Model-based Concept Development and Safety Driven Design

Cody H. Fleming

Department of Aeronautics & Astronautics
Massachusetts Institute of Technology
77 Massachusetts Avenue, 33-407C
Cambridge, MA 02139 USA
chf44@mit.edu

Abstract: This paper outlines a new approach for safety-driven concept development based on the systems-theoretic accident model and process. Compared to traditional approaches, this model captures more types of accident causes prevalent in modern systems. In addition, the model emphasizes functional behavior in addition to physical behavior, which makes it a promising candidate for use during early system development stages. The new approach represents an extension of the accident causality model that allows stakeholders to systematically develop a model of a concept and then query the model to identify potential vulnerabilities and mitigation strategies.

1 Introduction

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 the late stages of system development, when detailed design information is available. However, system safety cannot be assured by discovering problems at these late stages and adding patches or expensive updates to the design. Rather, safety must 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 research proposes 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 the accident causality model, called systems-theoretic accident model and process (STAMP), extending the process so that it can be applied systematically 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 vulnerabilities and undocumented assumptions, 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 using control theory to generate a model of the concept of operations. Formal, systems-theoretic concepts such as hierarchy, emergence, communication, and coordination are used to analyze the model and identify vulnerabilities in the concept.

This model-based approach represents a significant departure from the state of the art, where a concept is defined, developed, documented, and analyzed according to a control theoretic model rather than free form, natural language text. In this thesis, the power of the proposed approach will be demonstrated on a real concept currently being developed by the United States Federal Aviation Administration, in conjunction with the Single European Sky (SESAR) program.

2 Section heading

This research is different from—represents a contribution to—existing STAMP/STPA approaches and the general systems engineering literature for several reasons. First, systematic and rigorous methods are typically not used in the development and analysis at the conceptual stage. Secondly, model-based systems engineering approaches do not (explicity) consider hazardous behavior or help stakeholders to identify potential safety-related vulnerabilities.

The focus of this research 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 for example [INC11, Kap10, IEE98]), 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 typically 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 [JPD11]. All of these characteristics make it difficult to systematically analyze a Concept of Operations and in particular to rigorously, systematically:

1. identify *missing information* or *undocumented assumptions* that will be required for effective operation of the system;
2. identify *inconsistent* or *conflicting* information within a concept that may lead to hazardous behavior;
3. identify where *more specific* operational concepts are required to understand safety- and functionally-related behavior of the system;

Model-Based System Engineering methods and tools have been proposed in order to ensure consistency in requirements and testing efforts, develop specifications, define inter-

face requirements, develop test plans, and intellectually manage complexity. Elements of an effective model-based system engineering model include clear and unambiguous language, behavior expressed in relationships that represent the "structure" of the system, a system representation built from the language and structure, and the ability to express the representation from different views [LZ11].

Model-Based System Engineering methods represent the functional relationships among system components – the "structure" – with a generic set of guide words. For example, in Systems Modeling Language (SysML), structural relationships include Sequence, Select, Multiple-Exit, Iterate, Loop, Concurrency, or Replicate [Del13]. There are many other languages (e.g. Object-Process Methodology [Dor02, Sod02]) and representations (e.g. Design Structure Matrix [EB12]) used to represent entities, their attributes, and the relationships between them [LZ11].

Rather than directly using the control flaws and causal factors from STPA Step 2 (see [Lev12], page 223), this approach first examines the most *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? For the purposes of meeting document submission requirements, consider just two of the four basic elements in a control loop.

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

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; has algorithm built-in but no control authority)
- Synthesizes and integrates measurement data

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 later work.

The roles of the controller, actuator, controlled process, and sensor, and their interactions with the environment and other control loops can be summarized with 13 generic keywords

or tagwords. Figure 1 depicts these tagwords in the familiar control loop format. With a proper accounting of these 13 items, the control loop can achieve the four necessary conditions[1] [Ash57] of process control and adequately interact with its environment, other processes, and other controllers. In other words, these keywords are necessary to ensure that a control loop is controllable and coordinable with other controlled processes.

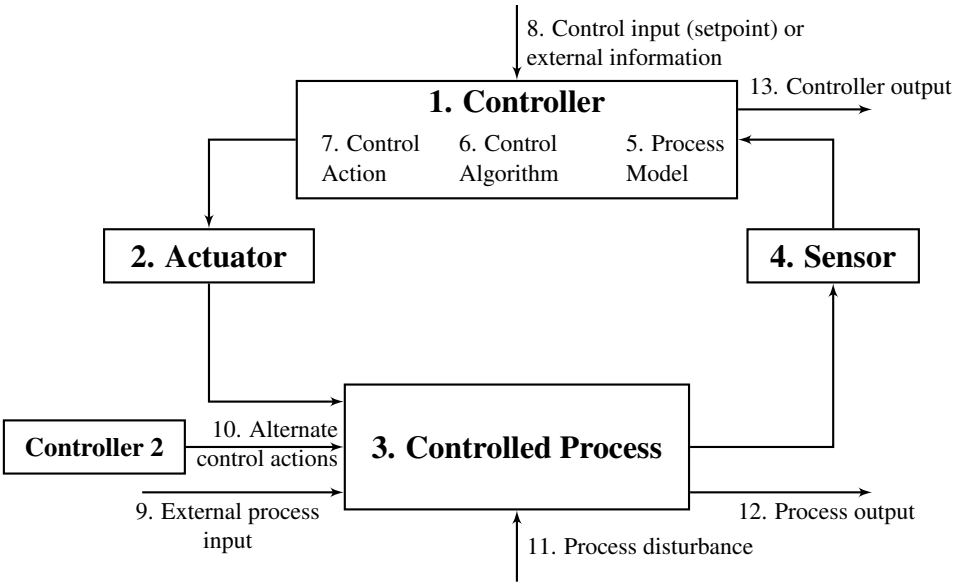

Figure 1: Control Loop with generic STAMP entities

The information in Figure 1 and the above lists (Controller, Sensor but also Actuator and Controlled Process) 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 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.

3 Conclusions

The workshop presentation will further show how these concept models can be interrogated systematically in order to identify vulnerabilities and suggest potential architectural alternatives. The presentation will use a real-world application in air traffic management being developed jointly in the United States and Europe called Trajectory-Based Operations (TBO).

[1] (1) Goal, (2) Action, (3) Model, and (4) Observability conditions are required for effective process control.

References

[Ash57] W Ross Ashby. *An Introduction to Cybernetics*. Chapman & Hall Ltd., 1957.

[Del13] Lenny Delligatti. *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Pearson Education, 2013.

[Dor02] Dov Dori. *Object Process Methodology: A Holistic Systems Paradigm; with CD-ROM*. Springer, 2002.

[EB12] Steven D Eppinger and Tyson R Browning. *Design structure matrix methods and applications*. The MIT Press, 2012.

[IEE98] IEEE. IEEE Std 1362—Guide for Information Technology—System Definition—Concept of Operations Document. Technical report, Institute of Electrical and Electronics Engineers, Inc., 3 Park Avenue, New York, New York 10016-5997 USA, 1998.

[INC11] SE INCOSE. INCOSE Systems Engineering Handbook v. 3.2. 2. Technical report, INCOSE-TP-2003-002-03.2. 2. October, 2011.

[JPD11] JPDO. JPDO Trajectory-Based Operations (TBO) Study Team Report. Technical report, Joint Planning and Development Office, 2011.

[Kap10] Stephen J Kapurch. *NASA Systems Engineering Handbook*. DIANE Publishing, 2010.

[Lev12] Nancy G. Leveson. *Engineering a Safer World*. MIT Press, 2012.

[LZ11] David Long and Scott Zane. Language: The Systems Model Is Language-Based. In A Primer For Model-Based Systems Engineering (2nd ed.), 37. Technical report, Vitech Corporation, Blacksburg, VA, 2011.

[Sod02] Nathan R Soderborg. *Representing systems through object-process methodology and axiomatic design*. PhD thesis, Massachusetts Institute of Technology, 2002.