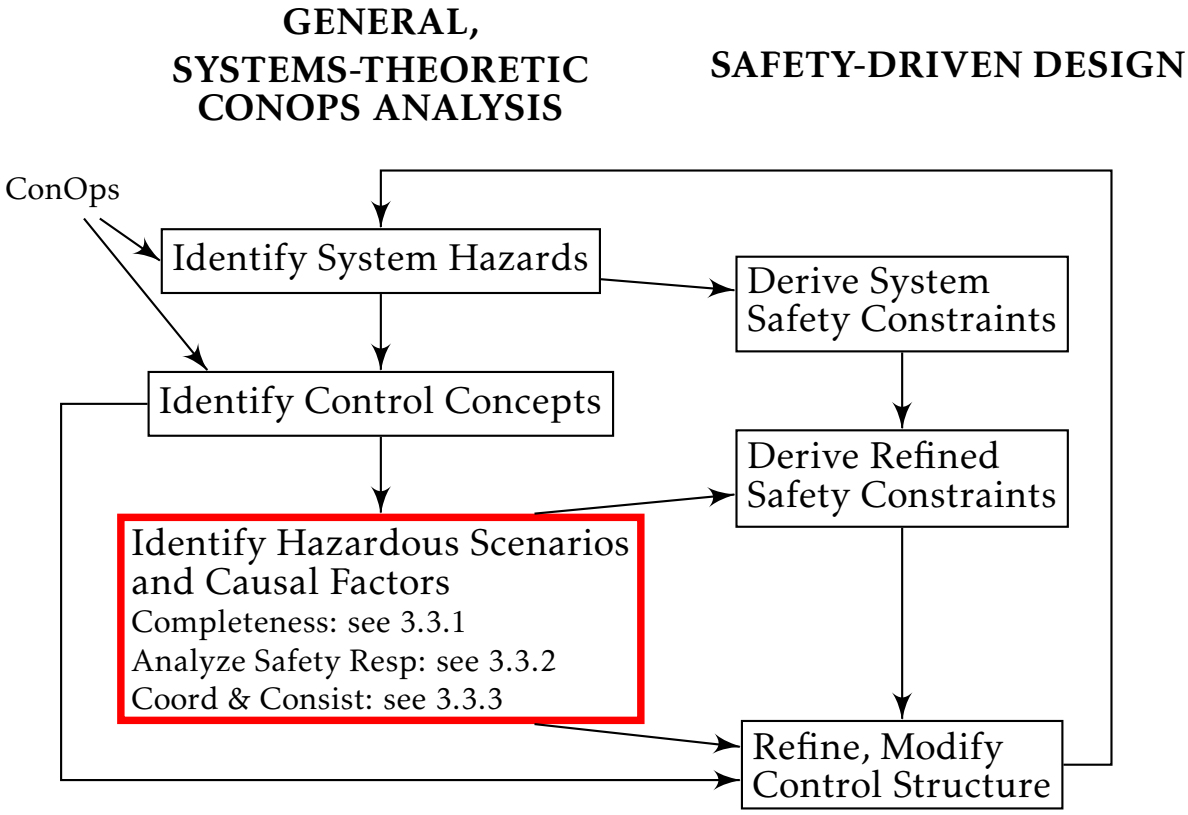


Figure 22: Methodology—Identifying Hazardous Scenarios

Simply applying these concepts to the model developed above (see Figure 21 on page 119) would certainly result in a set of potential causal factors. However, the model in Figure 21 is intentionally incomplete and represents only a subset of elements described by the entire TBO ConOps, and an analysis of such a model could result in the identification of potentially invalid or misleading causal factors. Appendix A documents the more general TBO model, which fills in some of the obvious gaps in the Conformance Monitoring model in the previous section.

The following analysis uses the more general model from Appendix B (see Figure 35 on page 191). For the purposes of demonstration, the analysis continues to focus on conformance monitoring but uses the more complete model in order to avoid identifying causal factors that arise simply because of an unfinished modeling effort.

4.4.1 Completeness Criteria for Individual Control Loops

Ignoring the flight operations centers, the model consists of two general control loops. The first model involves ANSP control of the airspace (Figure 23), and the second involves flight deck control of aircraft state (Figure 36). Each loop is analyzed for completeness individually, with the ANSP analysis included below and the flight

deck analysis in Appendix A.

ANSP (Ground) Control Loop

The model represented in Figure 23 contains more refined control structure information than the relatively abstract, functional model developed in the previous section. Elsewhere in the ConOps, the allocation of control tasks are assigned to various entities—for example trial planning automation, strategic TBO evaluation automation, tactical controller (i.e. human air traffic controller). Humans will be given choices to consider, negotiations to accomplish, agreements to be reached, and thus ATC is the primary control agent, while TBO automation is an input and feedback to the human [e.g. JPDO, 2011, p.41].

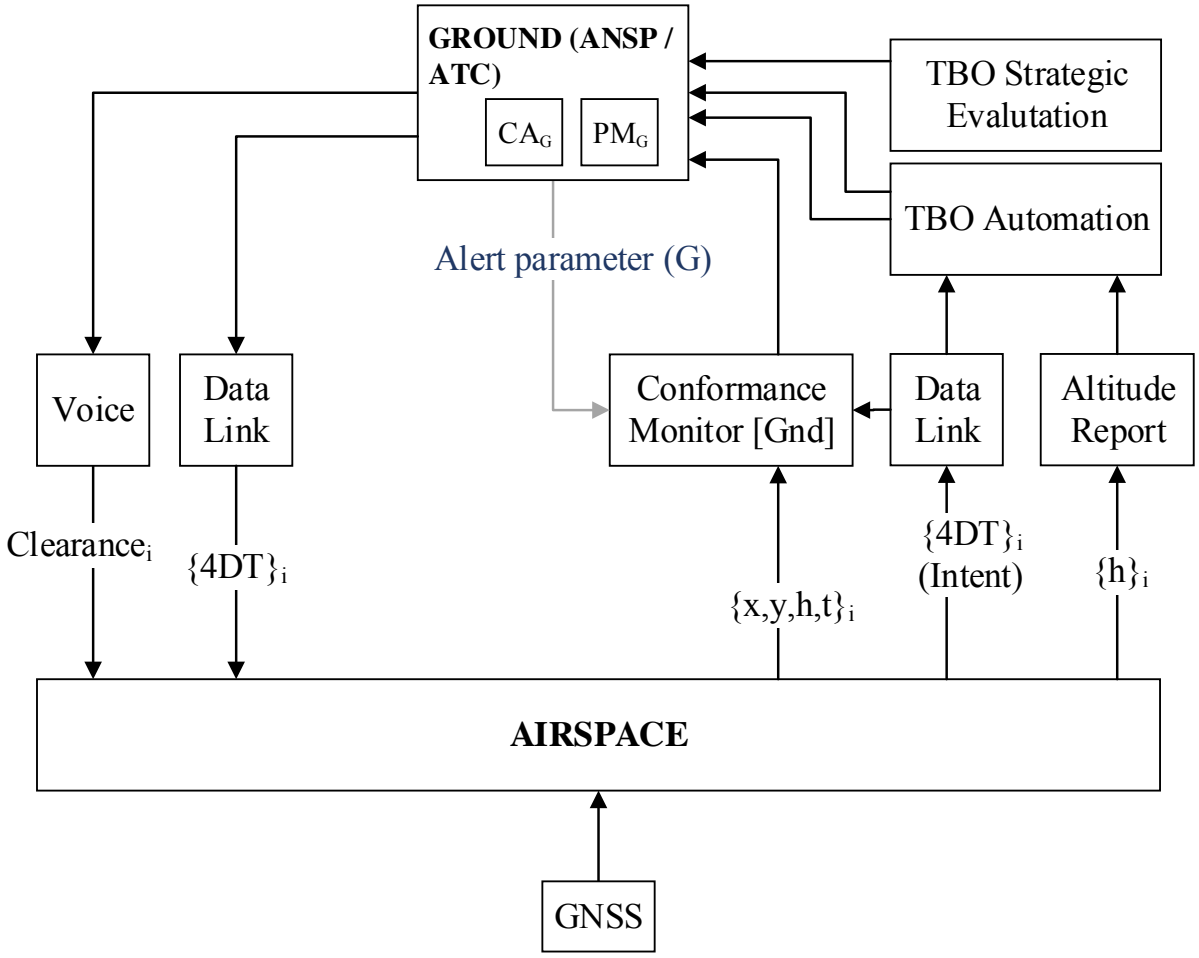
Because the ANSP is responsible for all the aircraft in its jurisdiction, individual aircraft are abstracted into a more general "Airspace" component. Upward information arrows in Figure 23 denote this using the $_i$ subscript.

- *Goal Condition* — what are the goal conditions, and how can they violate safety constraints and safety responsibilities?

  The conformance monitoring information in the TBO ConOps alludes to goals of merging, sequencing, and spacing, as well as assuring conformance. Conformance monitoring alone does not describe or specify how these goal conditions are generated or identified; the referenced chapter does, however, specify the information needed in the algorithm in order to support merging, spacing, and sequencing. See the description of Model Condition and Observability condition below.

  There are discussions and descriptions of merging, sequencing, and spacing elsewhere in the TBO ConOps, but it is left relatively vague in terms of algorithms or procedures[6]. There is nothing inherently negative about this lack of detail with respect to algorithms and procedures; however, the TBO concept emphasizes the goal of TBO being an improvement over current technologies and procedures for merging, sequencing, and spacing. Thus, the policy for how to achieve this goal must be made explicit. More important for the purposes of this demonstration, however, is the relationship between the goal of merging, sequencing, and spacing and the concept conformance monitoring.

[6] Other NextGen improvements that support TBO describe algorithms for merging and spacing, such as interval management and time-based metering.



NOTE: *i* subscripts: X_i represents the control and communication to the i^{th} aircraft

Figure 23: ANSP (Ground) Control Loops

The TBO ConOps *implies* a policy (and thus a goal) of assuring aircraft conformance in order to meet merging, sequencing, and spacing requirements. Therefore, it can be inferred that the Goal Condition related to conformance monitoring is to ensure conformance. However, the TBO ConOps provides relatively little information about the necessary policies (or goals, or algorithms) for scenarios where aircraft do not conform, to what extent they do not conform, and how non-conformance affects spacing.

In safety-driven design, it is not only important to identify the goal conditions for the various control agents but also to identify how goals can conflict with (or improve) safety. TBO emphasizes conformance and so-called "closed" trajectories. "A closed trajectory is one where the pilot, aircraft automation, the

controller, and ground automation all have the same view of what the aircraft is doing, from start of taxi through termination of flight operations" [JPDO, 2011, p.5]. This goal is potentially hazardous if adherence to the trajectory will lead to a hazard. Thinking in terms hazards and safety-driven design reveals the underlying assumption that the TBO system[7] will only generate trajectories without conflicts and can identify conflicts when they do occur.

However, an aircraft may have a good reason for not conforming, for example if the aircraft senses loss of separation or that conformance will exceed its capability and ground automation does not have access to this information. The goal of assuring conformance should therefore have the caveat that conformance is only desirable if the 4DT itself does not present a hazard to the aircraft or airspace. This caveat is intuitive and perhaps obvious, but it is either missing from, or obscured in, the TBO ConOps.

- *Action Condition* — how does the controller affect the state of the system? Are the available actions and actuators adequate or appropriate given the process dynamics?

  The ANSP controls the airspace using negotiated 4-dimensional trajectories as well as traditional clearances, which are mentioned elsewhere in the TBO ConOps. The "action" is a 4DT delivered via data link or a clearance delivered via voice communication. Are the actuators appropriate for the controlled process?

  Another important factor to consider is that 4DT actually implies 6-dimensional action. Aircraft are inertial systems with three dimensions of motion (when airborne), so their state space representations have six degrees of freedom. In traditional air traffic management, the six aircraft states are modified using speed controls, altitude modifications, and heading modification (i.e. vectors). Traditional clearances represent a relatively direct control input to the six aircraft degrees of freedom, while a 4DT represents a mapping to that six-DOF space. That is, 4-dimensional trajectories (may) require an extra step of mathematical or cognitive manipulation in order to obtain speed, heading, or altitude controls.

  The Action Condition is thus satisfied with the delivery or acceptance of a 4DT

[7] Based on the TBO ConOps, the responsibility is most likely allocated to automation.

by the ANSP. The Action Condition is satisfied completely with the assumption that the ANSP can issue a 4DT for every aircraft in its jurisdiction (and that the aircraft can actually receive and execute the 4DT). For mixed equipage, the Action Condition is not fully satisfied in the TBO ConOps because not every aircraft in the airspace can be issued a 4DT. Likewise, in delegated airspace, the ANSP does not issue commands to self-separating aircraft under nominal conditions.

In terms of the conformance monitoring, the action condition is satisfied by the generation of a new, closed 4DT based on the current (non-conforming) aircraft state or by a traditional clearance that causes the aircraft to return to its original 4DT parameters.

- *Model Condition* — what states of the process must the controller ascertain? How are those states related or coupled dynamically? How does the process evolve?

  As described in the *Action Condition* analysis, the most basic states that the ANSP must understand in order to control the airspace are each aircraft's position and velocity. The TBO ConOps chapter on conformance monitoring specifies several aspects of the conformance model, for example tracked versus intended aircraft state and additional information necessary to make predictions about intended state. What is not clear, in part because the concept is ill-defined with respect to human operators, is how a model—in this case a "mental model"—of a 4DT can be supported by a human agent.

  The ConOps implies a binary output from the conformance monitor, for example a Yes/No conformance alert. This approach only supports a primitive model of conformance. This binary type of model, and the type of sensing that results, does not align with the action and goal conditions described above. That is, the control agent is expected to issue a 4DT or clearance, as a function of some policy or algorithm, but conformance feedback (i.e. Yes/No, see also *Observability Condition* below) alone does not provide the information necessary to carry out an action that spans multiple dimensions.

  Clearly TBO automation will have access—and presumably the ability—to assess current and predicted states, but it is unclear how this approach supports appropriate human behavior. The law of requisite variety [Ashby, 1957] suggests that

a binary action condition is only appropriate for processes of two dimensions. Furthermore, a binary model condition can only support an action space of two dimensions.

Based on this analysis, it is unclear whether conformance monitoring satisfies the Model Condition. Because the model should support the execution of a goal and implementation of an action, conformance monitoring should (a) aide in determining whether spacing goals are being met and (b) assist both the TBO automation and the air traffic controller in generating new trajectories or clearances. If the air traffic controller is expected to issue clearances to non-conforming (or even conforming) aircraft, (s)he must be provided information about not only which aircraft do not conform but also the degree to which they do not conform and how it affects separation goals. Perhaps most importantly, and this is a distinguishing feature of human versus computer control agents, an air traffic controller will perform much better if (s)he knows *why* the aircraft is not conforming [e.g. Rasmussen, 1986; Leveson, 2000b; Dekker, 2013].

- *Observability Condition* — how does the controller ascertain the state of the system? Are the sensors adequate or appropriate given the process dynamics?

  From the perspective of the control agent, conformance monitoring alone does not satisfy the Observability Condition. Given the Goal Condition (Merging, Sequencing, and Spacing) and Action Condition (4D Trajectories, Traditional clearances), more is required than simply an alert about non-conformance. TBO automation will (presumably) have access to the process variables that underlie conformance monitoring, and thus the observability is satisfied by direct access to current and intended aircraft state information.

  Given the fact that ATC will still be able to issue traditional clearances, it may be assumed that the controllers will have access to information similar to that used in current and past operations. It is unclear how existing tools will help air traffic controllers manage 4D trajectories, however. Thus, conformance monitoring does not satisfy the Observability Condition for human operators or even higher level decision support tools or automation that do not have direct access to aircraft state and intent.

> The TBO ConOps emphasizes the importance of achieving the contracted 4D trajectory, which is the essence of conformance. Conformance monitoring represents one important feedback mechanism for determining if an aircraft is maintaining a "set point".
>
> In control theory, a robust control system is one that can reject a wide range of disturbances and maintain a set point. Non-conformance—the inability of an aircraft to achieve or maintain a set point in four dimensions—could be due to any number of disturbances, and the type of disturbance will effect the control policy. For example, a medical emergency to one of the passengers (or pilots) should elicit a different response than an unexpected change in wind direction. Again, it is important to know *why* the aircraft is not conforming, which could have an impact on both the action and goal condition, as well as the model condition.

While the TBO ConOps does not specify the role of human operators and their interface with TBO automation in great detail, it explicitly states in the description of conformance monitoring and elsewhere that human operators will be part of airspace management in the future. In fact, the TBO ConOps states that human operators will be able to issue traditional clearances. Reasons for providing a vector, for example, are because of non-conformance, conflicting trajectories, or the inability of TBO automation to converge on a viable trajectory. The human operator therefore needs more than just a binary "Yes/No" feedback mechanism to make decisions from a near-infinite set of alternatives. Currently air traffic controllers achieve this with radar, progress strips, and other decision support tools. However, one of the goals for TBO is to increase efficiency, and reducing current margins may render existing tools and mental models obsolete.

If the system will allow human operators to give clearances in order to solve problems with non-conformance, along with other issues, then the system must also support their ability to effectively issue these clearances. Although conformance monitoring is not the only tool that air traffic controllers will use for TBO, it is proposed as a primary source of information. The above analysis of the four control conditions calls into question whether this is appropriate. The analysis suggests that ground-based controllers must be made aware of non-conformance, but also the *degree* to which the aircraft does not conform along with information about *why* it is unable to conform. Thinking in terms of process control and completeness assists in reasoning about hu-

man roles, interfaces with automation, and possible control structures as the model in Figure 23 is refined.

This section describes the process and questions used to determine whether and how the model is complete with respect to the four control conditions. More specific scenarios and associated requirements will be presented in the next chapter. Appendix A provides additional details about causal factors and control structures related to these factors and other aspects of TBO beyond conformance monitoring. The Appendix also includes the analysis of the airborne control loop.

4.4.2 Analyzing Safety–Related Responsibilities

For every hazard there must exist at least one control agent that has responsibility for enforcing the related safety constraint. The control-theoretic corollary is as follows: for every hazardous state variable, there must exist (1) a control policy, (2) an affordance or available set of actions that can regulate the state, and (3) an observer of that state. The above analysis focused on completeness of the control loops, and now the analysis focuses on how the control components enforce (or do not enforce) safety-related constraints.

The TBO Concept of Operations emphasizes the importance of conformance and assigns responsibility to both ground and airborne elements in enforcing this constraint. But how do these responsibilities actually relate to enforcing safety-related constraints, i.e. the prevention or mitigation of hazards? How does conformance monitoring support or detract from these responsibilities and hazards? What are the roles of conformance monitoring in enforcing safety constraints?

The following analysis considers only one of the larger set of system hazards, while the appendix includes additional hazards.

Hazard—Aircraft violate minimum separation

Generally, loss of separation occurs whenever the protected airspace of any two aircraft overlap. Traditionally this has been done via a "hockey puck" model, where the protected airspace is represented by a cylinder with 5NM diameter and height of 1000' with the aircraft at its center[8]. Loss of separation occurs when any two of these virtual cylinders intersects. The TBO ConOps proposes other models of protected airspace, which will be discussed shortly.

In safety-driven design, there must be at least one control entity that is responsible for assuring that this loss of separation hazard does not occur (see equation 34 on

[8] This example is typical of en route operations, but separation minima change in terminal operations.

page 82). The *goal* of air traffic controllers, then, in traditional airspace is to generate clearances such that separation minima are always maintained.

One of the objectives prescribed in the TBO ConOps is to ensure that the aircraft conform to their assigned 4-dimensional trajectories. In addition to assuring separation, in TBO the air traffic controllers have the additional *goal* of assuring conformance. TBO thus satisfies the first general rule for Analyzing Safety-Related Responsibilities (equation 34, on page 82). That is, safety responsibility is assigned to at least one control agent for the minimum separation hazard.

The next aspect of analyzing the safety responsibilities involves identifying potential conflicts (equation 35 on page 82). With respect to conformance monitoring and loss of separation, the system must ensure that the respective goals do not cause conflicts. Does the TBO ConOps guarantee that such a condition does not, or cannot exist?

There is a conflict with safety responsibilities if there exists an action that can simultaneously result in the loss of separation hazard and fulfill the conformance condition. Such an action is possible if there are any aircraft (or any other debris or hazardous situation) in the presence of the intended aircraft trajectory or conformance volume.

Any argument against the previous statement must also make the relatively strong assertion that there will never be any conflict along the protected trajectory of an aircraft. An obvious rebuttal to this argument might go as follows. If aircraft α is not conforming and must "work to close" trajectory, it is equally plausible that aircraft β is in the same situation. It also plausible that aircraft β is now on the very trajectory that α is attempting to regain. There are many other scenarios, such as non-TBO aircraft along the trajectory, inclement weather, aircraft emergencies, and other factors. The goal of assuring conformance must be tempered by the larger goal of assuring separation, as well as preventing other hazards.

Appendix A.4 presents a more formal analysis associated with the preceding discussion, which extends the mathematical foundation developed in Chapter 3.

Clearly there are many good reasons for conformance, as well as conformance monitoring, and some of these reasons are related to safety. For example, if the ANSP is planning a crossing trajectory behind the trajectory of another aircraft, it must assure that this latter aircraft does not lag behind its assigned trajectory. However, safety-driven development assumes that worst-case scenarios arise, such as attempting to

close a 4D trajectory while another aircraft is on that trajectory. The key is identifying the scenarios and then reasoning about (a) how to prevent them and (b) how to mitigate the scenarios if they occur. The following chapter includes a set of example requirements related to the scenarios described above.

4.4.3 Coordination and Consistency

Conformance monitoring is intended to assure consistency among the various actors in TBO, including the ANSP, flight crews, and operating centers. Despite its intent, it is actually a source of potential inconsistencies and lack of coordination.

Consider again a conformance model, generalized from the development on page 121. Conformance *monitoring* is a mapping from surveillance and other data to a binary (or discrete set of) signal(s) that indicates whether an aircraft conforms to the desired trajectory. Conformance *alerting* is a function of current surveillance data in four dimensions and desired aircraft state in four dimensions, some allowed tolerance volume, and an "Alert Parameter".

This mathematical formulation of conformance monitoring and alerting (equations 47–50) helps identify issues with coordination and consistency, in at least three ways. First, the mapping is a function of an "Alert Parameter". This parameter is available to any agent with a conformance monitor, ground or airborne, and is thus independent and potentially inconsistent. For example, the ground controller sets an alert parameter for his or her own monitor, while the flight crew independently does so for their monitor.

The TBO ConOps does not describe or specify the rationale for including this function, but it may be assumed that it is to counteract alarm overload or over- and under-sensitivity. Furthermore, the TBO ConOps does not specify what the alert parameter entails with respect to human-computer interface design. The ConOps does refer to existing aircraft functions such as altitude alerts for the airborne monitor, but it is unclear how either the ground or airborne agents will "set" these parameters.

In addition to independence and potential lack of coordination due to the alert function, several question arise. These questions also relate to the Model Condition (page 129) but become increasingly important when considering the coordination of multiple agents.

- What actually constitutes "non-conformance"?
- Does non-conformance mean that an alert is flagged at any instant the aircraft

leaves a (continuously updated) elliptical shape?

- Alternatively, does the monitor take an average over a specified time interval and compare that trajectory to the allowable shape?
- How often is the elliptical shape updated? How do the relevant agents receive these updates?
- Who derives the conformance model?
- Is a deviation in one direction given greater importance than a deviation in another direction?

Consider a case where the aircraft is flying the correct 3D shape but does so at the wrong speed. In Figure 24, the aircraft would be flying along the prescribed line but either fore or aft of the elliptical bubble. Alternatively, the aircraft might fly the correct speed but incorrect 3D path and would thus be somewhere perpendicular to the prescribed line. The aircraft's projection onto the line would lie within the ellipsoid. While these factors may be important for the overall efficiency objectives of TBO, they are not important, in and of themselves, for safety.

To reiterate the concepts presented in Chapter 3, safety is an emergent property and arises from the interaction among components. Even in Figure 24, non-conformance by either or both of the aircraft has different implications for loss of separation, depending on the nature and degree of the non-conformance.

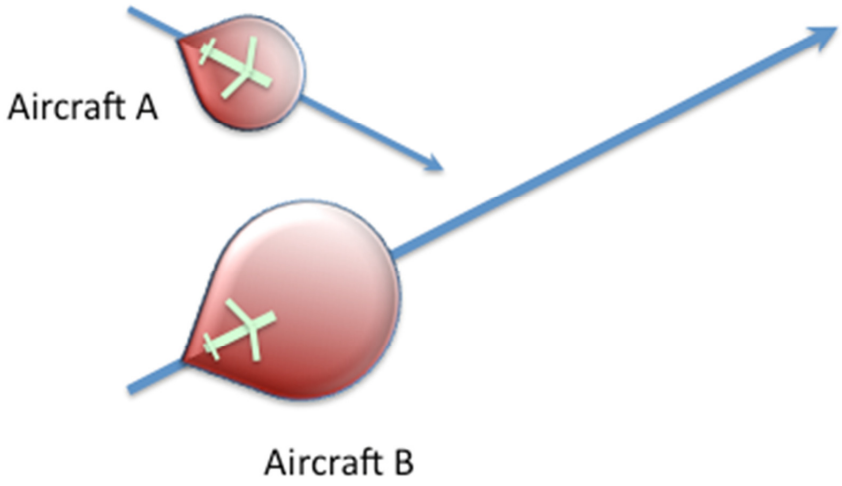


Figure 24: JPDO Proposed Conformance Monitoring Model [JPDO, 2011]

A brief, admittedly oversimplified example is given here for illustrative purposes. Latitude, longitude, and and altitude are measures of length, but their measures are not of equal importance. One pair of aircraft en route are separated in longitude by

0.1NM and vertically by 1NM, while another pair of aircraft that are separated in longitude by 1NM and vertically by 0.1NM. The former pair represents almost zero risk of collision relative to the latter pair.

This example is intended to illustrate the importance of the models that underlie the notion of conformance. The TBO ConOps suggests an elliptical shape of protected airspace, which size, shape and location would be updated at appropriate time intervals to coincide with the intended aircraft operation. Figure 24 illustrates such an elliptical model. What is unclear is which agent derives and updates the model, or if different agents can do this independently. Is the elliptical model prescribed to each aircraft by the ANSP? If so, how does the ANSP ensure that the flexibility and tolerances comply with the goals and capabilities of the aircraft or its operating centers? Alternatively, do Flight Operating Centers (Airlines) or even the aircraft manufacturers design their own conformance model?

Finally, and perhaps most importantly, the conformance monitor is ultimately used by some control entity to inform its model of the system. As discussed throughout this chapter, each agent's model is ostensibly used to generate control actions. Two or more control agents, each with different safety-related responsibilities (see Figure 16 on page 99), have a model of the same process but with different means of updating that process model. Because the various agents have different responsibilities, the different control agents not only form a potentially inconsistent model of the process but also have different means to act on that process.

The next chapter presents scenarios and requirements associated with consistency and coordination (or lack thereof), along with an option to potentially change the ConOps or pursue a slightly modified control structure.

Summary

The process described above used systems- and control-theoretic techniques to build a model of the TBO Concept of Operations, a crucial aspect of tomorrow's air traffic management system. To identify causal factors in the TBO concept, the model in Figures 21 and 23 were then queried with respect to completeness of the control components, gaps or conflicts with safety-related responsibility, and coordination and consistency among multiple control agents. These gaps and causal factors then guide the identification of requirements and potential alternative control structures, which is explored in the following chapter.

As will be shown, this approach shows promise in identifying a different class of

problems that existing hazard analysis techniques do not. Chapter 6 compares these results with those found using existing techniques.

[Page intentionally left blank]

Chapter 5
Using STECA in Early Systems Engineering

Myth: *It's software—we can fix it later (add safety, security, other "-ilities")*
Fact: *"-ilities" must be architected in, and can't be easily added later.*
adapted from [Boehm et al., 2002]

Hazard analyses or safety assessments should not be used to merely state whether the systems or components are "Safe" or "Unsafe". The results should drive the design of the system, particularly during early concept development when decisions to modify the system or identifying new requirements are most effective and least costly.

The analysis in the previous chapter is not intended to simply point out potential flaws in the concept. Rather, those results should be used to generate requirements and constraints that eliminate or mitigate against the potential design flaws, while also making undocumented assumptions explicit. The results can also be used to generate and modify the system control structure.

Once the scenarios have been identified, the key to safety-driven design is reasoning about (a) how to prevent the scenarios and (b) how to mitigate the scenarios if they occur.

5.1 Generating Safety Constraints

The following requirements are, to the extent possible, solution neutral. For example, requirements related to ANSP control agents do not assume that this function is performed by a particular entity, only that the function must be performed and the information given to the control function (which is also solution neutral in these requirements).

Section 4.4 presented the analysis in three groups: "Completeness of Control Loops" (page 125), "Analyzing Safety-Related Responsibilities" (page 141), and "Coordination and Consistency" (page 134). Requirements are presented in the same order and

are based on the relevant sub-section from the previous chapter, which is depicted in Figure 25.

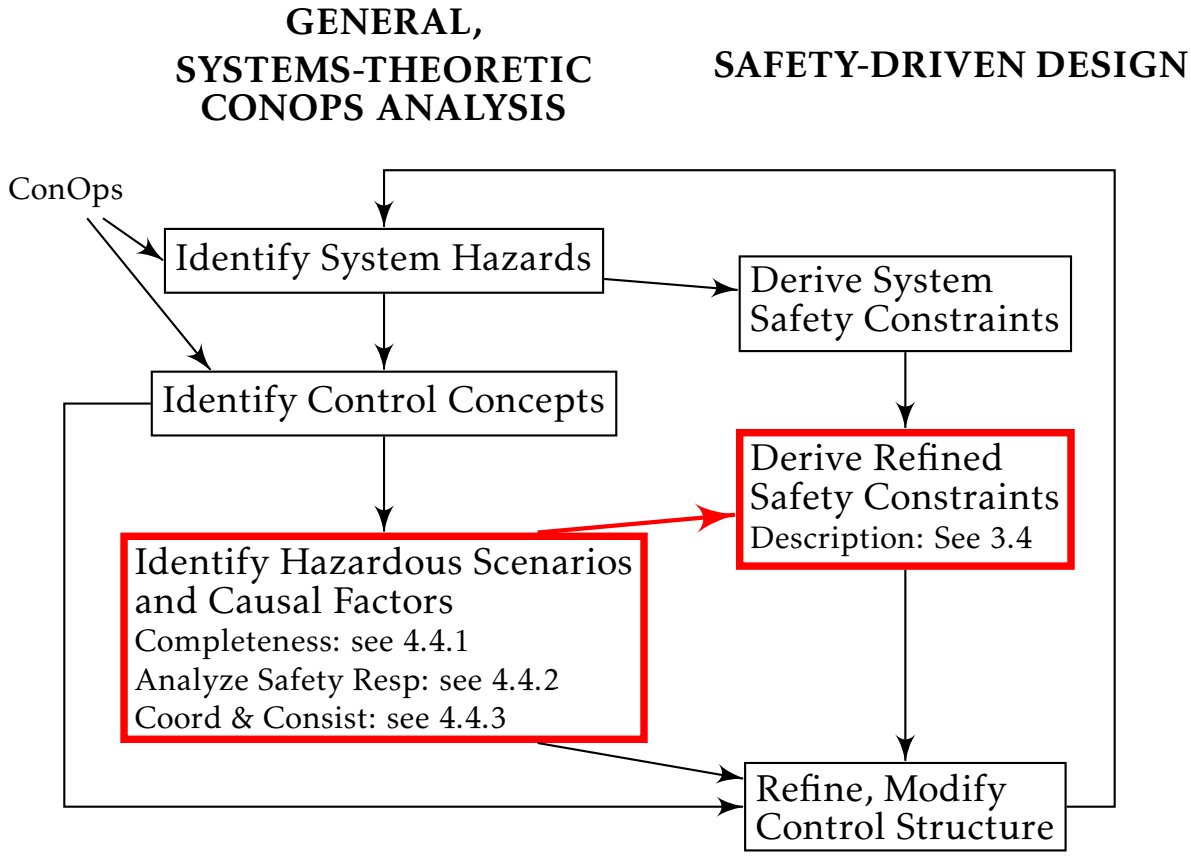


Figure 25: Methodology—Refine Safety Constraints

The example scenarios in Tables 22–24 contain links to the relevant analysis as well as a reference to the related hazard(s). The scenarios are then followed with example requirements and safety constraints that are based on both the included scenario as well as the analysis presented in the previous chapter.

Appendix A provides more details about hazardous scenarios and control structures related to these factors and other aspects of TBO beyond conformance monitoring. The Appendix also includes the analysis of the airborne control loop.

5.1.1 Completeness of Control Loops

The previous chapter describes the process used to determine whether and how the model is complete with respect to the four control conditions. More specific scenarios and associated requirements will now be presented. The example scenario in Table 22 on page 142 contains a link to the relevant analysis as well as a reference to the related hazard(s). The scenario is then followed with example requirements and

safety constraints that are based on both the included scenario as well as the analysis presented above.

5.1.2 Analyzing Safety-Related Responsibilities

Clearly there are many good reasons for conformance, as well as conformance monitoring[1], and some of these reasons are related to safety. For example, if the ANSP is planning a crossing trajectory behind the trajectory of another aircraft, it must assure that this latter aircraft does not lag behind its assigned trajectory. However, safety-driven development assumes that worst-case scenarios arise, such as attempting to close a 4D trajectory while another aircraft is on that trajectory. Section 4.4.2 describes some of the rationale and scenarios, while Section 3.3.2 describes the theory underlying the "Analysis of Safety-Related Responsibilities".

Conformance monitoring is only part of a larger set of tools and procedures, and the TBO ConOps recognizes this. Requirements related to the interaction between conformance monitoring and other aspects of TBO, and air traffic management in general, should ensure that separation and basic airmanship take precedence over conformance.

The preceding analysis (Completeness Criteria for Individual Control Loops on page 125) suggests some additional sources of feedback that may assist in fulfilling these requirements. The requirements and mitigations in Table 23 relate to the role that conformance monitoring has (or does not have) in fulfilling safety responsibilities.

The emphasis in STECA is explicitly considering hazards and safety-related responsibilities when developing and analyzing the role of any component. STECA makes explicit several potentially hazardous assumptions associated with conformance monitoring and its goals. The first hazardous assumption is that it is always desirable for pilots to close their trajectories. A second (perhaps more subtle) issue with the ConOps is that it assumes TBO automation will always generate “safe” trajectories. Further consideration of other hazards such as on-board emergencies, restricted airspace, or any other immediate desire to intentionally neglect following a trajectory reveals the need for other requirements.

[1] Conformance monitoring is described in detail in Chapter 4, as well as in the TBO ConOps [JPDO, 2011].

Table 22: Requirements Related to "Completeness of Individual Control Loops"

Scenario I. The control agent, focusing on achieving or maintaining conformance, issues commands that do not achieve the overarching goals of merging, sequencing and spacing. ↑[H-1]

(See page 126—description of Goal, Action, Model, and Observability Condition analysis)

SC-I.1. 4D Trajectories must support merging, sequencing, and spacing objectives. ↑[SC-1]

Rationale: Self evident based on definition of hazards and top-level safety constraints.

Relevant Causal Factors: This inadequate control agent may occur if the conformance model is unrelated to the models, procedures, and/or algorithms that must be used to assure separation and achieve merging and sequencing [Goal Condition, Model Condition].

SC-I.1.a. Trajectories must maintain TBD nmi separation in en route operations

SC-I.1.b. ...<other requirements should be levied here for other types of operations>...

SC-I.1.e. The distance between any two protected conformance zones shall always exceed the required separation minimum.

Rationale: Aircraft should be allowed to fly at the "edge" of their allowed trajectories, in all directions, and still maintain separation. Sub-requirements here should ensure that the conformance model is updated whenever the separation minima are updated.

Relevant Causal Factors: The conformance model is not maintained relative to updated separation requirements or across operations (e.g. on descent when separation minima are typically lower) [Model Condition].

SC-I.2. Current aircraft state, including current 3D position and velocity, must be made available to ANSP control agents.

Rationale: Current state and immediate intent is a significant (the most significant) information needed to assure separation.

Relevant Causal Factors: Overemphasis on information about conformance may also override the other types of information needed for merging, sequencing, and spacing (e.g. the location and intent of the aircraft in the airspace) [Model Condition, Observability Condition].

SC-I.2.a. <Sub-requirements here should ensure that aircraft state information takes precedence over information regarding conformance, particularly when there is a conflict> Requirements related to → Scenario III cover these factors.

SC-I.3. Flight deck must notify ANSP of their intent to deviate from trajectory, including rationale for deviation. This requirement is primarily for tactical maneuvers, which are required within <TBD minutes (or miles) of merging operation or loss of separation threat.

Rationale: Reasons for deviation help controllers (particularly human operators) in problem-solving. Because the ANSP may be negotiating with several aircraft simultaneous, the rationale for the request may help aide in the decision.

Relevant Causal Factors: The control agent is unaware of the aircraft's (potentially valid) reasons to avoid the 4DT [Model Condition, Observability Condition].

Scenario II. Separation is lost within trajectory because aircraft is flying differently than expected in intermediate parts of trajectory. Trajectory definition assures separation at specified 4D waypoints; but waypoints [Goal Condition, Action Condition]

SC-II.1. ...

Table 23: Requirements Related to "Analyzing Safety-Related Responsibilities"

Scenario III. ANSP issues command that results in aircraft closing (or maintaining) a 4DT, but that 4DT has a conflict. A conflict in these responsibilities occur when any 4D trajectory has a conflict (conflict could be with another aircraft that is conforming or is non-conforming). See also Scenario I. ↑[H-1]

(See page 141—"Analyzing Safety-Related Responsibilities" for a detailed description of this analysis)

SC-III.1. ANSP should not attempt to close the trajectories (i.e. attempt conformance) if a conflict between trajectories exists and updated trajectories cannot be generated within TBD seconds (or TBD NM of separation) ←Scenario I , ↑[SC-1]

Rationale: This scenario arises because the ANSP has been assigned the responsibility to assure that aircraft conform with 4D trajectories as well as to assure loss of separation.

See also: Many of the requirements related to completeness are also relevant here ←Scenario I

SC-III.1.a. Loss of separation takes precedence over conformance in all TBO procedures, algorithms, and human interfaces

Relevant Causal Factors: [Goal Condition]

For a human operator these requirements could be levied with respect to how the information is displayed; for automation the requirement could be levied in the algorithm in terms of the relative "weight" given to conformance versus generating new clearances.

SC-III.1.a.i. Loss of separation information must be presented to air traffic controller and/or flight crew

Rationale: feedback and information should support the goal condition [Observability Condition]

SC-III.1.a.ii. Loss of separation alert should be displayed more prominently when conformance alert and loss of separation alert occur simultaneously. This requirement could be implemented in the form of aural, visual, or other format(s).
Rationale: feedback and information should support the goal condition [Observability Condition]

SC-III.1.a.iii. Flight crew must inform air traffic controller of intent to deviate from 4DT and provide rationale ←[SC-I.3]

SC-III.1.a.iv. <similar requirements for algorithms; loss of separation should trigger response that takes precedence over conformance>

5.1.3 Coordination and Consistency

Section 3.3.3 describes the systems theory underlying Coordination and Consistency, and section 4.4.3 then provides a demonstration of how to apply that theory to TBO.

The TBO ConOps prescribes multiple conformance monitors, and these monitors are to be *independent*. This design decision is consistent with design decisions that often result from traditional hazard analyses—the TBO ConOps implicitly assumes that ensuring independence between component behaviors will lead to a safer design. The next chapter explores some limitations of traditional hazard analyses, but the point here is that STECA identifies indirect interactions, even when the ConOps assumes independence[2].

Table 24 presents scenarios and associated requirements. The scenarios relate to potential causal factors that actually arise due to these independence assumptions, and the requirements are intended to mitigate against these scenarios by ensuring that the relevant control agents are coordinated. However, modifying the concept might be a better solution, and thus Table 24 concludes with a suggested modification to the structure. The next section develops alternative control structures, using a different example from the TBO ConOps.

[2] The existing TBO Safety Assessment further asserts that this independence will serve as a mitigation against certain hazards

Table 24: Requirements Related to "Coordination and Consistency"

Scenario IV. ANSP does not issue a DE-conflicting command because it is unaware that the aircraft is not conforming or unaware that the flight crew begins taking action to "close" the trajectory. ↑[H-1]

SC-IV.1. Flight deck must notify ANSP of any changes to velocity (change in heading, airspeed, or altitude)

Rationale: Flight deck typically must request ANSP for a deviation from the filed flight plan, or from the current trajectory. This type of change to the velocity is actually due to the intent of *staying on* the trajectory, but it changes the aircraft's inertial state (3D velocity); see associated causal factors

Associated Causal Factors: due to the differences in conformance alerting models, the ANSP may be unaware of the need for change. The ANSP may have a different "Alert Parameter" setting in general, or may have adjusted the setting due to other circumstances (e.g. alarm fatigue, managing other conflicts, etc)

Note: This requirement may be levied to either the flight crew or avionics; this design choice would require further analysis of potentially dysfunctional interactions

SC-IV.1.a. Flight deck must notify ANSP that changes to velocity are due to non-conformance
Rationale: this is not a "nominal" change, e.g. a change in direction that was part of the flight plan.

SC-IV.1.b. ...

Scenario V. Flight crew does not conform to trajectory, pursuant to the ANSP conformance model. This non-conformance could arise even if the ANSP has instructed the aircraft to do so, and the flight deck has confirmed compliance.

SC-V.1. ANSP must issue commands that result in the aircraft closing on the ANSP's own conformance model. That is, the command should directly result in velocity changes that cause the aircraft to enter into desired, protected volume. This clearance is heretofore called a "Close Conformance".

Rationale: ANSP must do more than notify the flight deck that it is not conforming and instruct it to close. This requirement also assumes that the ANSP model of the overall airspace (and thus the conformance model) takes precedence over the flight deck model

Associated Causal Factors: Flight deck may try to close the trajectory to its own model, or already believe that it is conforming (and thus believe it is complying with the instruction). See also causal factors in ←[SC-IV.1]

SC-V.1.a. ANSP must be able to generate aircraft velocity changes that close the trajectory within TBD minutes (or TBD nmi).

Rationale: TBO ConOps is unclear about how ANSP will help the aircraft work to close trajectory. Refined requirements will deal with providing the ANSP feedback about the extent to which the aircraft does not conform, the direction and time, which can be used to calculate necessary changes.

SC-V.1.b. ANSP-generated clearances used to close trajectories must not exceed aircraft flight envelope ←[SC-V.1.a], ↑[SC-2]

SC-V.1.c. ANSP must be provided information to monitor the aircraft progress relative to its "Close Conformance" change request

Rationale: See "Associated Causal Factors" listed in ←[SC-V.1]

Associated Causal Factors: e.g. ANSP "turns off" or changes Alert Parameter once flight deck has confirmed that it will comply

---

Alternative Requirements & Control Structures: Such requirements could be written to eliminate the "Alert Parameter" and require that the black box models of all conformance monitors—every aircraft and on the ground—are identical. Alternative control structures could place the conformance monitoring only in the air (or, perhaps less desirably, only in the flight deck).

5.2 Generating the Control Structure

Recall the definition of a system architecture from Chapter 2. A system architecture is:

> *an abstract description of the entities of a system and the relationships between those entities* [Crawley et al., 2004],

Alternatively, architectures consist of:

> *the structure or structures of the system, which comprise...components, the externally visible properties of those components, and the relationships among them* [Bass et al., 1998].

While there are other definitions of architecture, STECA defines the architecture, or at least part of the architecture, in terms of the hierarchical control structure. The control structure developed in the previous chapter provides the *structure* of the system and the *relationships* between components, like the definition above. These relationships include not only information exchanges, but also hierarchical and authority structures. The control structure can serve as part of a broader definition of the system architecture. Chapter 3 describes the characteristics of a control structure, and Figure 26 depicts the inputs and outputs of this aspect of Safety-Driven Design.

All of the preceding model development and analysis focuses on conformance monitoring. The requirements (examples shown in Tables 22–24) and control models in Chapter 4 provide the basis for an control-theoretic description of conformance monitoring. For the purposes of demonstrating how the approach in the previous chapters can be used to develop a control structure, consider another example from the TBO ConOps.

In TBO, every approved 4D Trajectory is the result of a *negotiation* among the ANSP, the Flight Deck, and Flight Operations Centers. Appendix B documents the development of the control model of Trajectory Negotiation, which implements the methods described and demonstrated in previous chapters.

Figure 27 depicts the elements of the control model for Trajectory Negotiation, where $\mathcal{K}_y^x$ represents the action from control agent x to controlled process y, and $\mathcal{L}_y^x$ represents the feedback from process y to agent x.

A := Air Navigation Service Provider,

O := Flight Operations Center,

F := Flight Deck, Flight Crew.

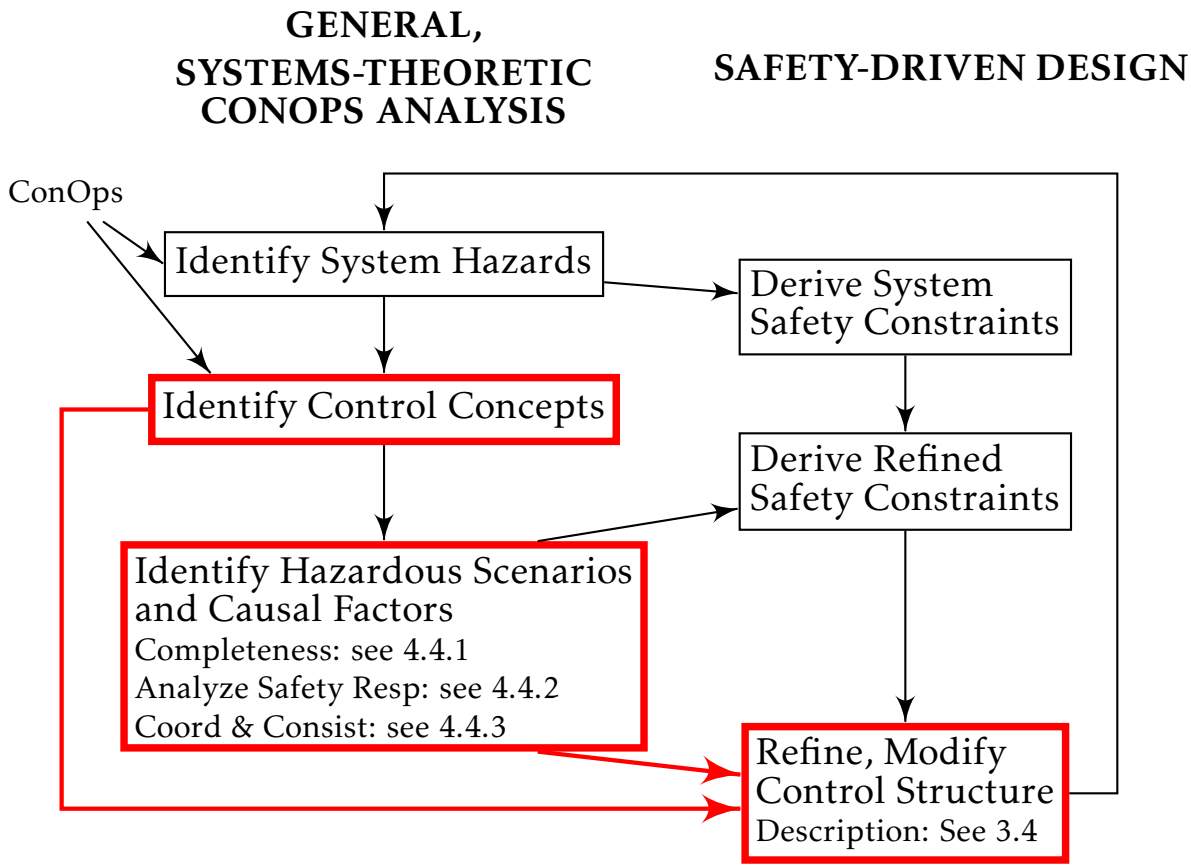


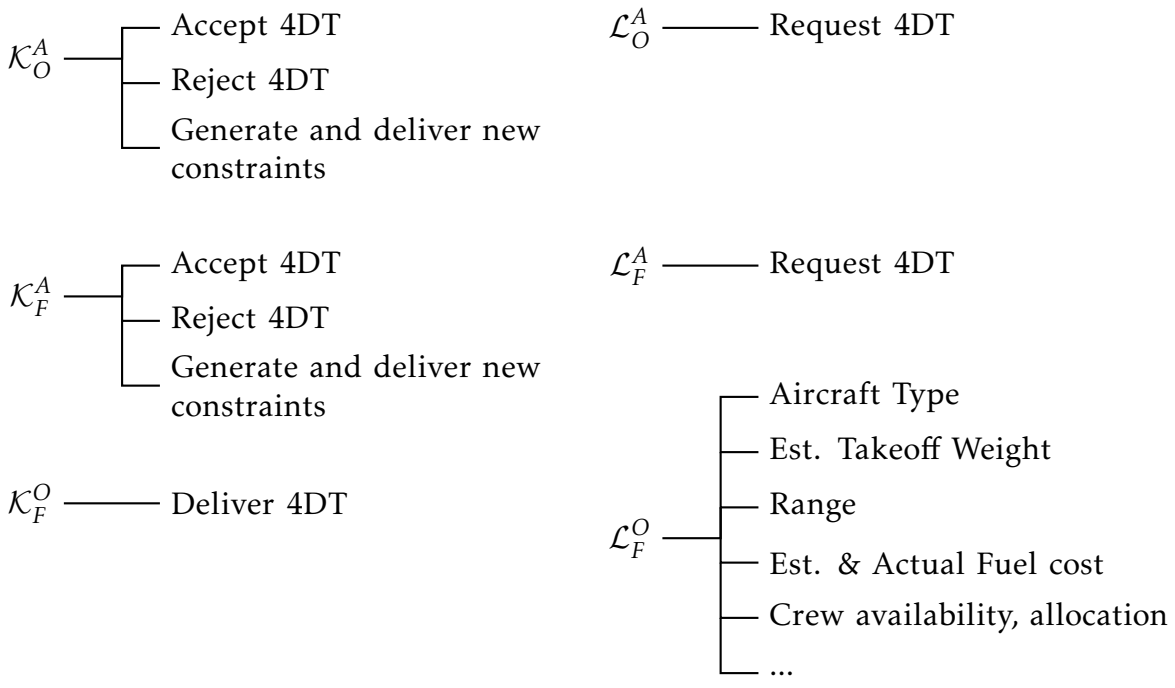
Figure 26: Methodology—Refine Safety Constraints

The model notionally shows one ANSP, two Flight Operations Centers, and eight aircraft. However, the model (and thus the airspace) may contain as many service providers, operations centers, and aircraft as necessary. Table 25 defines several of the variables in the model, which are omitted from Figure 27 in order to improve visual clarity.

By focusing on coordination and consistency, it can be seen by inspection of Figure 27 that aircraft have the potential of receiving control commands from multiple control agents. These control commands come in the form of approved 4D trajectories. Because the ANSP negotiates simultaneously[3] with flight operation centers and flight decks, even during flight in some scenarios, there is a potential for lack of coordination or consistency. There could be several solutions in the design of control algorithms that could mitigate against these inconsistencies, and requirements could be derived as such.

[3] Real-time FOC negotiation with ANSP, i.e. during the aircraft's flight, is in accordance with the TBO ConOps.

Table 25: TBO Negotiation Structure—Information Exchanges



For example, Figure 28 represents a slightly modified structure that mitigates one of the potential issues with respect to coordination and consistency. A requirement on the control algorithm might read as follows: if the ANSP begins negotiation with Aircraft X, then the ANSP must discontinue (and cannot begin) negotiations with the Flight Operations Center responsible for Aircraft X.

However, this problem is perhaps more easily solved with general, control structure modifications. One could implement a high-level requirement that the FOC stops negotiating with the ANSP for all active flights (e.g. within TBD minutes of departure). Such a requirement changes the control structure, from Figure 27 to Figure 29 on page 154, where the operations center no longer has control authority over active flights and only exchanges relevant aircraft state information. $\mathcal{I}_{\mathrm{F}}^{\mathrm{O}}$ in Figure 154 represents bi-directional information exchanges between operation centers and flight decks, where no control authority exists.

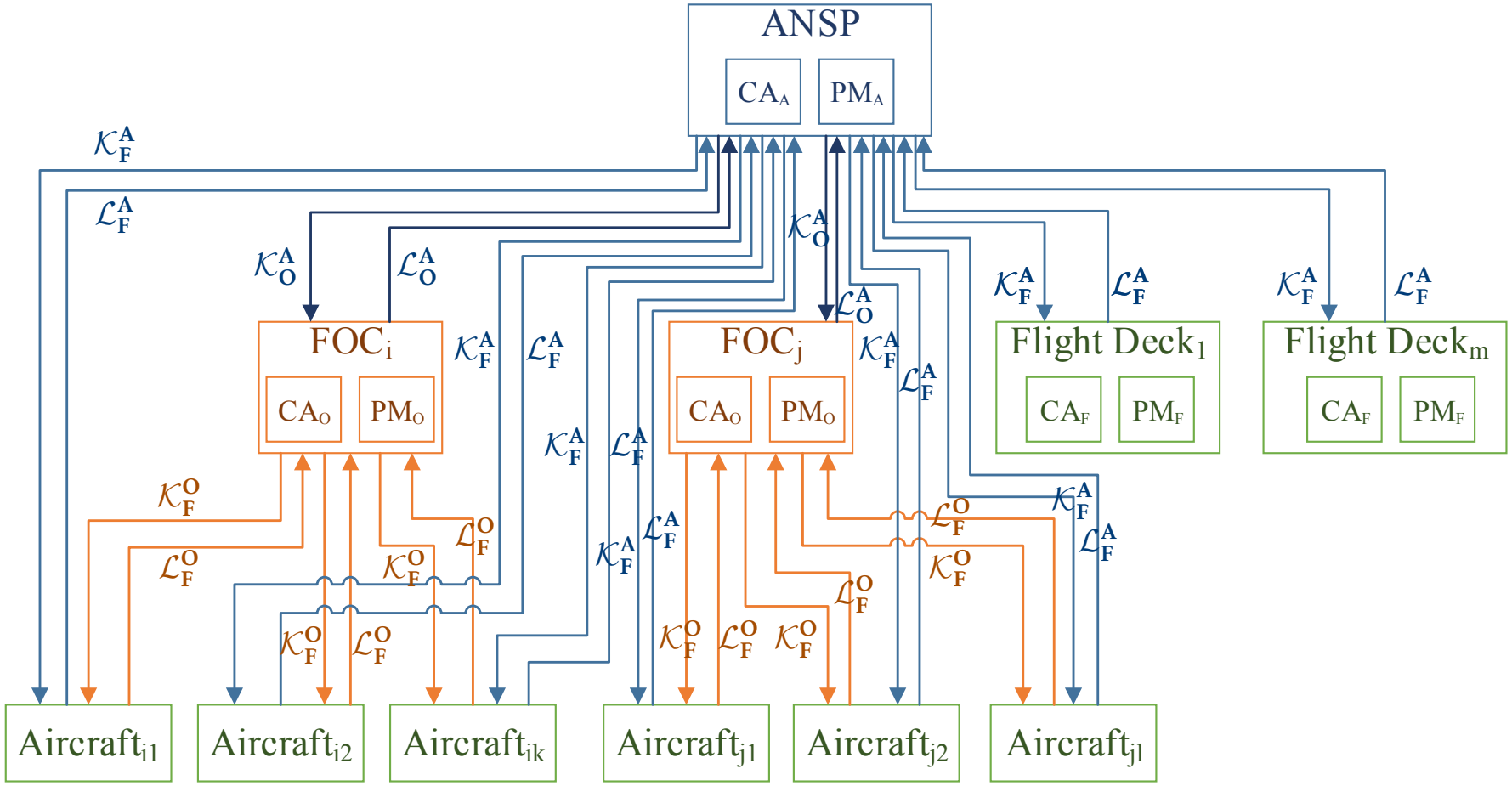
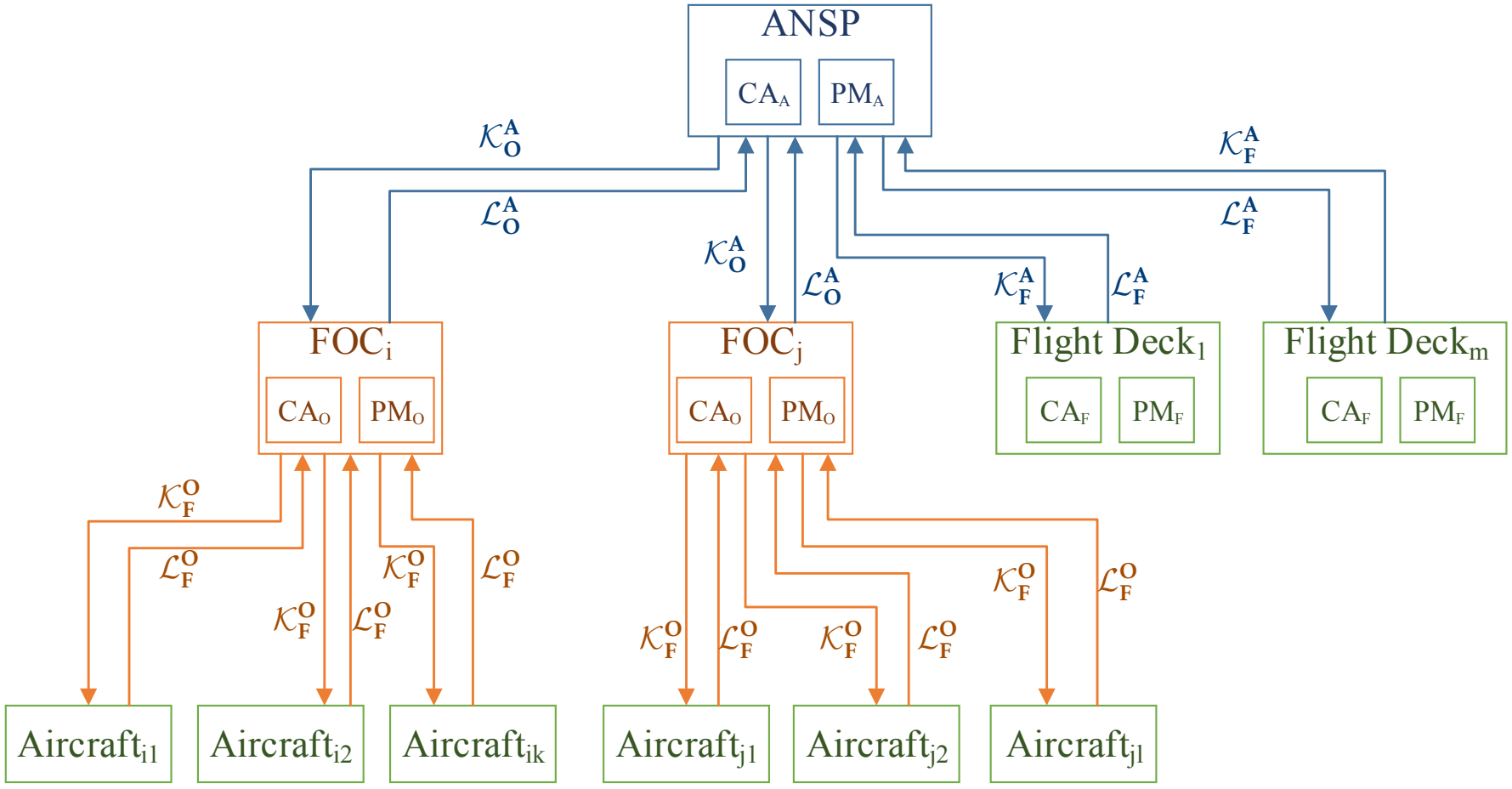
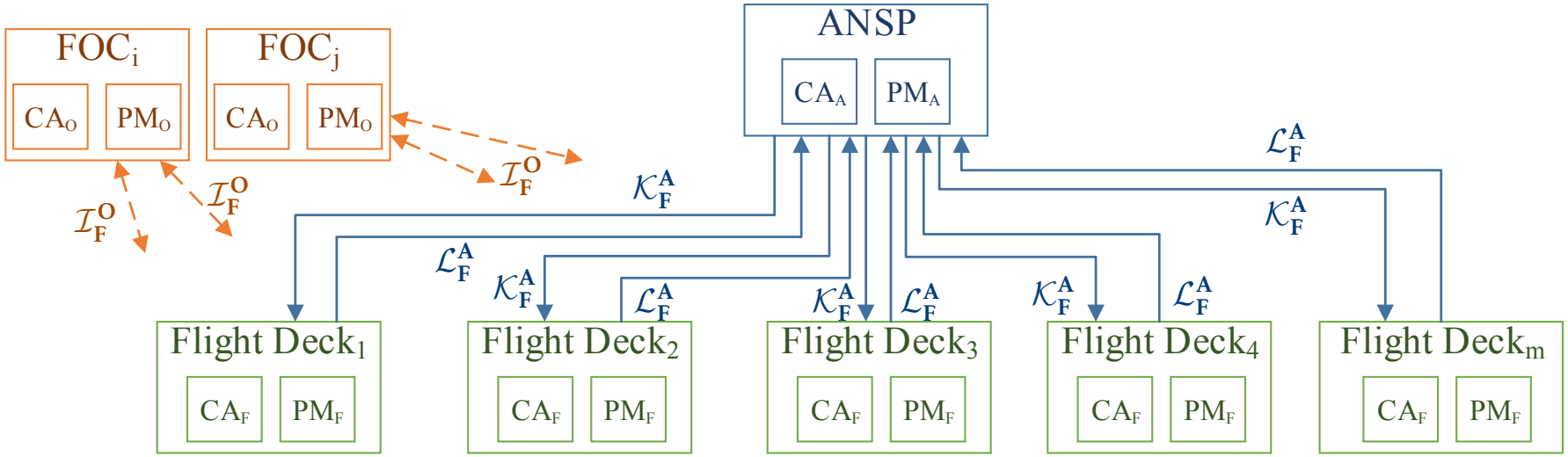


Figure 27: Nominal TBO Control Model—Trajectory Negotiation



Additional Requirement: $\mathcal{K}_F^A$ and $\mathcal{K}_F^O$ shall *not* occur simultaneously.

Figure 28: Modified TBO Control Model—Trajectory Negotiation



Additional Requirement: This becomes the active control structure within TBD minutes of gate departure.

Figure 29: Alternative Control Structure—Trajectory Negotiation

Summary

The process described above uses systems- and control-theoretic techniques to build a model of the TBO Concept of Operations, a crucial aspect of tomorrow's air traffic management system. To identify hazardous scenarios in the TBO concept, the model in Figures 21 and 23 is then queried with respect to completeness of the control components, gaps or conflicts with safety-related responsibility, and coordination and consistency among multiple control agents. These gaps and causal factors then guide in the identification of requirements and potential alternative control structures.

As will be shown, this approach shows promise in identifying a different class of problems that existing hazard analysis techniques do not. The next chapter compares these results with those found using existing techniques.

[Page intentionally left blank]

Chapter 6

Assessment of Results

The assessment of STECA involves comparing traditional hazard analysis techniques with the STECA approach demonstrated in the previous chapters. A working group of subject-matter experts applied existing techniques to develop an early safety assessment of TBO, and this assessment provides the basis for comparison.

Though the TBO preliminary hazard analysis provides the basis for assessing this new approach, the following analysis is not intended to be a critique of the TBO PHA working group. Instead, the following analysis asserts that the TBO PHA is representative of general characteristics of a traditional PHA approach and techniques used therein. See section 2.2 (on page 27 in the Literature Review) for a description of these general characteristics.

Recall what is necessary for stakeholders to develop a concept. Two significant artifacts of systems engineering, particularly in the early phases, are requirements and the definition of a system architecture. In terms of safety, requirements and architectures should eliminate or mitigate against as complete a set of hazardous scenarios as possible.

Therefore, a successful safety-driven design approach should (1) identify as many valid hazardous scenarios as possible, (2) assist in the identification of requirements and safety-related constraints, and (3) help stakeholders develop a system architecture that eliminates or mitigates hazards. In STECA, the control structure serves as part of the general system architecture definition.

6.1 Existing TBO Analysis—CapSA

The Capability Safety Assessment (CapSA) Team membership included individuals with backgrounds in aviation safety, commercial aviation, air traffic control, government regulations, planning and modeling, and aircraft manufacturing [JPDO, 2012]. Figure 30 shows the approach used by the CapSA team. The following description focuses on the two highlighted boxes in Figure 30 that most closely parallel the approach proposed in this thesis.

While there are other important steps in the CapSA approach, these steps are only effective if the hazard analysis yields comprehensive results that accurately reflect the types of accident causality found in modern systems. That is, grouping, rating, and sorting hazards and the resulting recommendations (the latter steps in Figure 30) are only as good as the hazardous scenarios themselves. The following analysis thus focuses on comparing the results of STECA with the hazard identification found in the CapSA report.

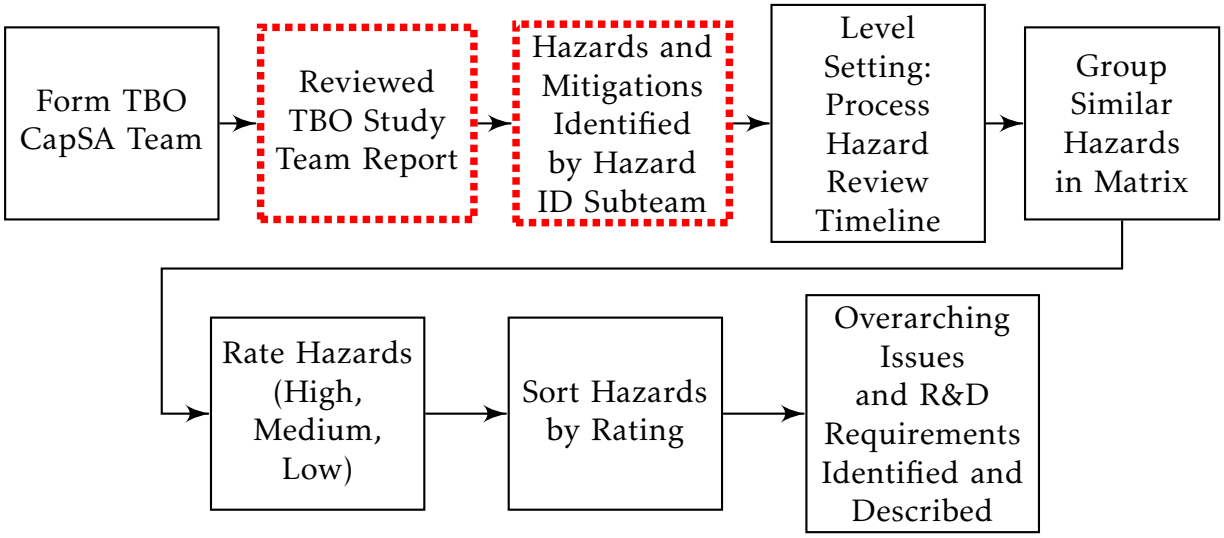


Figure 30: JPDO Safety Assessment Approach [adapted from JPDO, 2012]

The CapSA team and the example analysis presented in this thesis used the same primary resource document, the TBO Study Team Report[1] [JPDO, 2011]. "The physical, functional, and procedural elements of TBO were extracted and listed from the perspectives of a pilot in an aircraft and an air traffic controller in a ground-based Air Traffic Management (ATM) system" [JPDO, 2012, p.16]. This list formed the basis of the CapSA team's understanding of the TBO ConOps. Where there is insufficient information or detail in the TBO ConOps, the CapSA team achieved consensus on their assumptions and then did a bottom-up failure analysis of each element; each failure is considered a "Base Hazard".

The CapSA team also conducted a top-down analysis, resulting in what is interchangeably called upside down hazard trees or fault trees. The fault trees begin with six "bad outcomes", which are typically called Hazards (and sometimes Undesired or Top Events) in the safety literature [Vesely et al., 1981; Ericson, 2005]. These six events

[1] The CapSA team used Version 1.9.2 of the TBO Study Team Report as a primary resource document. This thesis used the "Final Report"; there was a 2.0 version—"coordination with members"—before the final release. The documents have minor, cosmetic differences.

include Loss of Separation, Loss of Control—In Flight, Controlled Flight into Terrain, Runway Collision, Ground Collision, and Wake Turbulence Encounter.

The final step of hazard identification compared the bottom-up failure analysis with the top-down fault tree analysis; duplicates were eliminated and a combined list becomes the final result.

6.2 Comparison of Existing CapSA to STECA

Recall from earlier in this chapter the three criteria for effective safety-driven design[2]: identify a complete set of scenarios and causal factors, derive requirements, and develop a system control structure.

The following analysis is intended to be a one-to-one comparison, to the extent possible. Tables 26, 27, and 28 contain representative artifacts of both the CapSA results and the results from this thesis.

The first column of each table represents the CapSA results and contains a Hazard Description, Causes (causal factors related to the Hazard Description), and Assumed Mitigations (what the JPDO working group identified to mitigate against the Hazard Description). The second column of each table shows the results using STECA and contain a Scenario (scenarios that lead to a hazard or undesired behavior), Causal Factors (causal factors related to the Scenario), and Requirements (constraints or requirements intended to eliminate or mitigate against the scenario).

Software Behavior

Table 26, column one, presents the risk assessment related to one of the software errors. There are at least two shortcomings in this risk assessment. First, the analysis identifies "design error" as well as "insufficient software testing" as potential causes of a software-related hazard, and the associated fault tree (Figure 31) gives no information about *why* certain types of design errors might be present. The biggest problem in developing effective software is related to specifying proper *requirement*s. Second, the associated mitigation emphasizes testing. While testing is clearly a worthwhile and necessary activity, it is unclear how testing will reveal a design error—testing is intended to demonstrate that software has implements its specified requirements. Testing also occurs very late in the development cycle when changes can be expensive and ineffective.

[2] Guiding the design so that the system is safe and effective is presumably what PHA efforts attempt to achieve.

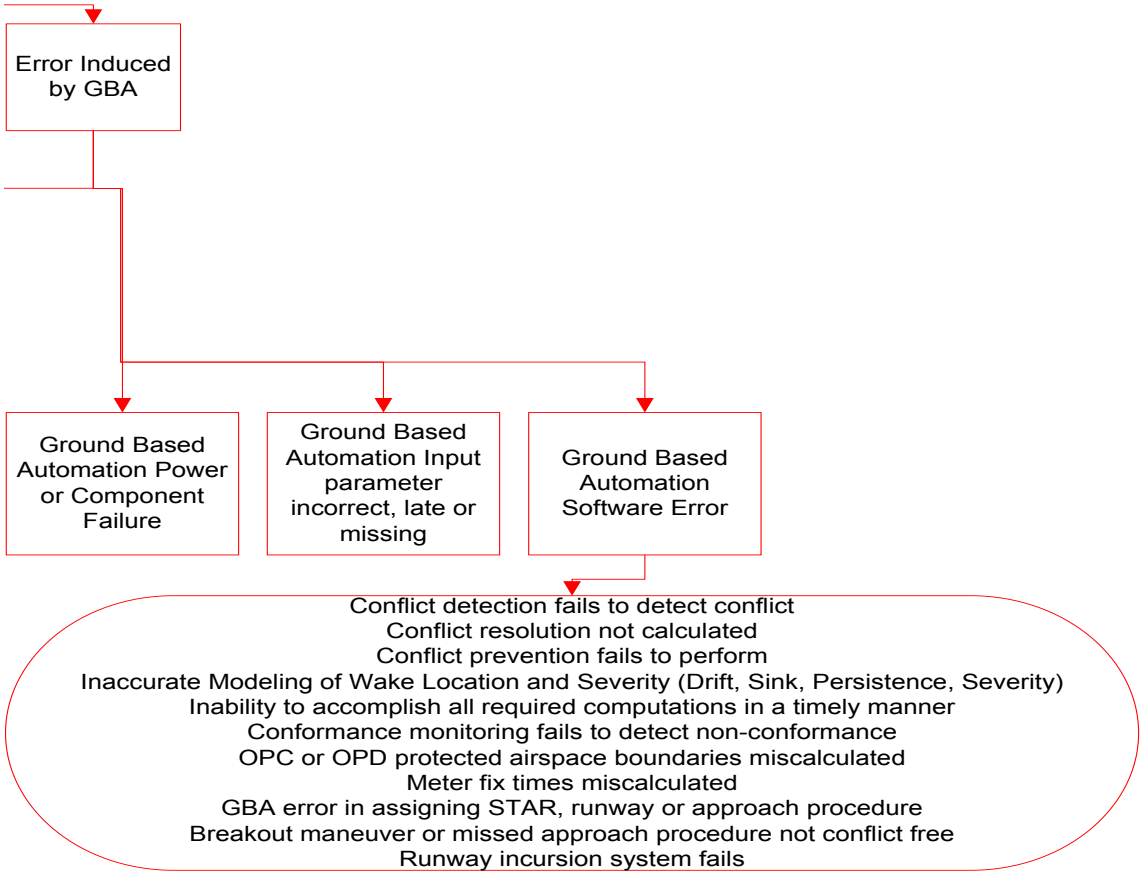


Figure 31: Software in Fault Tree Analysis [JPDO, 2012]

The analysis should help identify potential *requirements* flaws and guide the design. Because the CapSA analysis focuses on software "failures", it gives no detailed account of flawed requirements, incorrect input parameters, inappropriate behavior, or incorrect algorithm. Software "errors" are only hazardous in the larger context of the system, and the hazard analysis must be able to capture the software's interaction with other components.

The analysis in Chapter 4 identifies how hazardous scenarios arise with respect to conformance monitoring, and these factors can then be used to derive high level requirements as well as guide the design of the software and the interface with its users. The second column of Table 26 presents one scenario, associated causal factors, and a set of requirements that will drive the software design. Additional requirements are included in the previous chapter.

As an example, one potential flaw in the design of a conformance monitor is that the conformance model is not updated when the aircraft operation changes or additional aircraft join the airspace. This flaw is an example of an incorrect *Model Condi-*

tion. To mitigate against this flaw, conformance monitoring software must be provided with updated information about separation minima (further requirements would stipulate how and when the separation minima are updated).

Another potential design error identified using the safety-driven approach in this thesis involves the hazardous interaction among multiple computer systems, which will be explored later in this chapter (in the sub-section on *Component Interaction*).

Human Operator Behavior

Human error analysis in the CapSA report contains causal factors typically included in fault tree analysis and FMEA. The first column of Table 27 includes one hazard related to the ground control agent (the analysis also includes pilots), and the associated cause is "Human error". There are at least two problems with this cause. While many accidents have been attributed to human error, many behaviors that might be considered an "error" do not result in an accident and can actually be used by the operator to learn and improve his or her behavior [Dekker, 2005]. More importantly, like the factors associated with software error, the analysis omits any explanation about *why* an error occurs and how it might actually lead to a hazard. Because of this lack of definition, the assumed mitigations are equally vague.

The second column of Table 27 identifies hazardous human behavior[3] that may arise due to conflicting goals, missing information, or confusion in the way that information is presented. STECA also leads to specific requirements that can be used to develop the human-computer interface (see the last row of the table). For example, the air traffic controller's responsibility of separating aircraft should take precedence over other goals, which include assuring that aircraft remain on 4D trajectories. One way to enforce this constraint is to ensure that the information presented to controllers enforces their safety-related responsibilities.

While the failure-based approach seeks to overcome human error via automation—Assumed Mitigations in row three, column one of Table 27 assert that conformance monitoring provides a check against the ANSP mistake—STECA takes a different view. For example, STECA seeks to enforce safe human behavior by implementing constraints on the information displayed (ensuring that loss of separation information takes precedence over conformance information) as well as assisting the air traffic controller's understanding of the airspace by requiring additional information that may

[3] This sentence intentionally uses the term "hazardous human behavior" instead of "human error"

Table 26: Comparison of Software-related Results

Traditional PHA Example[JPDO, 2012]	STECA (this thesis)
Hazard Description: The software contains an algorithmic or programming error in its implementation. Software that negotiates with ground automation system on 4DT, conformance monitoring, checks for performance capability and navigates to execute the agreed trajectory.	*Scenario*: The conformance monitoring model, i.e. the protected airspace volume, is insufficient or inadequate to maintain spacing
Causes: Design error, coding error, insufficient software testing, software operating system problem; Poor I V&V	*Causal Factors*: This scenario might occur when the 4DT itself has a conflict; The conformance model is not updated to coincide with changing operations (e.g. en route vs. approach); [Model Condition, Observability Condition] The model does not ensure separation because additional traffic has joined the flow and constrained the airspace; [Model Condition, Observability Condition] Different aircraft have different conformance monitors (see "Component Interactions" section below)

(continued on next page)

Table 26: Comparison of Software-related Results

Traditional PHA Example[JPDO, 2012]	STECA (this thesis)
Assumed Mitigations: Comprehensive system testing before certification and operational approval. Other ASAS aircraft, if involved, and TCAS. Ground Based Automation would back up in some cases. See and avoid.	*Requirements*: 4D Trajectories must remain conflict-free, to the extent possible Air traffic controllers, flight crews, and/or operations centers must be notified within TBD seconds of an overlap between any two 4D trajectories Conformance volume must be updated within TBD seconds of change in separation minima Conformance monitoring software must be provided with separation minima information ...

Table 27: Comparison of Human Operator-related Results

Traditional PHA Example [JPDO, 2012]	STECA (this thesis)
Hazard Description: ANSP makes mistake during manual data load into GBA when negotiating a strategic change to the 4DT	*Scenario*: ANSP issues command that results in aircraft closing (or maintaining) a 4DT, but that 4DT has a conflict.
Causes: Human error	*Causal Factors*: This scenario arises because the ANSP has been assigned the responsibility to assure that aircraft conform to 4D trajectories as well as to assure loss of separation. A conflict in these responsibilities occurs when any 4D trajectory has a loss of separation (LOS could be with another aircraft that is conforming or is non-conforming). [Goal Condition] Additional hazards occur when the 4DT encounters inclement weather, exceeds aircraft flight envelope, or aircraft has emergency
Assumed Mitigations: Pilot will have to accept the change; Conformance monitoring; GBA tactical separation; TCAS; Quality of Data check	*Requirements*: Loss of separation takes precedence over conformance in all TBO procedures, algorithms, and human interfaces [Goal Condition]

(continued on next page)

Table 27: Comparison of Human Operator-related Results

Traditional PHA Example [JPDO, 2012]	STECA (this thesis)
	Loss of separation information must be presented to air traffic controller and/or flight crew [Observability Condition] Loss of separation alert should be displayed more prominently when conformance alert and loss of separation alert occur simultaneously. [Observability Condition] This requirement could be implemented in the form of aural, visual, or other format(s). Flight crew must inform air traffic controller of intent to deviate from 4DT and provide rationale [Model Condition]

not be contained in a data-link (flight crews provide intent information when deviating from, or requesting a change in, the 4DT).

Component Interactions

Before directly comparing the analytical results , it may be helpful to begin by comparing how interactions are treated in the models[4] themselves. In Figure 32a, interactions are considered explicitly in the control model, and the analysis proceeds by interrogating those interactions in addition to individual component behavior. In Figure 32b, causal scenarios are comprised of "branches" of the fault tree, and for the most part these branches are independent from one another. By using the control model, the analysis in Chapter 4 showed that different users of conformance monitoring may have conflicting goals with respect to conformance and different models for how the automation behaves. The fault tree analyses in CapSA assume that errors in the air and on the ground occur independently, and the only reason for a breakdown in their interactions is due to communication link *failure*.

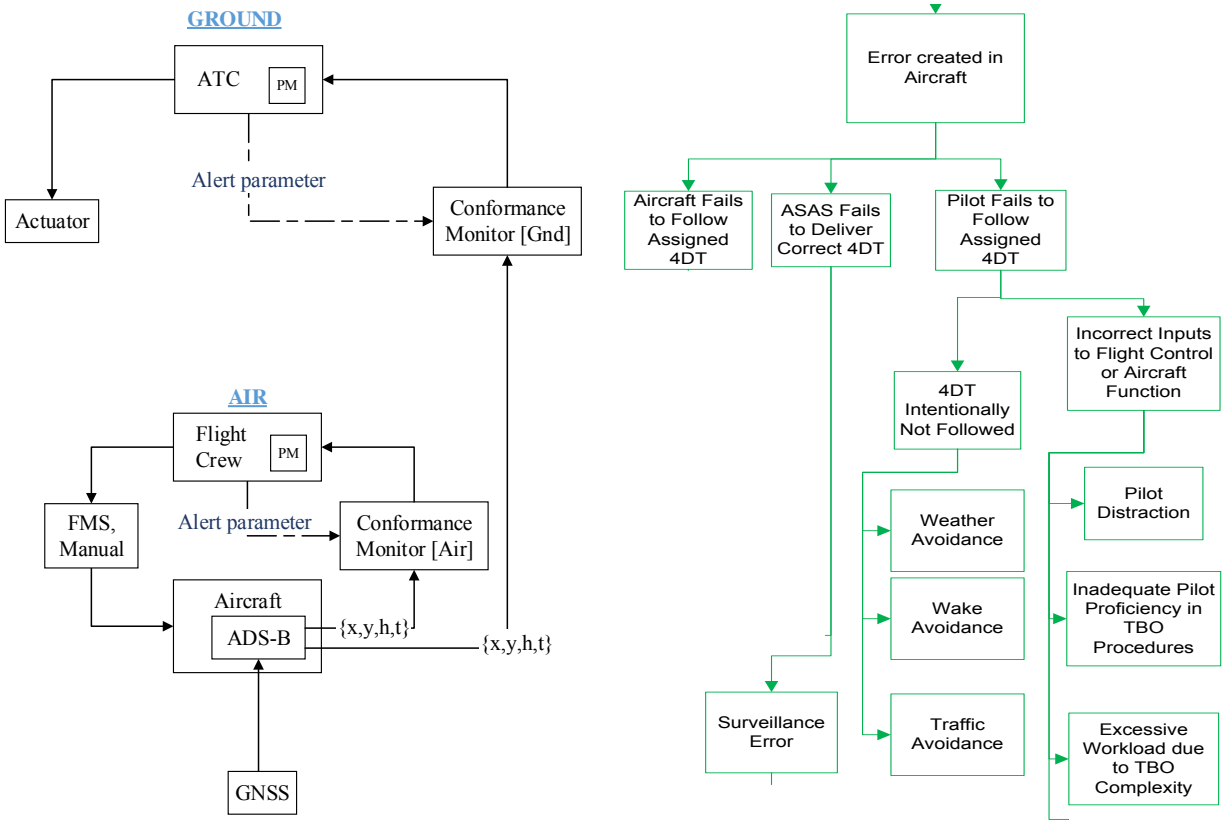
The Loss of Separation fault tree treats interactions in the following ways. There is a "connection" between automation and human error in which an error in the automation causes human error (e.g. the connection between Input Error and Error Induced by GBA in Figure 33). Though there are other similar connections between events in the CapSA fault trees, most of the causal factors are assumed to be independent.

Additionally, the FTA analysis assumes that the human error will only result from an error in the automation or because of mode confusion. While these are certainly valid concerns, they are not very instructive in terms of the design. At the risk of belaboring the point, there is no information about *why* the interaction between humans and automation may be confusing, and there are many other human factors issues beyond "mode confusion" [e.g. Rasmussen, 1986; Reason, 2000].

The other CapSA hazards related to component interaction involve communication, but the example in column one of Table 28 again emphasizes component failures (other hazards related to communication include data corruption or interference). The analysis notably omits cases when all components behave nominally but their interactions still result in hazardous behavior.

Table 28 compares specific results side-by-side. STECA finds subtle interactions

[4] Due to space and visualization constraints, Figure 32 includes simplified depictions of both the control structure (a) and fault tree (b). See Figure 24 (page 135) and Figure 33 (page 171) for representative depictions of hierarchical control structures and fault trees, respectively.



(a) Control structure

(b) Fault Tree

Figure 32: Comparison of Interactions in Analytical Models (see footnote 4)

that go far beyond failures in direct communication. While the independence assumptions in the TBO ConOps are used in the CapSA report as a possible mitigation, STECA shows how this very independence can actually become a source of hazardous behavior. Scenarios in column two of Table 28 illustrate the potential to lead to dysfunctional or hazardous component interactions, and the associated factors seek to explain *how* and *why* such a scenario might arise.

In addition to the requirements generated from this scenario, the previous chapter described how these results can be used to change the control structure (and thus the architecture and design of the system).

Summary

The preceding analysis represents a general comparison of STECA to the approach typically used to perform a preliminary hazard analysis. STECA provides a more complete analysis because it explicitly considers component interactions and uses a more sophisticated model of human and software behavior.

Table 28: Comparison of Component Interaction-related Results

Traditional PHA Example [JPDO, 2012]	STECA (this thesis)
Hazard Description: The aircraft communications link used to send ADS-B messages fails	*Scenario*: Flight crew does not conform to trajectory, pursuant to the ANSP conformance model. This non-conformance could arise even if the ANSP has instructed the aircraft to do so, and the flight deck has confirmed compliance.
Causes: Equipment failure	*Causal Factors*: Flight deck may try to close the trajectory to its own model, or already believe that it is conforming (and thus believe it is complying with the instruction). [Model Condition] Ground and flight deck have independent Alert Parameters ANSP "turns off" or changes Alert Parameter once flight deck has confirmed that it will comply
Assumed Mitigations: Redundant equipment; Primary radar; Operational procedures for ADS-B failure	*Requirements*: ANSP must issue commands that result in the aircraft closing on the ANSP's own conformance model. That is, the command should directly result in velocity changes that cause the aircraft to enter into ANSP desired, protected volume.

(continued on next page)

Table 28: Comparison of Component Interaction-related Results

Traditional PHA Example [JPDO, 2012]	STECA (this thesis)
	ANSP must be able to generate aircraft velocity changes that close the trajectory within TBD minutes (or TBD nmi). *Rationale*: TBO ConOps is unclear about how ANSP will help the aircraft work to close trajectory. Refined requirements will deal with providing the ANSP feedback about the extent to which the aircraft does not conform, the direction and time, which can be used to calculate necessary changes. ANSP must be provided information to monitor the aircraft progress relative to its "Close Conformance" change of clearance

This thesis has demonstrated how STECA can be used to identify missing and conflicting information in the Concept of Operations and then to use this information to derive requirements and generate a hierarchical control structure. Deriving this engineering information during conceptual design is vital to the successful implementation of tomorrow's complex systems.

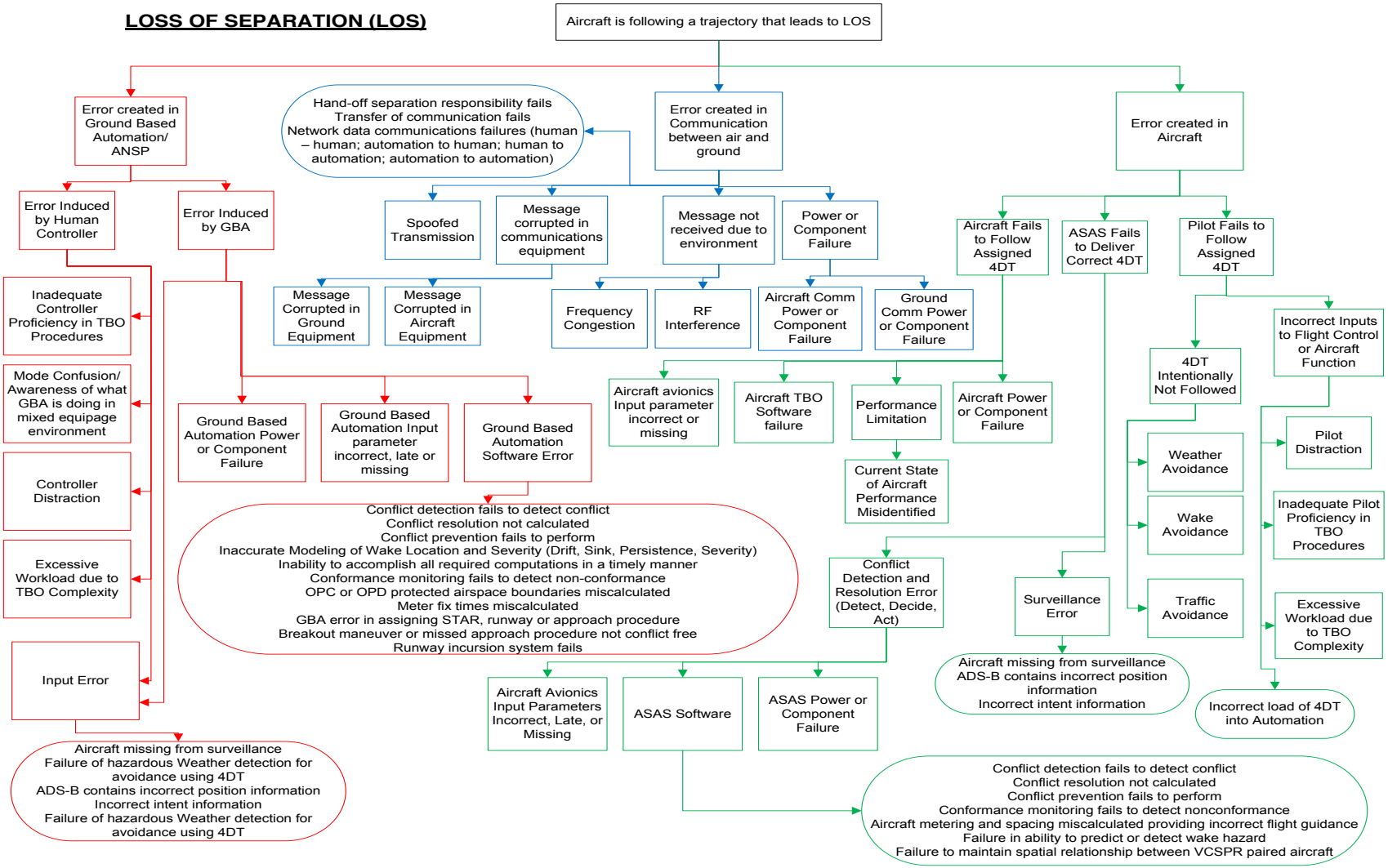


Figure 33: Loss of Separation Fault Tree Analysis [JPDO, 2012]

Chapter 7
Conclusions

This thesis has presented a new approach, called systems-theoretic early concept analysis (STECA), for performing hazard analysis on a concept of operations and a safety-driven approach to concept development. STECA is based on systems- and control theory and its usefulness and practicality is demonstrated on an important aerospace application, called Trajectory-Based Operations.

Before describing STECA, Chapter 3 begins by describing general systems theory, upon which the rest of the thesis is built. Chapter 3 then presents a practitioner-oriented set of steps and heuristics for developing and analyzing models along with a rigorous theoretical development for academics and systems theorists seeking to advance the state of the art.

STECA is based on two basic steps. The first step involves recursively applying control-theoretic concepts using guide words, heuristics, and feedback control criteria to parse an existing ConOps document, resulting in the development of a hierarchical control model of the concept. The second step—analysis—consists of examining the resulting model with the explicit goals of identifying hazardous scenarios, information gaps, inconsistencies, and potential trade offs and alternatives. That is, the analysis identifies incompleteness or gaps in the control structure, assures that all safety-related responsibilities are accounted for, and identifies sources of uncoordinated or inconsistent control.

The model development process itself provides a traceable database that can easily be queried and referenced back to the original concept documentation.

Chapter 4 demonstrates the principles developed in Chapter 3 on a real example concept being developed under the FAA's NextGen program. The model development and analysis yields important results, including several undocumented assumptions and scenarios associated with software behavior, potentially hazardous assumptions about human operator responsibilities, and potentially unsafe interactions among the various entities involved in TBO.

While Chapter 4 shows how the modeling effort and interrogation techniques can

identify hazardous scenarios, Chapter 5 uses those results to document specific hazardous scenarios. These scenarios then drive the identification of safety-related requirements as well as the generation of control structure alternatives.

Comparing the TBO results from Chapters 4 and 5 (and Appendix A) to existing analyses, performed by professional working groups using traditional techniques, Chapter 6 makes an assessment of STECA.

7.1 Contributions

The primary research contributions are in integrating safety earlier in systems engineering activities (Figure 34) via rigorous, systematic methods.

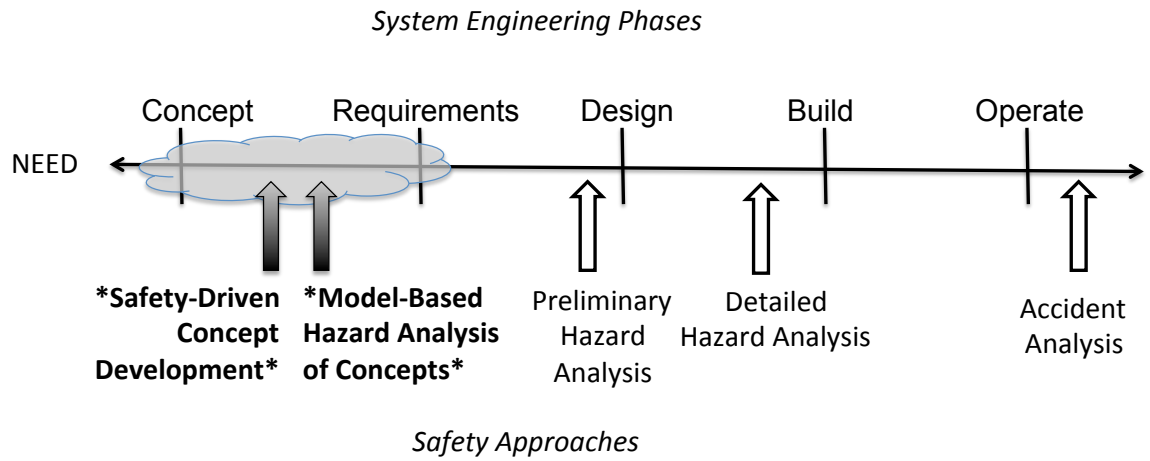


Figure 34: Research Contributions [adapted from Young, 2014]

The approach also fulfills many goals of model-based systems engineering. That is, (1) the approach contains an unambiguous language based on control theory, (2) behavior is expressed in relationships that represent the "structure" of the system in terms of a hierarchical control model, and (3) the representation can be expressed from different views according to the system hierarchy. While the approach fulfills general goals of MBSE, the safety-driven nature of the framework achieves specific objectives related to ensuring that safety is considered while the most important design and architecture decisions are made.

Specifically, STECA:

- Applies more rigor to the concept development process. The process described in Chapter 3 is repeatable and can be applied by individuals to understand a concept in systems-theoretic terms. Alternatively, the process can be used by

teams and working groups to build consensus on how the system should (and should not) behave, to allocate responsibilities to various actors, and to define the interactions between those actors;

- Identifies a class of hazardous scenarios that existing techniques cannot find during concept development. Most accidents and incidents in modern, computer-intensive systems arise due to factors that extend beyond component reliability. Accidents arise due to unsafe interactions among components, which include software and human operator behavior, and STECA is more powerful than existing techniques in identifying these types of interactions;
- Makes explicit the assumptions that are often undocumented or implicit during early concept generation. Because ConOps documents are typically developed by subject matter experts, many details that are obvious to a particular expert may seem obvious and thus go undocumented. The model development approach in this thesis forces many of these assumptions to be made explicit, and often the various subject matter experts who generate the ConOps actually make competing or inconsistent assumptions about various aspects of the system. Chapter 4 demonstrates how some of these inconsistencies can be identified.

7.2 Future Work

There are many potential paths of future research that build upon this work. While this thesis demonstrated how the results of the analyses can be used to generate alternative control structures, future work should demonstrate how these alternatives can be compared. Stakeholders identify potential tradeoffs or synergies, and tradeoffs could be made with respect to safety and/or extended to other system properties.

STECA was used in this thesis to generate and analyze a model of a ConOps. An interesting area of research relates to doing the inverse, or starting from model and generating a ConOps. The fact remains that many different stakeholders use a ConOps, not only engineers, and many of these stakeholders may not be comfortable with or fluent in the terminology and theory presented in the thesis. While the research presented here ultimately results in identifying scenarios and causal factors, those very scenarios can be used to refine the concept and, perhaps automatically, map into a more traditional format for Concept of Operations.

Tools should be generated to assist in the STECA process. Existing model-based

systems engineering frameworks could be adapted or integrated into the systems- and control-theoretic processes described in Chapters 3, 4, and 5.

Although this research focuses on the early phases of systems engineering (far left side of Figure 34), the framework has potential to applied to the very last phases of systems engineering (far right side of Figure 34). That is, when systems become operational, it is unfortunately necessary in some cases to perform accident and incident investigations. Accident reports typically share similar characteristics to Concept of Operations documents—they contain informal natural language text, use informal graphical depictions of events, and are often developed by committees comprised of potentially disparate views of the system. There exists an accident analysis process, called CAST (Casual Analysis using STAMP), based on the same systems- and control theory that has been successfully applied to accidents in a variety of domains [e.g. Dong, 2012; Hickey, 2012; Spencer, 2012; Hosse et al., 2013]. However, there is not yet a rigorous way to generate the necessary models from all the different sources of data associated with any major accident, and the methods presented in Chapter 3 represent a potential way to accomplish this goal.

Acronyms

4DT	4-dimensional trajectory (3-dimensional positions and time-of-arrival at those positions)
ADS-B	Automatic Dependent Surveillance—Broadcast
ANSP	Air Navigation Service Provider
ATC	Air Traffic Control
ATM	Air Traffic Management
ConOps	Concept of Operations
DIM	Differential Importance Measure
ETA	Event Tree Analysis
FAA	Federal Aviation Administration (United States)
FMEA	Failure Modes and Effects Analysis
FTA	Fault Tree Analysis
GNSS	Global Navigation Satellite System
MAUT	Multi-Attribute Utility Theory
NAS	National Air Space
PHA	Preliminary Hazard Analysis
PHL	Preliminary Hazard List
RAW	Risk Achievement Worth
RNP	Required Navigation Performance
RTP	Required Time Performance
RRW	Risk Reduction Worth
STAMP	Systems-Theoretic Accident Model and Process
STECA	Systems-Theoretic Early Concept Analysis, based on STAMP accident model
STPA	Systems-Theoretic Process Analysis (hazard analysis method based on STAMP accident model)

TBO	Trajectory-based Operations

[Page intentionally left blank]

Bibliography

Acar, L. and Ozguner, U. (1989). Design of hierarchically distributed expert controllers for large-scale systems. In *Intelligent Control, 1989. Proceedings., IEEE International Symposium on*, pages 12–17. IEEE.

Ale, B., Bellamy, L., Cooke, R., Duyvis, M., Kurowicka, D., Lin, C., Morales, O., and Roelen, A. (2008). Causal model for air transport safety. *Modified Seventh Interim report (15 March 2007)*.

Arnold, S. (2002). ISO 15288 Systems engineering – system life cycle processes.

Ashby, W. R. (1957). *An Introduction to Cybernetics*. Chapman & Hall Ltd.

ATSB (2013). Final investigation report AO-2010-089. in-flight uncontained engine failure Airbus A380-842, VH-OQA overhead Batam Island, Indonesia 4 November 2010, Australian Transportation Safety Board, 27 June.

Ballin, M. G., Williams, D. H., Allen, B. D., and Palmer, M. T. (2008). Prototype flight management capabilities to explore temporal rnp concepts. In *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th*, pages 3–A. IEEE.

Barnett, M. and Schulte, W. (2001). The abcs of specification: Asml, behavior, and components.

Bass, L., Clements, P., and Kazman, R. (1998). *Software Architecture in Practice, 2/E*. Pearson Education India.

Birnbaum, Z. W. (1968). On the importance of different components in a multicomponent system. Technical report, DTIC Document.

Boehm, B., Kind, P., and Turner, R. (2002). Risky business seven myths about software engineering that impact defense acquisition. *Program Manager*, 31(3):74–80.

Boeing (2009). Statistical summary of commercial jet airplane accidents. *Commercial Airplanes, Worldwide Operations*.

Booton, R. C. and Ramo, S. (1984). The development of systems engineering. *Aerospace and Electronic Systems, IEEE Transactions on*, (4):306–310.

Brown, L. (2013). Press release - FAA statement.

Chadwell, G. and Leverenz, F. (1999). Importance measures for prioritization of mechanical integrity and risk reduction activities. In *AICHE Loss Prevention Symposium, Houston TX*.

Checkland, P. (1999). *Systems thinking, systems practice: includes a 30-year retrospective*. John Wiley & Sons, Inc.

CIAIAC (2006). Failure of braking, accident occurred on 21 may 1998 to aircraft airbus A-320-212 registration G-UKLL at Ibiza airport, Balearic Islands. Technical report, Ministerio de Fomento Subsecreteria, Spain.

Clements, P., Kazman, R., and Klein, M. (2003). *Evaluating software architectures*.

Cone, E. (2002). The ugly history of tool development at the faa. *Baseline Magazine*, 4(9).

Cowlagi, R. V. and Saleh, J. H. (2013). Coordinability and consistency in accident causation and prevention: formal system theoretic concepts for safety in multilevel systems. *Risk analysis*, 33(3):420–433.

Cox Jr, L. A. T. (2008). What's wrong with risk matrices? *Risk analysis*, 28(2):497–512.

Crawley, E., de Weck, O., Eppinger, S., Magee, C., Moses, J., Seering, W., Schindall, J., Wallace, D., and Whitney, D. (2004). The influence of architecture in engineering systems. In *1st Engineering Systems Symposium*. MIT Engineering Systems Division, Cambridge, Massachusetts.

Croft, J. (2005). Airbus and boeing spar for middleweight title. *Aerospace America*, 43(7):36–38.

de Weck, O. (2009). Fundamentals of systems engineering.

de Weck, O. L., Roos, D., and Magee, C. L. (2011). *Engineering systems: Meeting human needs in a complex technological world*. The MIT Press.

Dekker, S. (2005). *Ten questions about human error: A new view of human factors and system safety*. CRC Press.

Dekker, S. (2013). *The Field Guide to Understanding Human Error: Second Edtion*. Ashgate Publishing, Ltd.

Dong, A. (2012). *Application of CAST and STPA to railroad safety in China*. PhD thesis, Massachusetts Institute of Technology.

Downer, J. (2013). Disowning fukushima: Managing the credibility of nuclear reliability assessment in the wake of disaster. *Regulation & Governance*.

Drake, B. G., Hoffman, S. J., and Beaty, D. W. (2010). Human exploration of mars, design reference architecture 5.0. In *Aerospace Conference, 2010 IEEE*, pages 1–24. IEEE.

Dulac, N. and Leveson, N. (2005). Incorporating safety in early system architecture trade studies. In *International System Safety Conference Proceedings*.

Dumoulin, J. (1988). *NSTS 1988 News Reference Manual*. Kennedy Space Center, NASA.

Ericson, C. A. (2005). *Hazard Analysis Techniques for System Safety*. John Wiley & Sons, Inc., Hoboken, NJ, USA.

FAA (2008). *Safety Management System Manual*. Federal Aviation Administration Air Traffic Organization.

FAA (2012). Nextgen segment implementation plan 5.0. *Federal Aviation Administration, Washington, DC*.

FAA (2013). Nextgen implementation plan. *Washington, DC, Available: http://www.faa.gov/nextgen/implementation/media/NextGen_Implementation_Plan_2013.pdf (accessed March 18, 2014)*.

Fleming, C., Ishimatsu, T., Miyamoto, Y., Nakao, H., Katahira, M., Hoshino, N., Thomas, J., and Leveson, N. (2012). Safety guided spacecraft design using model-based specifications. In *Proceedings of the 5th IAASS Conference*.

Fleming, C. H., Spencer, M., Thomas, J., Leveson, N., and Wilkinson, C. (2013). Safety assurance in nextgen and complex transportation systems. *Safety Science, Elsevier*, 55:173–187.

Forrester, J. W. (1987). Nonlinearity in high-order models of social systems. *European Journal of Operational Research*, 30(2):104–109.

Friedenthal, S., Griego, R., and Sampson, M. (2007). Incose model based systems engineering (mbse) initiative. In *INCOSE 2007 Symposium*.

Frola, F. and Miller, C. (1984). System safety in aircraft management. *Logistics Management Institute, Washington DC*.

Fussell, J. (1975). How to hand-calculate system reliability and safety characteristics. *Reliability, IEEE Transactions on*, 24(3):169–174.

Harel, D. (1987). Statecharts: A visual formalism for complex systems. *Science of computer programming*, 8(3):231–274.

Harkleroad, E., Vela, A., Kuchar, J., Barnett, B., and Merchant-Bennett, R. (2013). Atc-045 risk-based modeling to support nextgen concept assessment and validation. Technical report, MIT Lincoln Laboratory & Federal Aviation Administration.

Haskins, C. and Forsberg, K. (2011). Systems engineering handbook: A guide for system life cycle processes and activities; INCOSE-TP-2003-002-03.2. 1. Incose.

Hawkins, R., Habli, I., and Kelly, T. (2013). The principles of software safety assurance. In *31st International System Safety Conference, Boston, Massachusetts USA*.

Hickey, J. J. P. (2012). *A system theoretic safety analysis of US Coast Guard aviation mishap involving CG-6505*. PhD thesis, Massachusetts Institute of Technology.

Hollnagel, E., Woods, D. D., and Leveson, N. (2007). *Resilience engineering: Concepts and precepts*. Ashgate Publishing, Ltd.

Hosse, R., Spiegel, D., Welte, J., Schnieder, E., et al. (2013). Integration of petri nets into stamp/cast on the example of wenzhou 7.23 accident. In *Control and Automation Theory for Transportation Applications*, volume 1, pages 65–70.

INCOSE, S. (2011). Incose systems engineering handbook v. 3.2. 2. Technical report, INCOSE-TP-2003-002-03.2. 2. October.

Irvine, D. (2007). Shooting for the moon: The new space race, *accessed August* 2013.

Ishimatsu, T., Leveson, N., Thomas, J., Katahira, M., Miyamoto, Y., and Nakao, H. (2010). Modeling and hazard analysis using stpa. In *Conference of the International Association for the Advancement of Space Safety, Huntsville, Alabama*.

Jackson, M., Gonda, J., Mead, R., and Saccone, G. (2009). The 4d trajectory data link (4dtrad) service-closing the loop for air traffic control. In *Integrated Communications, Navigation and Surveillance Conference, 2009. ICNS'09.*, pages 1–10. IEEE.

Jackson, M. R. (2010). Role of avionics in trajectory-based operations. *Aerospace and Electronic Systems Magazine, IEEE*, 25(7):12–19.

JPDO (2010). Joint planning and development office concept of operations for the next generation air transportation system. *Version 3.2 (September 30, 2010)*.

JPDO (2011). JPDO Trajectory-Based Operations (TBO) study team report. Technical report, Joint Planning and Development Office.

JPDO (2012). Capability safety assessment of trajectory based operations v1.1. Technical report, Joint Planning and Development Office Capability Safety Assessment Team.

Kapurch, S. J. (2010). *NASA Systems Engineering Handbook*. DIANE Publishing.

Klien, G., Woods, D. D., Bradshaw, J. M., Hoffman, R. R., and Feltovich, P. J. (2004). Ten challenges for making automation a a "team player" in joint human-agent activity. *Intelligent Systems, IEEE*, 19(6):91–95.

Kochenderfer, M. J., M. Edwards, M. W., Espindle, L. P., Kuchar, J. K., and Griffith, J. D. (2010). Airspace encounter models for estimating collision risk. *Journal of Guidance, Control, and Dynamics*, 33(2):487–499.

Kollewe, J. and Gabbatt, A. (2010). Qantas to replace half of its rolls-royce a380 superjumbo engines *accessedaugust*2013.

Kuchar, J. E. and Drumm, A. C. (2007). The traffic alert and collision avoidance system. *Lincoln Laboratory Journal*, 16(2):277.

Leonard, J., Barrett, D., How, J., Teller, S., Antone, M., Campbell, S., Epstein, A., Fiore, G., Fletcher, L., Frazzoli, E., et al. (2007). Team mit urban challenge technical report.

Leveson, N. (2000a). Completeness in formal specification language design for process-control systems. In *Proceedings of the third workshop on Formal methods in software practice*, pages 75–87. ACM.

Leveson, N. (2004). A new accident model for engineering safer systems. *Safety Science*, 42(4):237–270.

Leveson, N. and Reese, J. (1994). Tcas ii intent specification.

Leveson, N. G. (1995). *Safeware: System Safety and Computers*. Addison Wesley.

Leveson, N. G. (2000b). Intent specifications: An approach to building human-centered specifications. *Software Engineering, IEEE Transactions on*, 26(1):15–35.

Leveson, N. G. (2009). Software challenges in achieving space safety.

Leveson, N. G. (2012). *Engineering a Safer World*. MIT Press.

Lutz, R. R. and Carmen Mikulski, I. (2003). Operational anomalies as a cause of safety-critical requirements evolution. *Journal of Systems and Software*, 65(2):155–161.

Luxhøj, J. T. and Coit, D. W. (2006). Modeling low probability/high consequence events: an aviation safety risk model. In *Reliability and Maintainability Symposium, 2006. RAMS'06. Annual*, pages 215–221. IEEE.

McCurdy, H. E. (1993). Inside nasa: High technology and organizational change in the us space program. *New Series in NASA History, Baltimore: Johns Hopkins University Press,| c1993*, 1.

McGregor, J. (2005). Arcade game maker pedagogical product line: Concept of operations. *Version*, 2:2005.

Mesarovic, M., Macko, D., and Takahara, Y. (1970). Theory of multilevel hierarchical systems. *New York, Academic*.

Mesarovic, M. D. (1970). Multilevel systems and concepts in process control. *Proceedings of the IEEE*, 58(1):111–125.

Mesarovic, M. D. and Takahara, Y. (1975). *General systems theory: mathematical foundations*, volume 113. Academic Press New York.

Miles Jr, R. F. (1973). Systems concepts: Lectures on contemporary approaches to systems.

Miller, C. (1988). System 3 safety. *Human factors in aviation*, page 53.

Modarres, M. (2006). *Risk analysis in engineering: techniques, tools, and trends*. CRC Press.

Morari, M., Arkun, Y., and Stephanopoulos, G. (1980). Studies in the synthesis of control structures for chemical processes: Part i: Formulation of the problem. process decomposition and the classification of the control tasks. analysis of the optimizing control structures. *AIChE Journal*, 26(2):220–232.

Morari, M. and Stephanopoulos, G. (1980). Studies in the synthesis of control structures for chemical processes: Part ii: Structural aspects and the synthesis of alternative feasible control schemes. *AIChE Journal*, 26(2):232–246.

Nakao, H., Katahira, M., Miyamoto, Y., and Leveson, N. (2012). Safety guided design of crew return vehicle in concept design phase using stamp/stpa. In *ESA Special Publication*, volume 699.

Nance, J. J. (2005). Just how secure is airline security?: The swiss cheese model and what we've really accomplished since 9/11. *ABC News*, 4:12.

Netjasov, F. and Janic, M. (2008). A review of research on risk and safety modelling in civil aviation. *Journal of Air Transport Management*, 14(4):213–220.

Northrop, L., Clements, P., Bachmann, F., Bergey, J., Chastek, G., Cohen, S., Donohoe, P., Jones, L., Krut, R., Little, R., et al. (2007). A framework for software product line practice, version 5.0. *SEI.–2007–http://www. sei. cmu. edu/productlines/index. html*.

Office, U. S. G. A. (1986). *Air traffic control: FAA's advanced automation system acquisition strategy is risky: report to the Secretary of Transportation*. The Office.

O'Neill, M., Dumont, J., Reynolds, T., and Hansman, J. (2011). Methods for evaluating environmental and performance tradeoffs for air transportation systems. In *11th AIAA Aviation Technology, Integration and Operations Conference, Virginia Beach, VA, AIAA Paper*, volume 6816.

Owre, S., Rajan, S., Rushby, J. M., Shankar, N., and Srivas, M. (1996). Pvs: Combining specification, proof checking, and model checking. In *Computer Aided Verification*, pages 411–414. Springer.

Patteau, E. (2009). Sesar joint undertaking – annual report 2009.

Pereira, S. J., Lee, G., and Howard, J. (2006). A system-theoretic hazard analysis methodology for a non-advocate safety assessment of the ballistic missile defense system. Technical report, DTIC Document.

Petty, J. I. (2002). Inflight crew escape system.

Radatz, J., Geraci, A., and Katki, F. (1990). Ieee standard glossary of software engineering terminology. *IEEE Std*, 610121990:121990.

Rasmussen, J. (1986). Information processing and human machine interaction: An approach to cognitive engineering. 1986. *New York. North-Holland*, 1(2):3.

Rasmussen, J. (1997). Risk management in a dynamic society: a modelling problem. *Safety science*, 27(2):183–213.

Rasmussen, N. C. (1975). *Reactor safety study: An assessment of accident risks in US commercial nuclear power plants*, volume 4.

Ray, E. (2014). Faa order 7110.65 v air traffic control. (april, 2014). *Federal Aviation Administration, Washington, DC: Air Traffic Rules and Procedures Service*.

Reason, J. (1990). *Human error*. Cambridge university press.

Reason, J. (2000). Human error: models and management. *BMJ: British Medical Journal*, 320(7237):768.

Roland, H. E. and Moriarty, B. (2009). *Preliminary Hazard Analysis, in System Safety Engineering and Management, Second Edition*. John Wiley & Sons, Inc., Hoboken, NJ, USA.

Ross, A. M. and Hastings, D. E. (2006). Assessing changeability in aerospace systems architecting and design using dynamic multi-attribute tradespace exploration. In *AIAA Space*.

RTCA (2008). Safety, performance and interoperability requirements document for the in-trail procedure in the oceanic airspace (atsa-itp) application. *DO-312, Washington DC*.

RTCA (2011). Safety, performance and interoperability requirements document for airborne spacing flight deck interval management (aspa-fim). *DO-328, June*, 19.

SAE (1996). *ARP-4761, Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment*. Society of Automotive Engineers. S-18.

Sherry, L. and Mauro, R. (2014). Controlled flight into stall (cfis): Functional complexity failures and automation surprises. In *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2014*, pages D1–1. IEEE.

Simon, H. A. (1977). The structure of ill-structured problems. In *Models of discovery*, pages 304–325. Springer.

Skogestad, S. (2004). Control structure design for complete chemical plants. *Computers & Chemical Engineering*, 28(1):219–234.

Spencer, M. B. (2012). *Engineering financial safety: a system-theoretic case study from the financial crisis*. PhD thesis, Massachusetts Institute of Technology.

Sterman, J. D. (1994). Learning in and about complex systems. *System Dynamics Review*, 10(2-3):291–330.

Stieglitz, W. I. (1948). Engineering for safety. *Aeronautical Engineering Review*, 7(2):18–23.

Strafaci, A. (2008). What does BIM mean for civil engineers? *CE News, Tranportation*.

Stroeve, S. H., Blom, H. A., and Bakker, G. (2009). Systemic accident risk assessment in air traffic by monte carlo simulation. *Safety Science*, 47(2):238–249.

Tatjewski, P. (2008). Advanced control and on-line process optimization in multilayer structures. *Annual Reviews in Control*, 32(1):71–85.

Thomas, J. (2013). *Extending and automating a systems-theoretic hazard analysis for requirements generation and analysis*. PhD thesis, Massachusetts Institute of Technology.

Toner, K. (2011). Opportunities and challenges for technology research and development. *ATCA Technical Symposium, Atlantic City NJ*.

UK MoD (1996). *UK Defence Standard 00-56, Safety Management Requirements for Defence Systems*. U.K. Ministry of Defence.

US DoD (2012). *MIL-STD-882E, Department of Defense Standard Practice System Safety*. U.S. Department of Defense.

USDoD (1949). *MIL-P-1629 - Procedures for performing a failure mode effect and critical analysis*. United States Department of Defense.

Vesely, W., Davis, T., and Saltos, N. (1983). Measures of risk and their application. Technical report, NUREG/CR-3243, March 1983, US NRC.

Vesely, W. E., Goldberg, F. F., Roberts, N. H., and Haasl, D. F. (1981). Fault tree handbook. Technical report, DTIC Document.

Vicente, K. J. (1999). *Cognitive work analysis: Toward safe, productive, and healthy computer-based work*. CRC Press.

Vincoli, J. W. (2005). *Basic Guide to System Safety, Second Edition*. John Wiley & Sons, Inc., Hoboken, NJ, USA.

Watson, H. (1961). Bell telephone laboratories. launch control safety study. *Bell Telephone Laboratories, Murray Hill, NJ USA*.

Wiegmann, D. A. and Shappell, S. A. (2012). *A human error approach to aviation accident analysis: The human factors analysis and classification system*. Ashgate Publishing.

Wise, J., Rio, A., and Fedouach, M. (2011). What really happened aboard air france 447. *Popular Mechanics*, 6.

Woods, D. D., Johannesen, L. J., Cook, R. I., and Sarter, N. B. (2010). *Behind human error*. Ashgate Publishing Company.

Yeo, G.-L. (2010). A380 flights to los angeles unprofitable with trent 900s: Qantas *accessedaugust*2013.

Yeo, G.-L. (2013). Faa approves 787 battery changes *accessedaugust*2013.

Young, W. (2014). Stpa-sec for cyber security / mission assurance.

Young, W. and Leveson, N. G. (2014). An integrated approach to safety and security based on systems theory. *Communications of the ACM*, 57(2):31–35.

Zheng, A., Mahajanam, R. V., and Douglas, J. (1999). Hierarchical procedure for plantwide control system synthesis. *AIChE Journal*, 45(6):1255–1265.

Appendix A: Case Study — TBO Conformance Monitoring

This appendix continues the analysis from Chapter 4. That chapter focused on the ground-based component, while the following analysis focuses on the airborne component. Following the airborne conformance monitoring analysis, this appendix concludes with a formal presentation of "Analyzing Safety-Related Responsibilities" and "Coordination and Consistency" that was started in Chapter 4.

A.1 Airborne Conformance Monitoring Model

The process model for the ground-based control agent (i.e. the ANSP) is simply,

$$\mathrm{PM_G} \quad := \quad \{\mathrm{Conf}_o, \dot{x}_o\} \tag{52}$$

where Conf_o represents whether the aircraft, o for "ownship", is conforming with its prescribed trajectory, and $\dot{x}_o$ represents some prediction of the future states of the ownship aircraft. As with the analysis of ground-based systems, the flight deck-based control actions are unspecified in the TBO ConOps' chapter regarding conformance monitoring. It is also assumed here that the airborne control algorithm is a function of conformance and that the piloting function will generate an action that enforces conformance.

The conformance model proceeds in a similar fashion as above (equations 47–50). The airborne conformance monitor is a mapping

$$\mathcal{B}_{\mathcal{L}A} := \mathcal{V}_{mA} \rightarrow \mathcal{I}_{Ac}. \tag{53}$$

$\mathcal{I}_{Ac}$, is the signal going to the piloting function (i.e. the airborne control agent) regarding conformance, and the functional behavior of the sensor is

$$\mathcal{V}_{mA} = \mathcal{L}_A \times D_{c,o} \tag{54}$$

where $\mathcal{L}_A$ is a model of the airspace state and $D_{c,o}$ is the decision criteria regarding conformance of the ownship aircraft o. The "airborne", (or piloting function) conformance model is defined as the set of dynamic variables,

$$\mathcal{L}_A \quad := \quad \{z_{\mathrm{int},o}, z_{\mathrm{act},o}, \rho, T, P_r, W, E_{cm}, F_D\} \tag{55}$$

where

$$
\begin{aligned}
z_{\text{int},o} &:= \{G, C, t\}_{\text{int},o} \\
z_{\text{act},o} &:= \{G, C, t\}_{\text{act},o} \\
\rho &:= \text{Traffic density} \\
\tau &:= \text{Operation type} \\
P_r &:= \{\text{RNP}, \text{RTP}\} \\
W &:= \text{Wake turbulence model} \\
E_{cm} &:= \text{Elliptical conformance model} \\
F_D &:= \{F, z_{\text{int},o}\}
\end{aligned}
$$

with similar definitions as equation (49) but with the ownship subscript o. Criteria for determining whether the ownship aircraft conforms with its assigned trajectory are,

$$
D_{c,o} = \left\{ z_{\text{act},0} \middle| z_{\text{act},o} \notin \bar{z}_o \left(z_{\text{int},o}, E_{cm}, a_{\text{A}} \right) \right\} \tag{56}
$$

where $\bar{z}_o$ is an allowed volume for ownship aircraft o, as a function of the intended aircraft state in time, the elliptical conformance model at a given time (E_{cm}), and an alert parameter set by the airborne operator. Like equation 50, evaluation of equation (56) to True indicates that the ownship aircraft o does not conform to its assigned trajectory.

The control algorithm of the airborne control agent (flight crew) consists of—at a minimum—use of some decision on conformance alerting. That is,

$$
\text{CA}_{\text{A}} \supseteq \text{Conf}_o \tag{57}
$$

where Conf_o is obtained, according to equation 9 on page 65, via signal processing of the input signal $\mathcal{I}_{Ac}$.

A.2 System-Level Conformance Monitoring Model

The following model (Figure 35) represents all of the development in the previous section, as well as the main body (section 4.3). This model is a continuation of the conformance monitoring model developed in the main body but includes relevant details from the rest of the TBO ConOps. Relevant details to Conformance Monitoring include the types of actions necessary to close or maintain a trajectory

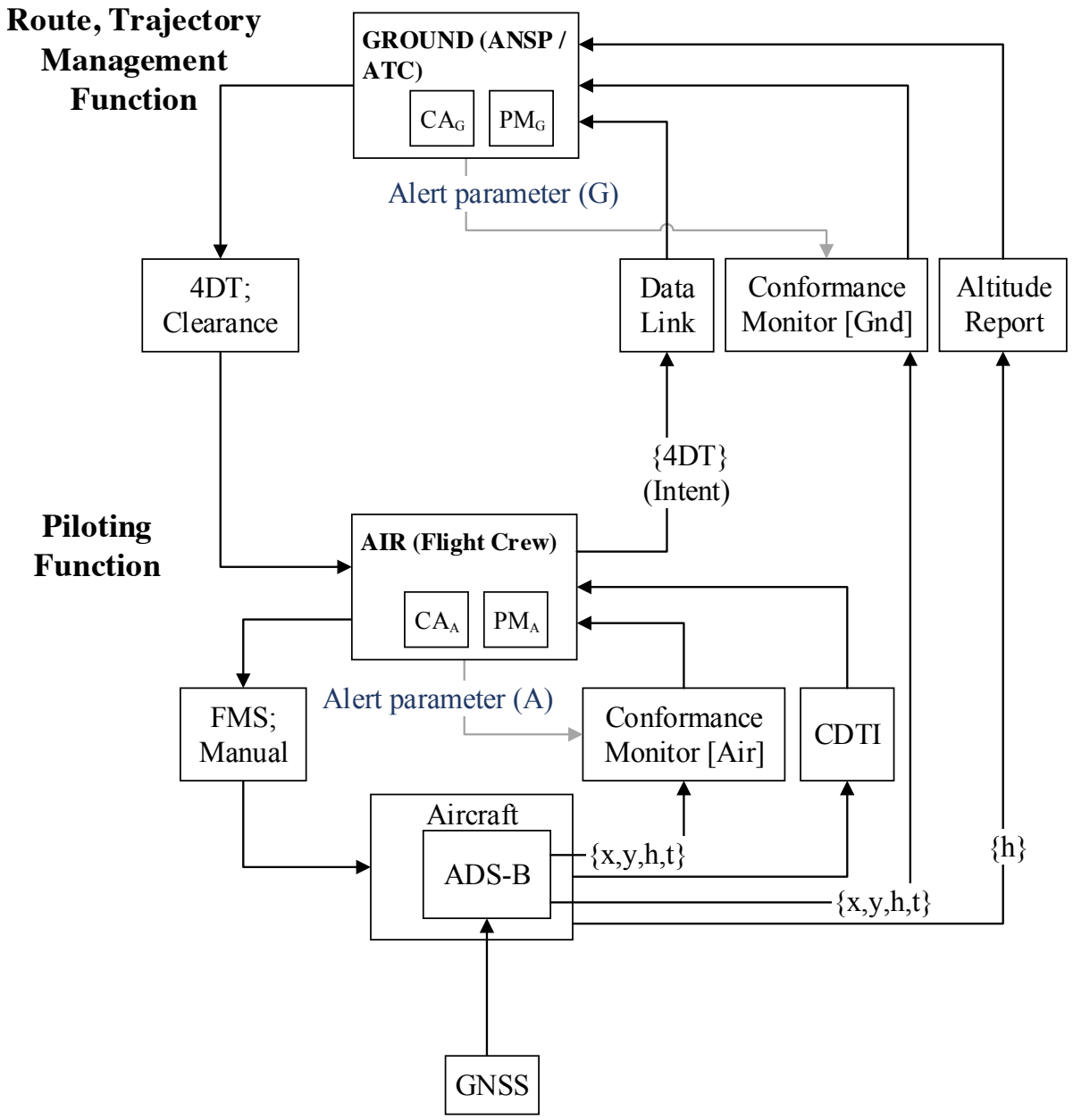


Figure 35: General Conformance Monitoring System Model

A.3 Airborne Conformance Monitoring Analysis

This section is a continuation of the model development and analysis in Sections 4.3 and 4.4 of the main body.

Flight Deck (Airborne) Control Loops

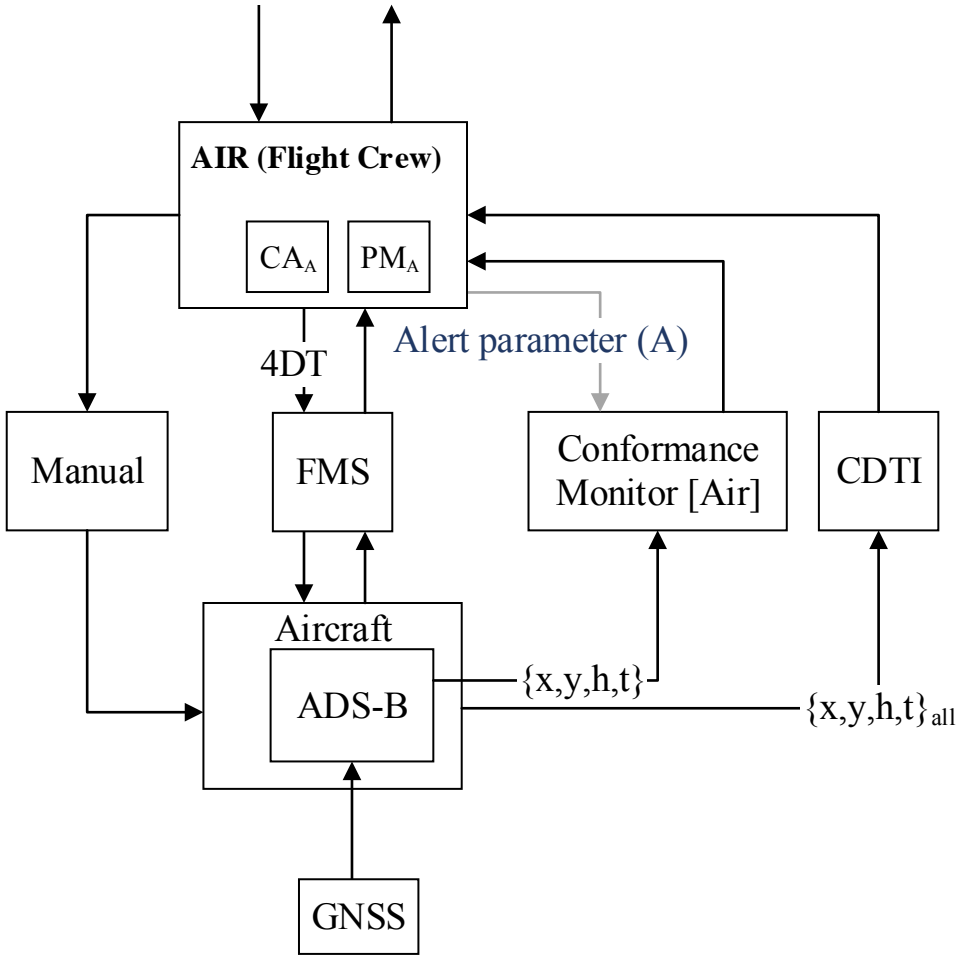


Figure 36: Flight Deck (Airborne) Control Loops

- *Goal Condition* — what are the goal conditions, and how can they violate safety constraints and safety responsibilities?

  Similar to the ANSP analysis above (on page 126), the Flight Crew is also responsible for merging, sequencing, and spacing, as well as assuring conformance. Most of the discussion in the ANSP is applicable here, because the Flight Crew and ANSP must work together to achieve flow and spacing goals and to implement achievable trajectories.

  In terms of conformance monitoring and the goal of assuring conformance, the TBO ConOps states that the "pilot must also work to close the trajectory" if the aircraft does not conform [JPDO, 2011, p.14]. The goal of conforming—closing a trajectory—should never take precedence over crew or passenger safety. Of course there are scenarios where conformance will be necessary for ensuring safety, but any assumption that conformance will *always* ensure safety is tenu-

ous. For example, if there is severe turbulence and the TBO automation cannot update the trajectory, then it should certainly not be a goal of the pilots to close the trajectory. The high level requirements should thus state that a pilot should *not* work to achieve[1] the desired trajectory if it causes loss of separation, controlled flight into terrain, flight into restricted areas, loss of aircraft control, or any of the system-level hazards.

- *Action Condition* — how does the controller affect the state of the system? Are the actuators adequate or appropriate given the process dynamics?

  The flight crew affects the state of the system by loading trajectory parameters into the Flight Management System (FMS). Elsewhere, i.e. outside of the chapter on conformance monitoring, the TBO ConOps acknowledges that a 4DT could be flown manually, with limited performance. The technical specification of a 4DT is is relatively ill-defined, and thus it is unclear how this action occurs other than to "load" the information into the FMS [e.g. JPDO, 2011, pp.11,42,58]. Other concepts provide higher fidelity information about FMS and the parameters of a 4DT [e.g. Ballin et al., 2008; Jackson et al., 2009], although issues still remain with respect to completeness of this control loop.

  The ConOps prescribes the use of the FMS to "close" the trajectory. In terms of the Action Condition, there are several issues with this approach. Not least is the fact that the FMS is the primary resource used to maintain the 4D trajectory; it follows that one cause of non-conformance is due to performance of the FMS itself. In such a scenario, the very thing that is causing the goal to be missed is then required to correct it. Furthermore, the crew will be required to input new parameters to enter into the FMS in order to close the trajectory. It is unclear how the flight crew will calculate these parameters. Again, the ConOps relies on TBO automation for many of these calculations, but it is (possibly) this very automation that caused non-conformance to begin with. Finally, it is expected that flight crews will sometimes be required to fly manually, but the TBO ConOps itself acknowledges that flight crews cannot be expected to fly with sufficient accuracy in order to achieve some of the 4DT performance parameters.

  For the merging, sequencing, and spacing Goal Condition (and given current regulatory standards), the action condition is only marginally satisfied. Currently, an aircraft can only control its own trajectory via FMS or manual inputs

[1] This discussion involves *achieving* or closing a trajectory, however similar requirements would be developed about *maintaining* or following a trajectory.

and does not have the authority to issue instructions to other aircraft. Merging, sequencing, and spacing depend on the actions of every aircraft entering the airspace or traffic flow. In the current system, spacing is ensured in part by collision avoidance systems, but these systems are part of a larger policy that implements actions on all aircraft involved in a conflict. Likewise, merging and sequencing must be part of a larger policy that coordinates actions among multiple aircraft. Such a policy is not necessarily in conflict with the TBO concept, but it is not explicitly defined. The TBO ConOps mentions "tools" that will help pilots monitor other aircraft (see below, Observability Condition), but it does not define action conditions that should result from the use of these tools.

- *Model Condition* — what states of the process must the controller ascertain? How are those states related or coupled dynamically? How does the process evolve?

  With respect to conformance, the Flight Crew analysis has similar results to the ANSP analysis. In particular, the Flight Crew will have similar human factors issues to the ANSP if they simply maintain a binary model of conformance. See page 129 for analysis of the ANSP Model Condition.

  For merging, sequencing, and spacing, the crew (and associated automation) must maintain not only a model of ownship state but also all the other aircraft entering its vicinity. This model should contain information about the current state of all the aircraft along with their intent. Because a flight crew and onboard automation do not have authority to issue instructions to other aircraft, the model must also be updated to include any changes in pilot intent, new clearances from ANSP, and non-conformance of other aircraft in the space. It is these latter factors that are relatively undefined in the ConOps, and thus the Model Condition for the airborne control loop is not completely satisfied.

  Given that the Flight Crew will still have ultimate responsibility over the aircraft[2], there are further human factors issues with assigning Merging, Sequencing, and Spacing responsibility. Problems with the Action Condition have already been described, but it is also unclear how Flight Crews will maintain a model of both the aircraft itself (and all its complexity) along with the state and intent of all the other aircraft. The Observability Condition analysis below describes one proposed approach to this problem.

[2] The TBO ConOps advocates for continuing current regulation with respect to crew responsibility: "the flight crew's authority over the aircraft's trajectory (FAR 91.3) do not change with trajectory negotiations" [JPDO, 2011].

- *Observability Condition* — how does the controller ascertain the state of the system? Are the sensors adequate or appropriate given the process dynamics?

  The observability analysis for the Flight Crew is similar to that of the ANSP on page 130. From the perspective of the flight crew and its goal of maintaining closed trajectories, conformance monitoring alone does not satisfy the Observability Condition. Like the ANSP, the flight crew must have information about the degree to which the aircraft is not conforming, which requires more than an alerting function. Many of the early proposals to implement concepts related to TBO involve time-based metering. In time-based metering, conformance is relative to a scheduled time of arrival, and feedback simply involves comparing the estimated time of arrival to scheduled time of arrival and then adjusting the speed accordingly. As 4DT operations involve more dimensions, including increasingly tight lateral and vertical constraints, the feedback will become more complex.

  For merging, sequencing, and spacing, the airborne component must have access to all aircraft that will be part of the traffic flow or entering into the ownship trajectory. As proposed, this will be done via Cockpit Display of Traffic Information (CDTI). While CDTI systems will display current state information for all the aircraft in a given vicinity, what is not clear in terms of the flight crew's Goal Condition is how the crew will discern the intent of all the aircraft. That is, all the predicted trajectories may not be directly observable via a CDTI, nor will recent, current, or future clearances or trajectory changes. All of these conditions affect the ability of a flight crew to achieve merging, sequencing, and spacing objectives.

The analysis of the ANSP control conditions concluded with a brief discussion about role allocation, interfaces, and other architectural considerations. Assuming that pilots have some separation authority, and they almost certainly will still have authority over their own aircraft per regulation, similar factors to the ANSP analysis apply here.

A.4 Analyzing the Safety-Related Responsibilities

The analysis in Section 4.4 describes the systems-thinking that goes into identifying potential flaws and scenarios associated with the ConOps. The following analysis describes the underlying mathematical formalism that can be used to more rigorously identify flaws and missing information.

Hazard — Aircraft violate minimum separation

Generally, loss of separation occurs whenever the protected airspace of any two air-

craft overlap. The loss of separation hazard is thus:

$$\mathcal{H}_s = \left\{ \mathscr{C}, \bar{z} \,|\, \mathscr{C}_i \times z_i \mapsto \bar{z}_i, \mathscr{C}_j \times z_j \mapsto \bar{z}_j, \bar{z}_i \cap \bar{z}_j \neq \emptyset \right\} \tag{58}$$

where $\bar{z}_k$ is a protected airspace volume of aircraft k around the current state $z_k = (x_k(t), y_k(t), h_k(t))$ and x, y, h represent latitude, longitude, and altitude, respectively. Traditionally this has been done via a "hockey puck" model, where $\bar{z}$ is a cylinder with 5NM diameter and height of 1000' with the aircraft at its center. Loss of separation occurs when any two of these virtual cylinders intersects. The TBO ConOps proposes other models of protected airspace, which will be discussed shortly.

Assume now that the responsibility of conformance is fulfilled; that is, an action is applied to maintain or resume conformance. This conformance condition can be defined as:

$$\mathcal{M}_c = \left\{ \mathscr{C}_k,\, V_k \,|\, \mathscr{C}_k \times z_k \mapsto z'_k,\, z'_k \in V_k \right\} \tag{59}$$

where V_k is a conformance volume of aircraft k around the desired state $\tilde{z}_k = \left(\tilde{x}_k(t), \tilde{y}_k(t), \tilde{h}_k(t)\right)$. Finally, conformance monitoring sounds an alert whenever an aircraft deviates from the desired state. See the development of equations (47) and (55) on page 189, and observe that the aircraft state, z_k, is a function of ground track and climb performance in that development. Does conformance alerting enforce safety constraints? The answer is yes and no.

Reiterating the definition from Chapter 3, let $\mathsf{P}(\mathsf{x}, \mathbb{S})$ be defined for all pairs $(\mathsf{x}, \mathbb{S})$, where

$$\mathsf{P}(\mathsf{x}, \mathbb{S}) \equiv \mathsf{x} \text{ yields } \mathbb{S}. \tag{60}$$

The predicate $\mathsf{P}(\mathsf{x}, \mathbb{S})$ is true, iff $\mathbb{S}$ is a defined condition or system state, and x is an action resulting in that condition. A safety-related conflict occurs if the following proposition does not hold:

$$\neg \exists \mathsf{c} \in \mathscr{C} \left[\mathsf{P}(\mathsf{c}, \mathcal{H}_s) \wedge \mathsf{P}(\mathsf{c}, \mathcal{M}_c) \right] \tag{61}$$

That is, there is a conflict with safety responsibilities if there is an action that can simultaneously result in the loss of separation hazard, $\mathcal{H}_s$, and fulfill the conformance condition, $\mathcal{M}_c$. Such an action is possible if there are any aircraft (or any other debris or hazardous situation) in the presence of the intended aircraft space, V_k.

Any argument against the proposition in (61) must also make the relatively strong assertion that there will never be any conflict along the protected trajectory of an aircraft. An obvious rebuttal to this argument might go as follows. If aircraft α is not conforming and must "work to close" trajectory, it is equally plausible that aircraft β

is in the same situation. It also plausible that aircraft β is now on the very trajectory that α is attempting to regain. The goal of assuring conformance must be tempered by the larger goal of assuring separation.

Hazard — Aircraft enters uncontrolled state

Although there could be any number of reasons an aircraft loses control, consider the following formal definition of the hazard.

$$\mathcal{H}_c = \{\mathscr{C}, \bar{z} | \mathscr{C}_i \times z_i \mapsto z_i, z_i \cap \hat{z}_i \neq \emptyset\} \tag{62}$$

where $\hat{z}_i$ is the allowable flight envelope of aircraft i. That is, a loss of control hazard occurs when the aircraft exceeds its flight envelope. This definition omits common causes of loss of control, but the omission is intentional. Because this analysis focuses on trajectory-based operations, the definition in equation (62) includes scenarios where the trajectory *itself* causes loss of control. There are myriad scenarios leading to loss of control, and analysis of these scenarios constitutes a lower level in the system hierarchy. Given the definition of conformance in equation (59), there is a potential conflict if the following does not hold:

$$\neg\exists \mathrm{c} \in \mathscr{C}\left[\mathsf{P}(\mathrm{c}, \mathcal{H}_c) \wedge \mathsf{P}(\mathrm{c}, \mathcal{M}_c)\right] \tag{63}$$

There is a potential conflict with safety responsibilities if an action simultaneously exceeds aircraft capability, causing loss of control $\mathcal{H}_c$, and fulfills the conformance condition, $\mathcal{M}_c$. Such an action is possible if the trajectory generation algorithm is inadequate or if it has inaccurate information about aircraft capability, aircraft status, and environmental conditions. The TBO ConOps thoroughly documents the necessary parameters for computing aircraft capability and including this in TBO automation.

A.5 Coordination and Consistency

Conformance is intended to assure consistency among the various actors in TBO, including the ANSP, flight crews, and operating centers. Despite its intent, it is actually a source of potential inconsistencies and lack of coordination.

Consider again a conformance model (generalized from the development on page 121 and included again here for reference). Conformance monitoring and alerting is a mapping

$$\mathcal{B}_{cm} := \mathcal{L}_{cm} \times D_{cm} \rightarrow \mathcal{I}_{cm}, \tag{64}$$

where $\mathcal{L}_{cm}$ is a model of the airspace state and D_{cm} is the decision criteria regarding

conformance.

$$\mathcal{L}_{cm} := \{z_{\text{int}}, z_{\text{act}}, \rho, T, P_r, W, E_{cm}, F_D\} \tag{65}$$

$$\begin{aligned}
z_{\text{int}} &:= \{G, C, t\}_{\text{int}} \\
z_{\text{act}} &:= \{G, C, t\}_{\text{act}} \\
\rho &:= \text{Traffic density} \\
\tau &:= \text{Operation type} \\
P_r &:= \{\text{RNP}, \text{RTP}\} \\
W &:= \text{Wake turbulence model} \\
E_{cm} &:= \text{Elliptical conformance model} \\
F_D &:= \{F, z_{\text{int}}\}
\end{aligned}$$

$$D_{cm} = \{z_{\text{act}} | z_{\text{act}} \notin \bar{z}(z_{\text{int}}, E_{cm}, a_{cm})\}, \tag{66}$$

where $\bar{z}$ is an allowed volume for an aircraft. Conformance *alerting* is a function of current surveillance data in four dimensions and desired aircraft state in four dimensions, some allowed tolerance volume, and an "Alert Parameter" ($z_{\text{act}}, z_{\text{int}}, E_{cm}, a_{cm}$, respectively, in equation 66).

This mathematical formulation of conformance monitoring and alerting helps establish issues with coordination and consistency, in at least three ways. First, the mapping, $\mathcal{L}_{cm} \times D_{cm}$ is a function of an "Alert Parameter", a_{cm}. This parameter is available to any agent with a conformance monitor, ground or airborne, and is thus independent and potentially inconsistent. For example, the ground controller sets an alert parameter for his or her own monitor, while the flight crew independently does so for their monitor. The TBO ConOps does not describe or specify the rationale for including this function, but it may be assumed that it is to counteract alarm overload or over- and under-sensitivity. Furthermore, the TBO ConOps does not specify what the alert parameter entails with respect to design. The ConOps does refer to existing aircraft functions such as altitude alerts for the airborne monitor, but it is unclear how either the ground or airborne agents will "set" these parameters.

Analysis

In addition to the coordination scenarios described in Section 4.4, there is another coordination problem from the perspective of the ANSP (Figure 23 on page 127). This issue arises because the ANSP receives conformance information about every aircraft, $i \in N$. Given Aircraft A and B in Figure 24, assume that Aircraft B is non-conforming.

Aircraft B is early, which represents a potential conflict because it will arrive at a crossing route with Aircraft A. While their intended trajectories do not conflict, it has become a coordination problem, particularly if Aircraft B cannot conform quickly. That is, the ANSP has to coordinate the actions given to A and B (and potentially other up- and downstream aircraft) in order to avoid a conflict. The TBO ConOps recognizes such a problem and suggests the use of tactical clearances by air traffic controllers along with collision avoidance systems.

[Page intentionally left blank]

Appendix B: TBO Model Development and Analysis

This appendix demonstrates how the methods described in Chapter 3 can be used to store the model in a database and ensure traceability to the original, natural language text. The model is stored in a different format than the format used in Chapter 4 and Appendix A. This appendix also represents a different aspect of TBO—negotiation of 4D trajectories. The graphical model (Figure 27 on page 152) is included in the main body in Chapter 5.

The following tables have six columns: (1) text quoted directly from the TBO ConOps [JPDO, 2011]; (2)–(5) are each element of the tuple $(\mathcal{S}, \mathcal{R}, \mathcal{B}, \mathcal{A})$, respectively; and (6) is a description of the updated control model. Abbreviations are used for various elements of the control model, e.g. "CA" means "Control Algorithm".

Original Text	**Source**	**Role**	**Behavior**	**Context**	**Control Model**
Quote directly from TBO ConOps	$\mathcal{S}$	$\mathcal{R} \in \{\mathcal{C}, \mathcal{K} \ldots \mathcal{P}, \mathcal{L}\}$	$\mathcal{B} \in \mathcal{B}_c, \mathcal{B}_\mathcal{K} \ldots \mathcal{B}_p, \mathcal{B}_\mathcal{L}\}$	$\mathcal{A}$:= Context, assumptions, or rationale for choice of model elements	Model Elements Abbreviations CA:=Control Algorithm PM:=Process Model

Table 30: Analysis for Negotiation with ANSP

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
From the ANSP perspective, the main emphasis will be on planning and enhanced system prediction,					
with the objective of making most strategic ATM actions prior to flight departure or, where this is not	ANSP	Controller	Generates actions		CA (policy for strategic versus tactical ['tactical' used later])
practical, before the flight is forecast to enter a block of airspace or an airport terminal area where a					PM (block of airspace, terminal, constraints); CA (determining when to act relatve to knowledge in PM)
constraint exists. The ANSP moves towards "Management by Exception," maximizing collaborative					CA ('collaborative')
pre-flight iterative planning, and refining the proposed trajectory using all available relevant					CA (what is considered 'pre-flight' versus what is not)
information as the takeoff time approaches, minimizing the need for in-flight intervention. The SESAR					PM (take off time); CA (policy for what is considered 'take off time approaches')
concept of Reference Business Trajectory (RBT) is useful here because the RBT represents the best	RBT	Actuator	Translation	Unclear if this is the proper allocation	Actuator (translate from
compromise between the flight operator's objectives for the flight and the trajectory that the ANSP can					

Continued on next page

Table 30 Analysis for Negotiation with ANSP

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
agree to support. The trajectory that exists for a flight as it commences incorporates all the known					PM (time of flight, all known constraints, trajectory, criterion for 'commencement')
constraints for that flight, providing significantly improved predictability for the flight over today.					
No matter how good the planning, and even under the most nominal conditions, nearly all trajectories	Trajectories	Actuator	Translation		
will need revision as the flight progresses. A continual trade-off exists between efficiency, capacity,					CA (policy on tradeoffs); PM (model of eff, cap, flex)
flexibility, and rate of update of trajectories that also depends on weather, traffic density, and the type					PM (Weather, density, op type)
of operation. 4DT management and improved weather prediction will support some advance de-	Weather prediction	Sensor		Provides continuous real-time and predicted weather info; 'intelligent' sensor	Sensor (in this case, although weather prediction obviously has its own alg)
confliction of traffic flows, thus reducing the need for tactical interventions during flight. How this will					CA ([or Cntl Input] reduce tactical interventions)

Continued on next page

Table 30 Analysis for Negotiation with ANSP

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
finally be implemented will depend on research, as well as on mid-term implementation, but some					
general possibilities are provided.					
It is likely that most problems, whether flow management involving many aircraft or separation	Aircraft & Airspace	Controlled Process	Interacts	Separation and flow	PM
management involving two aircraft, will be solved as early as practical, as soon as the information					Process (2 or more conflicting aircraft, flow of a/c)
about the problem is good enough to support deriving an acceptable solution, with low probability that					PM and CA (PM: 'goodness' of data; CA to determine what to do with relative 'goodnesses')
the "problem" was a false alert, or that the "solution" fails to solve the problem. This is to give the best					
chance of minimizing deviations from the operator's desired trajectory and to promote overall					CA (objectives: stability, predictability, early negotiation)
trajectory stability and predictability. Thus, the ANSP will initiate negotiation earlier rather than later.	ANSP	Controller	Generates actions		

Continued on next page

Table 30 Analysis for Negotiation with ANSP

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
The ANSP will generally try to make minor trajectory changes through timing or speed adjustments	ANSP	Controller	Generates actions		Cntl Algorithm (Policy for preferring minor traj changes); Action (Speed adjustments)
where possible for separation, initiating airborne merging and spacing or other aircraft procedures, and					CA (merging / sequencing algorithm); PM (separation, spacing requirements, merging)
flow management. Path modification would be used when timing adjustments are insufficient or not					CA (decision about insufficiency of spd/tm adjustment); Action (path modification)
the best solution. The overriding objective will be to maintain trajectory stability (and thus the ANSP					Cntl Algorithm (cntl objectives, policy for maintaining traj stability)
prediction functions) and minimize the need for tactical vectoring, but how this is achieved will depend	ANSP	Controller	Generates actions		CA (objectives: Policy for minimizing tactical maneuvers)
on the conditions and type of operation. Minor trajectory updates will replace most of today's open-					CA (decision on 'minor' traj mods versus open vector)
loop vectors for spacing or separation. Vectors are not really a negotiation in that the flight crew will	Flight Crew	Controller	PM		Control Action (Vector); Safety resp (fc must follow clearance); Control Action (Execute heading change, [FC])

Continued on next page

Table 30 Analysis for Negotiation with ANSP

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
accept the clearance unless they have an overriding safety reason to not do so. Vectors take the aircraft					PM (Open v closed traj)
onto an open trajectory that subsequently must be closed with a new 4DT.					
P.14-15					
The pilot must also work to close the trajectory. Pilots will need to update waypoints leading to a	Pilot	Controller	Actions		Control Action (update FMS/waypoints)
closed trajectory in the FMS, and work to follow the timing constraints by flying speed controls.					PM (waypoints/times of a closed Trajectory); Actuator (FMS); Cntl Action (speed controls)
Under dense traffic conditions, buffers will be needed in the system so that local trajectory changes can					PM of ANSP (traffic density); CA of ANSP (4DT or other for ensure buffer ->policy a function of density, desired stability/propagation)

Continued on next page

Table 30 Analysis for Negotiation with ANSP

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
be made without propagating beyond the aircraft initially affected. ANSP automation will predict	ANSP Auto	Sensor		lower level cntlr, possible a "sensor" - algorithm to predict aircraft traj given conds and capabilities.	PM (constraints, ac type, weather, ac traj)
conditions when it will become infeasible for aircraft to meet future constraints, and will propose	Human ATC	Controller	form PM	see next line	PM (Aircraft capability, predicted future trajectory [via automation], alt trajs [ibid])
trajectory updates to the controller, possibly before the aircraft start requesting them, both to optimize	Human ATC	Controller	form PM		CA (policy of preempting aircraft traj requests)
overall system flows and to reduce communications. For instance, if winds are more strongly out of the					CA (objectives: Policy for reducing communication)
west than forecast at Dallas-Ft. Worth (DFW), all the flights from the west coast will be arriving early,					PM (weather, trajectory prediction)
and those from the east coast will be arriving late. This will trigger revised times at the arrival fixes.					
Users may include predefined preferences (similar to the FMS cost index function) with the filed flight	Users	Sensor		(actually external info)	Input / Feeback (?) (Cost index; predefined preferences; flight plan)

Continued on next page

Table 30 Analysis for Negotiation with ANSP

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
plan for ANSP automation to consider in proposing alternative trajectories.					Controller Ouput (not necessarily a cntl action: alt trajectories)
The ANSP may negotiate by proposing revised constraints to the aircraft or the FOC, and let the	ANSP	Controller	Generates actions; transmits info		PM (necessity of revised constraints [further questions]); CA (policy for determining whether to send rev constr, alt trajs, or new [single] traj)
operator choose a preferred trajectory to meet the constraints. This new preferred trajectory could then	Operator	Controller	Generates actions	This is a lower level controller than ANSP, but above FC	PM (preferred traj. Does 'pref traj' = '4DT' [i.e. 'actual traj'])
become the authorized trajectory that now has to be renegotiated if a further change is required. This is					
likely to be the case when the ANSP is using the trajectory for separation management.	Trajectories	Actuator			Action (Traj for separation mgmt)
Performance requirements for airspace and operations will generally be set before flight, but	Performance requirement	Actuator	Translation		PM (performance reqs; time of application of perf reqs)
occasionally might be used opportunistically during flight. This might occur, for instance, during					

Continued on next page

Table 30 Analysis for Negotiation with ANSP

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
en route convective weather, where the ANSP negotiates optimal routes for high-performing aircraft					PM (convective weather, ac performance)
(data link, low RNP) through gaps in the weather, and lower performing aircraft are routed around it.					PM (RNP, communication type, weather/gaps); CA (algorithm for determining optimal vs route around)
In this example, the better-equipped aircraft gets the advantage of using the more direct route.					CA (policy for routes / equippage)
					End of Table 30

Table 31: Analysis for Negotiation with FOC

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
The dispatcher is typically a person responsible for managing flights; one or more dispatchers perform	Dispatcher	Controller	Generates actions		Controller (Dispatcher)
their work in a FOC. For single-aircraft operations, the flight planning function may be performed by	FOC	Controlled Process			Controller (Collection of Dispatchers); Action (Flight Planning); Actuator (Flight Plan)
the pilot in command, member of the flight crew, or another entity on their behalf such as a flight					Controller (PIC, FC, Flight Service Station, 3rd Party Co); PM (responsible flight planner)
service station, or to a third-party company who offers collaborative air traffic management (CATM)					
service to Part 91 operators.					
The flight planner's horizon for negotiating a flight may vary from tens of minutes to weeks. Typical	FOC	Controller			PM (time horizon); Alg (when to make decision)
reasons for negotiating a trajectory with the ANSP include the following:					"Reasons" - control algorithm and process model
- For an airline, build and update a schedule for its overall flight operation to meet the schedule					Process (Schedule); Action (Build/update overall flight operation)
- Get best initial trajectory for individual aircraft before takeoff					CA (Trajectory ['best' - CI]); Process (Individual Aircraft)

Continued on next page

Table 31 Analysis for Negotiation with FOC

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
- Prioritize among multiple flights (under the flight planner's jurisdiction) entering congested					CA ([FOC] Prioritization policy); PM (multiple flights; airspace congestion)
airspace or terminal area					
- Re-route, delay or substitute one or multiple flights around weather or congestion to maintain	Trajectory	Actuator			CA ([FOC] Prioritization policy, re-route/delay/substitute policy); PM (multiple flights; airspace congestion; weather)
business objectives					
- Diversion from original planned destination due to severe conditions, weather, fuel needs,					CA ([FOC] Prioritization policy, re-route/delay/substitute policy; diversion); PM (multiple flights; airspace congestion; weather); Proc Disturbance (severe condition, weather, fuel need, aircraft emergency, passenger consideration)
aircraft emergency, passenger considerations, etc.					

Continued on next page

Table 31 Analysis for Negotiation with FOC

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
The FOC will maintain a model of the aircraft performance and other key characteristics to anticipate	FOC				PM (a/c performance, key characteristics, expected flight progress, environment, fuel usage)
the expected flight progress in the given environment, fuel usage, etc. This information informs the					
flight planner of what trajectory options are feasible for the aircraft when negotiating a new trajectory					PM (feasible trajectories); Alg (negotiation); Cntl Input (from ANSP)
with the ANSP. New processes and protocols by which revised trajectories are negotiated and					PM/Act/Sens/Alg (processes and protocols);
approved between the FOC, ANSP, and aircraft may be needed. This may include some form of partly					Cntl Input (ANSP); Process (a/c)
automated negotiation to replace the daily and hourly teleconferences between the ANSP and various	Partly automated negotiation (ANSP, FOC)	Controller (or sensor)			
FOCs to strategize about weather and other events.					PM (weather, other events); Alg ('strategize')

Continued on next page

Table 31 Analysis for Negotiation with FOC

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
Preflight planning and flight following are key roles of the FOC in order to develop and maintain the	FOC	Controlled Process		"flight following" is the controlled process; preflight planning is a sensor	Process (FOC following flights)
business plan and business trajectory of the operator through optimization of both the individual					Alg (optimization); PM (individual a/c; fleet); Action (Business Plan, Business Trajectory)
aircraft and the fleet. This includes specification of the airframe to be used to conduct the operation,					Alg (spec of ac, fuel, FC); PM (capability of a/c, FC; fuel(?))
fuel decisions, and flight crew assignments. Once payload and fuel decisions have been made and the					Alg (spec of ac, fuel, FC, payload); PM (capability of a/c, FC; fuel, payload)
fuel for the flight has been loaded, flexibility is very limited. This is especially true for very long haul					PM (time of fuel loading; route [long v short-haul]; aircraft weight); Alg (haul length & fuel, loading)
flights limited by weight. Typically, these decisions are made anywhere from a few hours before the					CA (determining time [rel to departure] of decision)
flight up to the time of departure, depending on the latest payload, weather, and other related					PM (time of dept, payload, weather forecast)

Continued on next page

Table 31 Analysis for Negotiation with FOC

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
information. As with the ANSP, the objective is to make most strategic decisions before the flight					Alg (decision time [pre- v post-departure])
commences. Even during the flight, the dispatcher is the primary and preferred decision maker for					PM (negotiator [FOC, FC]); Alg (decision on who to negotiate with)
strategic negotiations with the ANSP because the FOC has access to more information, and the					Assumes FOC has access to more information; this assumption could become invalid for a wide variety of circumstances
negotiation can take place over net-centric operations. The cockpit is also part of net-centric operations	Cockpit (Crew or Automation)	Controlled Process AND Controller			
and works with their dispatcher in concurrence on changes. The FOC will generally be negotiating	Net-centric ops	Sensor, actuator			Acts as path for sensing, actuating flight path via 4DT
trajectories greater than 20 to 30 minutes into the future with the ANSP, and their role in negotiation	ANSP	Controller			PM (time to "negotiation even"); Alg (decision on who to negotiate with; 20-30 minute criterion)
diminishes relative to the flight deck as the time gets closer to the event for which the negotiation was					Assumes 20-30 minute timeframe constitutes "strategic" negotiations

Continued on next page

Table 31 Analysis for Negotiation with FOC

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
initiated. Once a revised trajectory is negotiated, this new trajectory is conveyed to the pilot for	Pilot	Controlled Process	Translate higher level action into lower level one		PM (revised trajectory [for all controllers - FOC, FC, ANSP])
approval and execution. The expanded role for the FOC enabled by the shift to more strategic decision-					CA (execution); Cntl Status (approval)
making will need to be refined.					
The ANSP provides a forum to facilitate collaboration between flight planners representing multiple	Flight planner	Controller		Flight planner is then in 'control' of the multiple FOCs	
FOCs when there is a system-wide event or constraint. Among multiple operators, aggregate solutions	System-wide event	Controlled Process		Event is actually a disturbance to CP	Process Disturbance (system-wide event or constraint)
to demand/capacity imbalances may be proposed to and by the ANSP. They should provide improved					PM (Demand, Capacity); CA (comparing imbalance -); Action ("aggregate solutions")
operations within the context of the operator's business objectives in comparison to solutions that					PM (Operator business objective); CA (policy function of biz obj, demand, capacity)

Continued on next page

Table 31 Analysis for Negotiation with FOC

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
might be individually imposed by the ANSP. Common situational awareness across the flight planning	Common situational awareness	Sensor	Continuous, discrete data	Assume net-centric operations based on previous data	
participants improves the options for dealing with constraints.					
					End of Table 31

Table 32: Analysis for Negotiation with Pilots

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
During the flight, the flight crew complies with the cleared trajectory except in emergencies; non-	Flight crew	Controlled Process	Translate	In Emergencies it becomes 'top level' controller	Process (FC - compliance w/ traj)
emergency changes are negotiated and agreed to before being executed. Flight crews will have access					PM (emergency status [y/n or continuum])
to 4D weather and NAS status information relevant to their flight through network-centric operations.	Network-centric Ops	Sensor	Provide binary or cont info	Sensor is actually something else, but net-centric provides the "lines" between blocks	PM (F/C PM: 4D weather; NAS status; both "relevant to flight")
Some aircraft may have sophisticated flight-planning functionality onboard, including trial planning	Flight Planning Avionics	Sensor	Provide binary or discrete info	This is an intelligent sensor; could provide binary 'Y/N' for trajectories as well as discrete parts of flight plan(s)	Sensor (Flt Pln Avionics - trail plan, eval of trajs); PM (F/C PM: 'eval of trajs')
and evaluation of proposed trajectories. However, even with advanced airborne decision-support	Advanced Airborne Decision-support Automation	Sensor	Provide binary or discrete info	See above comment	

Continued on next page

Table 32 Analysis for Negotiation with Pilots

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
automation, pilot-initiated trajectory negotiations may be limited by workload considerations.	Flight crew	Controlled Process		Part of the FC output is also in terms of feedback, in this case negotiation or emergency	PM (grnd: neg from F/C)
The pilot monitors progress toward meeting assigned constraints and initiates negotiation directly or	Flight crew	Controller	Form, update process model		PM (Assigned constraints, progress to constraints; negotiating entity (FC/FOC))
through the FOC if projected to be unable to meet a constraint. If the aircraft can still meet the					PM (Able/unable to meet constraints)
constraints of the 4DT, but would prefer to renegotiate the constraint for efficiency or scheduling					CA (efficiency or scheduling); Actuator (Constraint)
reasons, the pilot or FOC may request negotiation of the constraint, or the preferred constraint changes					
may be listed as alternatives in the flight plan. When a pilot requests a minor trajectory change that still	Flight crew	Controller	Generate action	Request is only a precurser to actually generating action on aircraft	CA (policy for determining type of change); PM (deltaTraj meets/doesn't meet constraints; status of request); FB to higher cntl (change request)

Continued on next page

Table 32 Analysis for Negotiation with Pilots

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
meets all constraints, this trajectory change should be a fairly easy process expedited by local ANSP	Local ANSP Automation			Unclear if this automation has decision authority; thus unclear what it's role is. "Expedited" implies that the automation may simply approve it. This statement also implies that constraints are not the only thing that defines a trajectory	Actuator (Trajectory; Constraints)
automation. Negotiating a change in constraints can be much more complex, since constraints have	Constraint negotiation	Actuator			
typically been negotiated and agreed upon through collaborative air traffic management (CATM)	CATM Procedure	Controller	Generate action	This is in the general ANSP controller function	CA (negotiation, agreement criteria)
procedures.					

Continued on next page

Table 32 Analysis for Negotiation with Pilots

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
For most airspace and operations, trajectories will include some flexibility (sometimes referred to as	Trajectories	Actuator		Trajectories are the constraints that ultimately define aircraft behavior, if followed	Actuator (Trajectory; Constraints; Windows); PM ([both grnd and FC]: traj, consts; windows)
"windows") that allows the operator to optimize within limits without renegotiating the trajectory.	Operator	Controller	Generate action		CA (optimization alg); PM (cost index/function; limits of window)
This might apply to an aircraft choosing a path in real time between thunderstorms. Conformance					
monitoring on the ground considers the range of these windows, so minor maneuvering is authorized	Conformance Monitoring	Sensor	Provide binary and cont info		Sensor (Aircraft state; Conformance notification); PM ([grnd] Window; Conformance; 4D Contract); Actuator (4D Contract)
within the context of the 4D contract.					
In 2025, aircraft will vary widely in their ability to accurately adhere to a 4DT and in their ability to					
exchange trajectory information with the ground. At the lower end of performance, some aircraft are	Aircraft	Controlled Process			Process (Capability, Information Exchange)

Continued on next page

Table 32 Analysis for Negotiation with Pilots

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
only capable of adhering to wide lateral trajectories and flexible timing requirements (windows), while	Aircraft	Controlled Process			Actuator (Trajectory; Constraints; Windows); PM ([both grnd and FC]: traj, consts; windows; aircraft capability); Process (aircraft)
high-performing aircraft can transmit via data link detailed 4D trajectories that can be executed with	High-performing aircraft			Datalink is the line between boxes of control model, not an actual sensor/actuator	Sensor (Datalink air-to-grnd)
high accuracy. In general, NAS operations and ANSP decision-support automation must be designed	NAS Operations				
to deal with the entire spectrum of aircraft capabilities. However, significant operational efficiencies	ANSP decision-support automation				PM (Aircraft capability); CA (Efficiencies; policy w/r/t aircraft capability)
can be gained by taking advantage of the additional information-sharing capabilities and performance	High-performing aircraft	Controlled Process	Interact with env		Process (Navigation performance; Info-sharing capabilities)
accuracies of high-performing aircraft.					

Continued on next page

Table 32 Analysis for Negotiation with Pilots

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
The far-term JPDO Operational Improvements include a self-separation capability, which may be	Self-separation	Controller		This implies the pilot/ac as its own control agent	
restricted to airspace where only self-separating aircraft can operate or may include mixed-equipage					
environments where some aircraft are self-separating while others are ANSP-managed . There are also					
a number of pair-wise delegated separation operations in mid- and far-term NextGen, such as airborne	Delegated-separation	Controller			
merging and spacing, parallel runway operations, and oceanic procedures. Self-separating aircraft	Merging and spacing, parallel runway ops, oceanic procedures	Controlled Process		Controlled Process (Operation type)	
separate themselves using ADS-B as the primary means of airborne surveillance. ADS-B provides the	ADS-B	Sensor	Transmit continuous and discrete data	Discrete data is aircraft plan/trajectory; continuous is aircraft state	Sensor (ADS-B [for self-sep aircraft cntl loop])

Continued on next page

Table 32 Analysis for Negotiation with Pilots

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
location and velocity vector, plus the near-term intent of other self-separating aircraft in the vicinity.	ADS-B	Sensor	Transmit continuous and discrete data	Discrete data is aircraft plan/trajectory; continuous is aircraft state	PM (Location, velocity, near-term intent, self-separating aircraft, 'vicinity')
Research is required to determine how to safely and effectively utilize the capabilities of these					
advanced aircraft in increasing efficiency and throughput of NextGen operations. When an aircraft has					
been delegated some separation responsibility; there needs to be sufficient flexibility in the trajectory					Actuator (Trajectory with flexibility)
that maneuvers for separation can be accomplished without trajectory negotiation. In this case, the					Action ([FC] Manuever inside of [ANSP-sanctioned] trajectory)
trajectory might be constrained only sufficiently to support traffic flow management and would be					PM ([ANSP] constraints); CA (traffic flow mgmt - constraints sufficiency/flexibility tradeoff)
virtually useless for separation management decisions. However, these self-separating aircraft are					CA (decision wrt usefulness of constraints for flow vs separation mgmt)
themselves operating based on highly accurate trajectories, which are continuously maintained to be					PM (accuracy of trajectory)

Continued on next page

Table 32 Analysis for Negotiation with Pilots

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
conflict free using onboard strategic and tactical conflict detection and resolution functionality.	Onboard avionics			Role allocation is subject to discussion (CD&R can be modeled as a controller or a sensor for pilots)	
To ensure predictable operations, trajectory changes that result in the near-term generation of new	Trajectory change	Actuator			CA (control objective: predictability, no near-term conflicts), PM ('near-term'), Action (trajectory change)
conflicts are not acceptable. To sustain situational awareness, the new trajectory path is broadcast via					PM (new trajectory path)
ADS-B before the aircraft begins maneuvering. This is especially important outside of the range of any					
ANSP surveillance. If this highly accurate trajectory were made available to the ground via data link,	ANSP Surveillance	Sensor	Continuous data	This implies that ANSP does NOT use ADS-B as a primary data source	Sensor (ANSP surveillance)

Continued on next page

Table 32 Analysis for Negotiation with Pilots

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
then the ANSP decision-support automation could accurately predict the movement of both self-	ANSP decision-support automation				PM (prediction of movement; self-, delegated- and ANSP-separated aircraft)
separating and delegated-separation aircraft. If future ANSP separation management decision-support					
automation were designed to effectively use this more accurate information for conflict detection and					
resolution functions, then more effective use of airspace capacity should result. Thus, an open research					
question is whether there should, in fact, be two trajectories in effect for these aircraft: a negotiated	Trajectories	Actuator			PM ([ANSP] Negotiated vs "4D Intent" Trajectory); CA ([ANSP] deciding on when/if to issue different version of traj)
trajectory used for traffic flow management that provides sufficient flexibility for self-separation					
maneuvers, and a highly accurate "4DT intent" used for separation management that is periodically					

Continued on next page

Table 32 Analysis for Negotiation with Pilots

Original Text [JPDO, 2011]	Source	Role	Behavior	Context	Control Model
updated by the aircraft without negotiation. NAS operational efficiency would probably benefit from					
having access to 4DT intent information from all aircraft that are capable of generating, transmitting					Action (performing 'highly accurate' 4DT); CA (generating 'highly accurate' 4DT); PM ([ANSP] All a/c - 'highly accurate' 4DT)
via data link, and performing to a highly accurate 4DT.					Sensor ([a/c - ANSP] datalink and 'highly accurate' 4DT)
					End of Table 32

[Page intentionally left blank]

Appendix C: Formalism of Hazardous Control Actions

The following definition of a hazardous (unsafe) control action comes from the work of Thomas [2013]. A hazardous control action in the STAMP accident model can be expressed formally as a 4-tuple $\left(\mathcal{C}^i, T, \mathscr{C}, Co\right)$ where:

- $\mathcal{C}^i$ is the source controller that can issue control actions in the system. The controller may be automated or human.
- T is the type of control action. There are two possible types: Provided describes a control action that is issued by the controller while Not Provided describes a control action that is not issued.
- $\mathscr{C}$ is the control action (i.e. command) that is output by the controller.
- Co is the context in which the control action is or is not provided.

Each element of an unsafe control action is a member of a larger set. The following properties must hold:

1. $\mathcal{C}^i \in S$, where S is the set of source controllers in the system ($\mathcal{C}^i$ is then the source of action in each level of the system, per equations (19) and (22))
2. $T \in \mathcal{T}$, where $\mathcal{T}$ ={Provided, Not Provided}
3. $\mathscr{C} \in \mathcal{O}_a\left(\mathcal{C}^i\right)$ where $\mathcal{O}_a\left(\mathcal{C}^i\right)$ is the set of control actions that can be provided by controller $\mathcal{C}^i$
4. $Co \in \rho\left(\mathcal{C}^i\right)$, where $\rho\left(\mathcal{C}^i\right)$ is the set of potential contexts, or process model states, for controller $\mathcal{C}^i$

To assist in enumerating or aggregating individual contexts, the context Co is further decomposed into variables, values, and conditions:

- V is a variable or attribute in the system or environment that may take on two or more values. For example, heading, position, and airspeed are three potential variables for a aircraft.
- VL is a value that can be assumed by a variable. For example, M0.85 is a value that can be assumed by the variable airspeed.
- Cd is a condition expressed as a single variable/value pair. For example, aircraft conformance is a condition.

- The context Co is the combination of one or more conditions and defines a unique state of the system or environment in which a control action may be given.

The following additional properties related to the context of a hazardous control action can therefore be defined:

5. $V \in \mathcal{V}\left(\mathcal{C}^i\right)$, where $\mathcal{V}\left(\mathcal{C}^i\right)$ is the set of variables relevant to the system hazards $\mathcal{H}$
6. $VL \in \mathcal{VL}(V)$, where $\mathcal{VL}(V)$ is the set of values that can be assumed by variable V
7. $Cd = (V, VL) \in \mathcal{CD}\left(\mathcal{C}^i\right)$, where $\mathcal{CD}\left(\mathcal{C}^i\right)$ is the possible set of conditions for controller $\mathcal{C}^i$
8. $Co = (Co_1, Co_2, \ldots)$, where each Co_i is independent. That is, no two Co_i refer to the same variable V.

Finally, each hazardous control action must be linked to a system-level hazard:

9. To qualify as a hazardous control action, the action $\left(\mathcal{C}^i, T, \mathscr{C}, Co\right)$ must be able to cause a hazard $H \in \mathcal{H}$, where $\mathcal{H}$ is the set of system level hazards.

A hazardous control action expressed as a 4-tuple $\left(\mathcal{C}^i, T, \mathscr{C}, Co\right)$ must satisfy the above properties 1-9.