

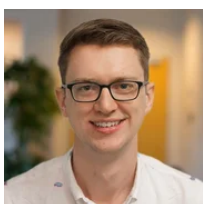
Search the blog

# Software Engineering Institute

## SEI Blog

[Home](#) > [Publications](#) > [Blog](#) > Simultaneous Analysis of Safety and Securi...

## Simultaneous Analysis of Safety and Security of a Critical System



**SAM PROCTER**

SEPTEMBER 11, 2017

### CITE

[Get Citation](#) 

As computers become more powerful and ubiquitous, software and software-based systems are increasingly relied on for business, governmental, and even personal tasks. While many of these devices and apps simply increase the convenience of our lives, some--known as *critical systems*--perform business- or life-preserving functionality. As they become more prevalent, securing critical systems from accidental and malicious threats has become both more important and more difficult. In addition to classic safety problems, such as ensuring hardware reliability, protection from natural phenomena, etc., modern critical systems are so interconnected

that security threats from malicious adversaries must also be considered. This blog post is adapted from a [new paper](#) two colleagues ([Eugene Vasserman](#) and [John Hatcliff](#), both at Kansas State University) and I wrote that proposes a theoretical basis for simultaneously analyzing both the safety and security of a critical system.

### Why is this an issue now?

One common way of determining the safety of a critical system is to perform what's called a *hazard analysis*. There are a number of traditional hazard-analysis techniques; two of the most popular are [failure modes and effects analysis \(FMEA\)](#) and [fault tree analysis \(FTA\)](#). The development of these traditional techniques predates modern levels of interconnectivity, so most versions of the techniques do not explicitly address security concerns. This omission has been a problem in a number of domains ranging from industrial control systems, attacked by [Stuxnet](#); to the smart power grid, which faces [a number of challenges](#); to even personal medical devices that [face attacks](#) now that they expose functionality over wireless radios and the Internet.

### What's being done about it?

This rise in security threats has led some researchers to adapt traditional hazard-analysis techniques to include or focus on security-related issues. Christoph Schmittner and his colleagues showed how [FMEA can be used for security analysis](#) and Philip Brooke and Richard Paige have demonstrated the use of [FTA for secure system design](#). But the field of hazard analysis research is moving in other directions as well. In addition to the inclusion or exclusion of security concerns, a second dimension of hazard analysis is the incorporation of *systems theory*.

[Nancy Leveson](#) from the Massachusetts Institute of Technology is perhaps [the biggest proponent for the use of systems theory, which advocates a more holistic approach](#). Systems theory--as opposed to the *analytic-reduction* style of analysis used in the traditional scientific method--has been integrated into a new causality model and hazard analysis technique:

- **Systems-Theoretic Accident Model and Processes (STAMP)** - A new causality model (initially [presented here](#)) that incorporates systems theory and shifts the focus from preventing safety problems to enforcing safety constraints.
- **Systems Theoretic Process Analysis (STPA)** - A new hazard analysis technique based on STAMP. Note that while STPA did not address directly security concerns in its [initial version](#), Leveson's student William Young has begun work on a security derivative known as [STPA-Sec](#).

Others are working in this area as well: Friedberg et al. developed their own security-and-safety derivative of Leveson's technique called STPA-SafeSec. And, as part of my [Ph.D. research](#) before I joined the SEI, I created a refinement of STPA that's focused on hardware- and software-based subsystems called the [Systematic Analysis of Faults and Errors](#) (SAFE).

*Table 1: Safety and Security Analysis Techniques*

	Safety-Focused Techniques	Security-Aware / Focused Techniques
Traditional Causality	FMEA, FTA	FMEA (Sec), FTA (Sec)
System Theoretic	STPA	STPA-Sec, STPA-SafeSec, SAFE

## What's New in our Approach?

For this paper, our approach differed from previous efforts in that we were not attempting to prescribe an exact series of steps for analysts to follow when analyzing their systems. Rather, we examined the basis of one of the key elements shared by most analysis techniques: their use of [guidewords](#). Guidewords are high-level concepts or terms that guide analysts to consider particular ways that a system can fail. These exist in both safety-analysis techniques--[STPA](#) has its Step 1 and Step 2 terms, [FMEA](#) has failure modes, and [HAZOP](#) is based around finding deviations from design intent using terms like *Late* and *More*--and security-analysis techniques ([STRIDE](#) is centered around the terms that make up its eponymous acronym). When most techniques were created, though, the guidewords were also created in an ad hoc manner, rather than being directly traceable to existing literature.

Our work showed how **SAFE**, which is a guideword-agnostic analysis technique, could be used with a set of terms derived from one of the classic adversary models used in security. That is, guidewords can be supplied to SAFE at runtime as parameters (a concept we refer to in the paper as *parametricity*), rather than being ad hoc and essentially inseparable. The **classic security model** we used is the one proposed in 1983 by Danny Dolev and Andrew C. Yao that describes the actions an adversary could potentially take if the analyzed system communicates over a non-trusted network. For many systems, even those that do not use the Internet, this is a reasonable assumption: keeping an entire network perfectly secure is often prohibitively hard. What's more, the attack types that arise from the Dolev-Yao model are so foundational that they can map cleanly (if informally, in this work) to concepts from both system safety and network security. Table 2 shows this mapping:

*Table 2: Mapping between system safety, Dolev-Yao, and network security concepts*

System Safety	Dolev-Yao	Network Security
None	Read	Violate Privacy
Corrupt Value	Modify Existing	Craft Arbitrary Packets
Late/Dropped Message	Delay/Drop	Increased Latency/Packet Loss
Early Message	Craft and Send	Impersonate, Deny Service

## Payoffs

The adversary described by Dolev and Yao's model controls a compromised component on a network. It can read any message, modify messages before they are received by their intended recipient, delay those messages (possibly indefinitely, effectively dropping the message), and craft/send custom messages to impersonate legitimate users of the network.

We believe that there are a number of benefits to a guideword-based, safety- and security-aware component-focused analysis like SAFE.

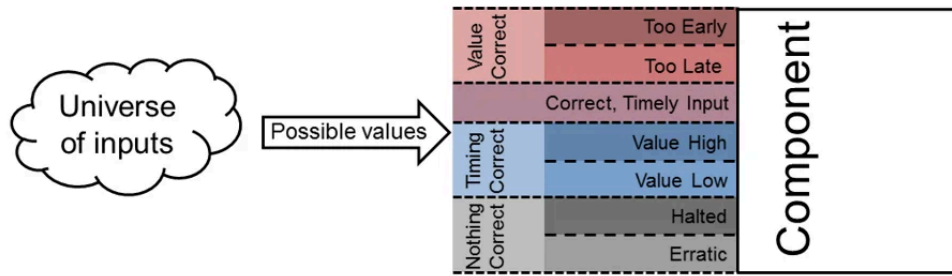


Figure 1: Compressing the input space

1. **Merging safety and security**--Perhaps most obviously, a safety and security co-analysis like STPA-SafeSec or SAFE benefits from simultaneously considering both aspects in a couple of key ways:
  1. **Reduced overhead**--Rather than perform separate analyses for safety and security, performing a single co-analysis means less rework. Many basic problems should be discovered by both analysis techniques, and that duplicated effort can mean less time is spent finding more subtle problems.
  2. **Fewer problems get missed**--As [Friedberg et al.](#) discuss, a system's safety and security properties often depend on and interact with one another; considering them separately can mean these interactions get missed.
1. **Analysis space reduction**--A SAFE analysis moves backwards through a system, starting with actuation points (i.e., the components a system uses to affect its environment), and then considers the sources of the inputs to those actuation points. As there are myriad ways inputs to a component can fail, considering them all can be prohibitively hard. Instead, we advocate and explain how *classifying* inputs--using an effects-focused guideword set like the one derived from Dolev and Yao's model--can reduce the number of errors to analyze, effectively *compressing* the analysis space. Figure 1 shows this graphically for a time- and value-sensitive input: either or both the input value and time can be wrong. Rather than focusing on the precise magnitude of the delay or value error, however, simply classifying the input as "too late" or "too high" produces a manageable number of failure cases.
2. **Independence of effects-based analyses**--Note that the input failure classifications from Figure 1 say nothing about the *cause* of the failure. That is, if an input message is delayed by a network failure, or active

adversary, or even a problem in a component's hardware, the result--delayed input--is the same. This means a component can, to some degree, be analyzed independently of the components that produce its input. In other words, regardless of what causes a failure, we can typically guarantee that its effects will manifest in one of the ways contained in our guideword set.

3. **System safety and formal methods**--There is an exciting connection between our work and **similar work** by John Rushby taking place in pure software systems using bounded **model checking**. At a high level, both techniques allow designers to derive the assumptions that their systems/components rely on for safe operation.
4. **Explicit incorporation of security model**--Explicitly incorporating an adversary model has two benefits:
  1. **Knowing what to expect**--Most obviously, it guides system designers to consider the behaviors that the adversary is allowed to do. That is, since the Dolev-Yao model says an adversary can snoop on messages sent on the network, the need for encryption immediately becomes clear.
  2. **Knowing what *not* to expect**--Less obvious, but just as important, is what an adversary model rules out. The Dolev-Yao model, for example, describes a network-based adversary with no direct access to components themselves. Making this explicit means it can be discharged as a formal environmental requirement (i.e., an assumption) such as "physical security for the component must be provided."

## Evaluation

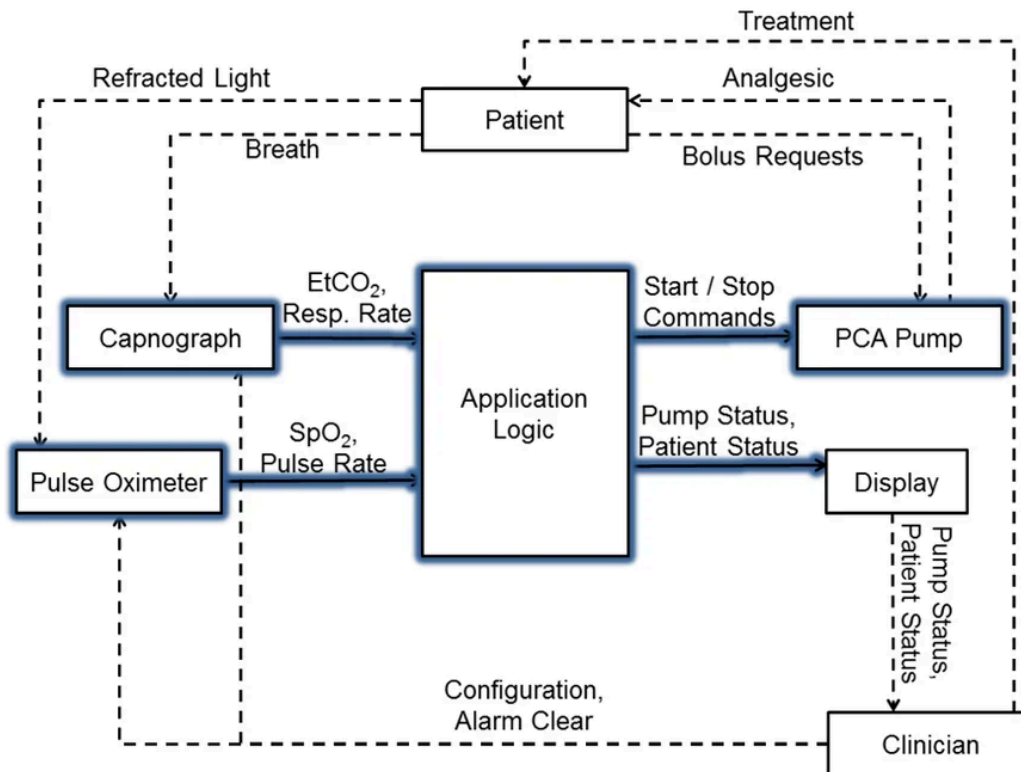


Figure 2: An integrated medical application

In the initial presentation of **SAFE**, the evaluation was based on an analysis of hazards in a system of interconnected medical devices and governing software. The **motivation and details** of the distributed medical application aren't germane to this blog post, but a high-level overview is provided in Figure 2. For this work, we adapted the previous analysis from my dissertation: we selected a single element of the system and repeatedly re-analyzed it using SAFE with different guideword sets derived from a range of sources. These included the following:

- The **Dolev-Yao** model
- A canonical taxonomy of dependable and secure computing developed by **Avizienis et al.**
- Terms from **STPA-SafeSec**'s first step
- **STPA/STPA-Sec**'s 'Step 1' guidance

Our evaluation was based on the likelihood that an analyst, following the process of SAFE, would detect hazards leading to various design improvements. These possible improvements include

- **Alarming**--The system should be able to issue an alarm to notify a clinician of a problem it cannot independently resolve.
- **Timeouts**--The critical part of the system should stop to enforce minimum message inter-arrival times to prevent message overload.
- **Timestamps**--Messages are time sensitive, so they should be timestamped to prevent late (or early) messages from being acted upon.
- **Hashed/Signed Messages**--Messages should be cryptographically signed and hashed to prevent forgery or modification.
- **Encrypted Messages**--Messages should be encrypted to prevent snooping on potentially sensitive message values.

Of course, none of these design improvements are particularly novel, but this exercise wasn't intended to come up with clever or unintuitive solutions to subtle problems in the medical system's design. Rather, we were interested in finding a set of guidewords that would consistently suggest the broadest set of improvements.

Guidewords aren't used in a vacuum, and hazard analysis isn't a computerized process. Our evaluation was thus necessarily somewhat subjective--see Table 3 for the full result. We rated whether an analysis would be *likely* to suggest an improvement (denoted with a "✓"), *might* suggest the improvement (denoted with a "?"), or would *likely not* suggest the improvement (denoted with a "✗"). Of course, a particularly skilled or experienced system designer/analyst might come up with the design improvements regardless of the guideword set used; the terms are used only to guide analysts to think about particular classes of errors.

### Table 3: Evaluation

Table 3: Evaluation

	Dolev-Yao	Avizienis et al.	STPA-SafeSec	STPA / STPA-Sec
Alarms	✓	✓	✓	✓
Timeouts	?	✓	?	✓
Timestamps	✓	✓	✓	✓
Negative Ticket Values	?	?	?	?
Hashed / Signed Messages	✓	✗	✓	?
Encryption	✓	✗	✗	✗



In the next few months, we'll explore how some of the foundational ideas from this work can integrate with ongoing projects here at the SEI. One promising direction is the integration of hazard/security analysis and semiformal architecture models such as those built in the SEI's popular *Architecture Analysis and Design Language (AADL)*. Not only does SAFE have an AADL-based implementation, but the SEI has the *Architecture-Led Safety Analysis (ALSA)* technique. *David Gluch*, a fellow SEI researcher, and I are looking at how this technique might be adapted to also address security concerns; we expect to produce a technical note here in a few months that describes what we've learned so far.

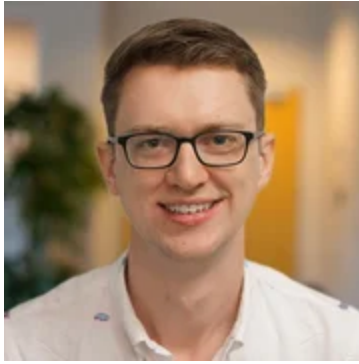
## Next Steps

I'm also particularly interested in automating analyses, so that domain experts can most efficiently leverage their personal expertise and not have to learn a lot of computer science/hazard-analysis theory. To that end, I think the links between this SAFE's style of backwards-chaining analysis and Rushby's *assumption synthesis* are particularly promising, and I want to continue exploring overlaps in that area as well.

## Additional Resources

*The paper on which this blog post is based*, authored by Sam Procter, Eugene Vasserman, and John Hatcliff, was presented at *SAW '17*, the 4th International Workshop on Software Assurance.

## WRITTEN BY



Sam Procter

[DIGITAL LIBRARY PUBLICATIONS](#) ▶

[SEND A MESSAGE](#) ▶

**MORE BY THE AUTHOR**

The OSATE Slicer: Fast Reachability Query Support for Architectural Models

NOVEMBER 13, 2023 • BY [SAM PROCTER](#)

---

A Model-Based Tool to Assist in the Design of Safety-Critical Systems

MARCH 7, 2022 • BY [SAM PROCTER](#)

---

Integrating Safety and Security Engineering for Mission-Critical Systems

MAY 10, 2021 • BY [SAM PROCTER](#), [SHOLOM G. COHEN](#)

---


The AADL Error Library: 4 Families of System Errors

MAY 20, 2019 • BY [SAM PROCTER](#)

---

Get updates on our latest  
work.

Subscribe

 Get our RSS feed