An Examination of Software Tool Features Needed to Help Secure Energy Delivery Industrial Control Systems

Taylor Andrews, Allen Moulton, and Stuart Madnick

Working Paper CISL# 2018-07

August 2018

Cybersecurity Interdisciplinary Systems Laboratory (CISL)
Sloan School of Management, Room E62-422
Massachusetts Institute of Technology
Cambridge, MA 02142

Working Paper – August 2018

An Examination of Software Tool Features Needed to Help Secure Energy Delivery Industrial Control Systems

Cybersecurity at MIT Sloan (CAMS)
Massachusetts Institute of Technology
Cambridge, Massachusetts, United States of America

Taylor Andrews
Allen Moulton
Stuart E. Madnick

MITsdm

Table of Contents

Keywords: Systems Thinking, Cybersafety, Cyber Safety, Systems Theory, Systems Theoretic, Systems Theoretic Cybersafety, STAMP, STPA, STPA-Sec, STPA-SafeSec, Industrial Control Systems, ICS, Operational Technology, OT, Cyber Security, Cybersecurity, Cyber Physical, Cyber-physical, Cyberphysical, Software Tools, Microsoft Office, Draw.IO, XSTAMPP, STAMP Workbench, RM Studio, SafetyHAT

Table of Figures

1. Abstract and Motivation

In December 2015, coordinated cyberattacks targeting Ukrainian power distribution systems' information technology (IT), industrial control systems (ICS), and operational technology (OT) resulted in physical damage to Ethernet serial converters, intentional disabling of distribution facility backup generators, denial of service attacks on customer support call centers, and permanent destruction of workstation hard drive data, causing temporary citywide power grid failure that affected 225,000 people [21]. It was discovered the attack in Ukraine took place months after initial network penetration, after extensive surveillance and data gathering was first performed, indicating cyber attackers are attempting to prolong intrusions and avoid detection in an effort to practice, simulate, and perfect militarized-style attack architectures to maximize damages [21]. In March of 2018, after joint collaboration, the U.S. Department of Homeland Security and FBI released an alert that documented details of a multi-year, extensive surveillance and intrusion campaign from state sponsored "threat actors" that widely penetrated U.S. energy distribution systems with malware designed to enable covert remote access and technical manipulation abilities, to be able to perform similar attacks on American power grids [16].

The growing number of cyber-physical intrusions to energy distribution systems require preventative, structured cybersecurity analysis to produce attack scenarios, causal factors, design changes, and new requirements to secure energy systems before systems are compromised, ideally at system design and development time. Hazard analysis, safety analysis, and reliability analysis must no longer be considered solely from the point of view of single component, engineering-based failures, but must all evolve to foresee premeditated, malicious, and coordinated actions of human organizations that intentionally cause disastrous multi-component failure scenarios after careful reconnaissance and reverse-engineering. In this paper, we explain *systems theoretic cybersafety,* we document an exploration of software tool features that support *systems theoretic cybersafety analysis automation,* provide a detailed list of *STAMP software tool specification requirement areas to consider* when designing future systems theoretic cybersafety tools, and finally we include some data structures for *systems theoretic cybersafety analysis information organization*. Through an energy distribution system example in Section 3, we also demonstrate how one currently may use software tool features to perform systems theoretic cybersafety analysis using STAMP, and produce system changes to defend and defeat when analyzing existing systems or designing new ones.

1.1 Why is Systems Theoretic Cybersafety based on STAMP?

STAMP is an accident causality model developed by MIT's Aeronautics and Astronautics Department in the early 2000s by Nancy Leveson, which then led to the formation of STPA (System-Theoretic Process Analysis) for preventative safety and hazard analysis [28]. In her paper, "A Systems Theoretic Approach to Safety Engineering," Nancy Leveson differentiates that "in STAMP, accidents are conceived as resulting not from component failures, but from inadequate control or enforcement of safety-related constraints on the design, development, and operation of the system" [25]. The 2018 STPA Handbook written by Nancy Leveson and John Thomas states "STPA is a relatively new hazard analysis technique based on an extended model of accident causation. In addition to component failures, STPA assumes that accidents can also be caused by "unsafe interactions of system components, none of which may have failed" [17]. Systems theoretic cybersafety analysis is rooted in this systems theoretic accident

model called STAMP due to its focus on loss of system control, emergent interactions and behaviors, and inadequate system "controller" decisions as potential causes that produce vulnerable system states, during which system hazards and accidents can take place. "Controllers" in systems can be either human or technical. Controllers use logic and algorithms to make decisions they believe are correct, but unfortunately, unbeknownst to them, they may not have a clear picture of what the true state of the system is; a malformed controller "process model" may lead them to believe otherwise. A process model is nothing more than the "mental model" of the controller; a human controller who has had procedural training (such as an operator or other employee) forms mental algorithms that allow them to independently view the state of their work environment, using their human senses to update their mental "process model," and then to continuously make decisions (through the logic provided to them through training) on how to perform their job responsibilities. Conversely, but in the same spirit, a technical controller will possess numerous specific values in its memory (or perhaps sometimes periodically persisted to disk for backup) that correspond to the technical controller's process model. The internal software algorithm programming is the technical controller's "procedural training," and the technical controller's "view" of its "work environment" comes from considering the digital information in its process model, helping it decide how to perform its job function as well. "Control actions" and "feedbacks" are also passed between controllers and the processes they control to enlighten the rest of the system about the state of system operations, so all controllers can respond and manage the system together. It is believed this systems-theoretic view of accidents and hazards (STAMP analysis) can also be applied in a security context for cybersecurity, providing the theoretical foundation for systems theoretic cybersafety.

Full Definition	*Abbreviated Mnemonic*
Systems-Theoretic Accident Model and Processes	STAMP
Causal (post) Analysis based on STAMP	STAMP/CAST
Systems Theoretic Process Analysis	STAMP/STPA
Systems Theoretic Process Analysis for Security	STAMP/STPA-Sec
Safety Analysis (STPA) and Security Analysis (STPA-Sec) [20]	STAMP/STPA-SafeSec

Table 1.Systems Theoretic Cybersafety STAMP Nomenclature

Some software tools aim to support STAMP/STPA analysis, so an in-depth examination of existing features was first performed to examine what features currently exist, and to develop a list of feature requirement areas where future development of tools must focus to support systems theoretic cybersafety. This paper aims discuss a way to evaluate and compare STAMP software tools when integrating them into existing systems engineering toolchains, existing software tool features available for performing STAMP analysis (using an energy distribution industrial control system example), and finally a promising list of systems theoretic cybersafety software feature requirement areas to inspire future tool development.

STAMP analysis was used by the Composite Information Systems Laboratory (CISL) to explore and document the Stuxnet cyberattack, which targeted Iranian nuclear enrichment centrifuges, and root-cause the component interactions that had taken place [29]. STAMP analysis was also used by CISL to explore and document the TJX cyberattack, which resulted in the theft of "millions of customer's payment card data" and "financial losses amounting to over $170 million" [30].

1.2 Identify Energy Distribution Cyber Weaknesses Before Attackers Do

Significant dangers arise from dutifully engineered, thoughtfully timed, intentional attacks that inject, omit, or otherwise tampering with specific control actions and feedbacks in the system to carefully deviate controller process models (discussed in Section 1.1) to cause intentionally dangerous and unexpected interactions between components of the system. Modern cyberattacks are now repeatedly demonstrating this type of sophistication [21][29]. Systems theoretic cybersafety is currently being performed on MIT's Cogeneration Plant by Cybersecurity at MIT Sloan (CAMS), and structured safety analysis has been performed on nuclear facility industrial control systems [22]. These examples of structured security analysis methods aim to allow engineers and analysts to identify unacceptable losses and attack scenarios before they happen, and to generate safety constraints, requirements, and architectural decisions that keep cyber-physical systems secure from cyberattacks. For this process, CAMS recommends utilizing a system theoretic methodology called STAMP/STPA-Sec, along with aspects of a complimenting methodology STAMP/STPA-SafeSec [19] [20].

Dr. Bill Young extended STAMP's preventative STPA analysis for specific use in security, resulting in Systems Theoretic Process Analysis for Security (STPA-Sec) [19]. STPA-Sec aims to provide an analysis methodology that identifies a top-down approach to safety and security by identifying losses, hazardous vulnerabilities, security constraints, and unsafe control actions to systematically create causal factors and attack scenarios, identify security constraints, and necessary design or requirement changes in a preventative manner during system development. STAMP/STPA-Sec methodology steps are analogous to the 2018 STPA Handbook's foundational STPA steps, with extensions to produce malicious attack scenarios to drive secure system change. STAMP/STPA-SafeSec built upon STPA by integrating formalization of some general physical network component structures and network interactions to compliment the core functional control structures found in traditional STAMP analysis, and to organize attack scenarios into a hierarchy to attempt to provide an interface for external attack-tree strategies [20].

Currently engineers and analysts often perform hazard, safety, and cybersecurity analyses by hand or through general purpose document creation software platforms, using a variety of tools and methodologies to produce custom structured reports, which aim to disseminate security related information. We believe STAMP software tool features can further the ease of creating such reports, by leveraging computer systems to aid in STAMP information organization/linking/prioritization, control structure modelling, component interaction consideration, attack scenarios, and outputs to help make the system changes necessary to secure the cyber-physical system before it is penetrated, explored, and intentionally exploited. We conclude with a brief discussion of future systems theoretic cybersafety software areas of exploration, a basic framework for evaluating and comparing general STAMP software tool attributes when integrating them into existing systems engineering processes, and a detailed list of STAMP feature requirement areas. We hope this paper will help inspire further development of future software tools that help the analysist or engineer to performing systems theoretic cybersafety analysis, starting with energy delivery systems.

The following section offers an evaluation of the existing STAMP tool features aimed primarily at showcasing existing features required to perform STAMP analysis, specifically for systems theoretic cybersafety analysis of energy delivery systems. Please first note this paper was not intended to document bugs in the software tools, and instead aims to portray the landscape of available software tool features available to aid in performing systems theoretic cybersafety analysis of energy distribution systems.

This paper will next show the experiences gained from using some leading STAMP tools to duplicate S. Khan's cybersafety analysis of MIT's Central Utilities Plant (CUP) Facility: a cybersafety analysis of an energy distribution system (Section 3). All general purpose tool example screenshots showcasing MS Office and Draw.IO based STAMP analysis using general purpose document creation software platforms are from S. Khan's working paper and research materials. This gas turbine generator example was chosen to showcase many existing software features that already are available to support systems theoretic cybersafety, and to familiarize the reader with a systems theoretic approach to analyzing and securing energy distribution IT, industrial control, and OT systems through cybersafety. Again, as the reader progresses through an understanding of the steps in the example analysis, software features are highlighted that support the rapid, organized completion of analysis steps to produce structured reports for organized and professional cybersafety information dissemination. Considering systems theoretic cybersafety analysis findings can preventatively enlighten new architectural design decisions, system security constraints generation, and system requirement generation, or refine existing systems. This paper concludes by highlighting some areas of further research and feature development for new software features to move systems theoretic cybersafety software tool capabilities forward. This collection of content aims to help operationalize systems theoretic cybersafety analysis software tools to help secure energy distribution cyber-physical systems.

1.3 A Brief Summary of Existing STAMP Tools

There are various tools for STAMP assistance, in various stages of development, but all with the expectation that the user has some level of STAMP analysis experience. Some aim to support specific STAMP methods such as STAMP/STPA and, more recently, STAMP/STPA-Sec and STAMP/STPA-SafeSec. The following bullet lists shows the current landscape, and a list is also available online from the Partnership for Systems Approaches to Safety and Security (PSASS) [26].

Some STAMP tools are in the project planning phases and are considered in-development:

- A separate partnership between Stiki−Information Security and Zurich University of Applied Sciences advertises "a 2.5 year project" to create a STAMP/STPA plugin for their RM Studio product, Stiki's risk-management platform [2].

Some proprietary STAMP related tools are also available for purchase:

- Safeware Engineering Corporation offers a proprietary product SpecTRM that uses a requirements language to produce models of software that can "support execution of the specification as well as automated safety analyses" [3].
- Sparx Systems' Enterprise Architect UML/SysML modeling tool had offered a STAMP/STPA extension SAHRA, but it now claims to be transitioning to a successor "ANSHIN," which will "soon be available" [4].

Finally, three more STAMP tools are available for immediate and free evaluation and are evaluated in this paper:

- The U.S. Dept. of Transportation has released a STAMP/STPA tool aimed at transportation systems, called the Safety Hazard Analysis Tool (SafetyHAT) [5].
- STAMP Workbench (iSTAMP) was created by Japanese Information-technology Promotion Agency (IPA), advertising STAMP/STPA support. IPA announced iSTAMP as open source in 2018, and a free productized version of the project was released branded as STAMP Workbench [1] [27]. The product is Java based and cross platform, built to run on various PC operating systems.
- The University of Stuttgart's XSTAMPP Platform is a diverse open source STAMP software platform, claiming to support a wide array of STAMP methods [6]. The product is Java based and cross platform, built to run on various PC operating systems.

The following general-purpose drawing tools also implicitly support STAMP analysis. The following software tools were also evaluated for supporting STAMP analysis:

- Microsoft Office (MS Office) (Proprietary)
  - Microsoft Word
  - Microsoft PowerPoint
  - Microsoft Excel
  - Microsoft Access
- Draw.IO (https://www.draw.io) (Evaluation Versions)

2. A STAMP Software Tool Evaluation Framework

Some considerations for STAMP software tools were formalized that focused on safety specification categories that STAMP software tool component classifications when considering common software standards such as IEC 61508 (E/E/PE), EN50128 (Railway), ISO26262 (Automotive), and finally D0-178C/D0-330 (Aerospace & Defense) [7]. These qualifications aim to quantify potential dangers from software tool failures in a safety sense, rather than how a systems engineer would integrate STAMP tools into their existing processes and software toolchains. When considering software tools for effectiveness in automating and supporting systems theoretic cybersafety analysis of complex cyber-physical systems, more managerial as well as technical categories must be considered for socio-technical completeness. Deciding how different STAMP software tools may or may not compliment an existing IT system architecture and as systems engineering toolchain is important for integrating STAMP tools into existing processes. This framework first aims to expand these considerations by documenting a socio-technical set of general STAMP software tool attributes, spanning from common traditional software architectural decisions to IT managerial concerns. The framework then aims to set a future direction for STAMP software tool development by describing some future research areas, and a preliminary set of systems theoretic cybersafety software tool feature requirement areas to leverage STAMP. This paper then evaluates some popular, widely available STAMP tool features to document a collection of novel, existing STAMP-specific software tool features that aid in operationalizing systems theoretic cybersecurity and cybersafety analysis. Complementing the general software attribute consideration framework, a preliminary list of STAMP-specific feature set areas is provided to help guide future software tools supporting analysts performing cybersafety analysis on energy distribution systems. The paper concludes with a discussion of possible future feature development areas, a general STAMP software attributes framework evaluating a popular STAMP tool (Appendix B – STAMP Software Tool General Attribute Framework), a detailed list of feature set areas on which future systems theoretic cybersafety tool development should focus (Appendix A – Detailed Categories of Software Features Supporting STAMP Analyst Items), and an initial data structure and organization for systems theoretic cybersafety analysis data.

2.1 General Software Features and Architectural Decisions

The STAMP tool evaluation begins with a traditional collection of software product attributes in a table showcasing general software attributes. This captures a set of common software attributes generally considered when comparing systems engineering software products for a production application, such as technical details, software architecture information, acquisition availability and type, collaboration and teamwork features, and other managerial and logistical concerns. In this respect, we believe STAMP software tools can be first evaluated in a somewhat high-level, general way, allowing similar comparison to other software products, before moving into STAMP methodology specific features (Section 3 and Appendix A – Detailed Categories of Software Features Supporting STAMP Analyst Items). Appendix B – STAMP Software Tool General Attribute shows general STAMP software attributes and how an example STAMP software tool (XSTAMPP) is classified using them.

2.2 STAMP-Specific Software Features

The compliment to the general software attributes evaluation provides STAMP-specific feature areas that support completing STAMP analysis in an optimized, engaging, and rapid way to support systems theoretic cybersafety. A detailed list of STAMP-specific software feature areas follows in Appendix A – Detailed Categories of Software Features Supporting STAMP Analyst Items. In the same spirit of STAMP's top-down approach, the features were first aggregated into 15 general software feature requirement areas.

1. Features supporting methodology guidance
2. Features supporting identification of system purpose and description
3. Features supporting the identification of system goals
4. Features supporting the identification of system accidents/losses
5. Features supporting the identification of system hazards/vulnerabilities
6. Features supporting the identification of system-level constraints
7. Features supporting the identification of links between analysis objects
8. Features supporting the modeling of functional control structure(s)
9. Features supporting the modeling of physical control structure(s) [23]
10. Features supporting the identification of process model(s), their variables, and values
11. Features supporting the modeling of the causal control model [17, Appendix G]
12. Features supporting the identification of unsafe control actions (UCAs)
13. Features supporting the identification, exploration, and navigation of context table states
14. Features supporting the identification of causal factors and attack scenarios
15. Features supporting the identification of new requirements, architecture design changes, and safety / security constraints
16. Features supporting the automated creation of reports documenting cybersafety information

Figure 1. A Systems Theoretic Cybersafety Methodology Progression (STAMP)

3. Software Trials Using an Example Energy Distribution Analysis

This section details the evaluation of leading STAMP software platforms (XSTAMPP, STAMP Workbench, and SafetyHAT) and some general purpose document creation platforms (Microsoft Office & Draw.IO) when performing an example systems theoretic cybersafety analysis with the high-level STAMP-specific features from Section 2.2 in mind. Along the way, leading existing features are showcased to provide a thorough landscape of some innovative features currently helping mature STAMP software platforms into useful systems engineering tools that can help secure cyber-physical systems using systems theoretic cybersafety. In conclusion, beginning with Section 4, we present some high-level discussion points supporting longer term areas of new feature development, a thorough list of STAMP feature requirement areas (Appendix A), a list of general STAMP software attributes to consider when comparing or integrating STAMP software tools (Appendix B), and a preliminary description of a STAMP data structure organization (Appendix C). Section 4 and the appendices aim to introduce an initial starting point for formalizing new requirement specifications and design plans for future STAMP software tool projects that aim to support systems theoretic cybersafety, specifically for securing information technology, industrial control systems, and operational technology within energy generation and distribution facilities.

The following sub-sections of Section 3 describe software features found in general purpose document creation tools, along with some leading STAMP software platforms' optimized and STAMP-specific features. The STAMP platforms were first used to recreate S. Khan's systems theoretic cybersafety analysis of the MIT CUP Cogen Facility (a working paper in Cybersecurity at MIT Sloan), where systems theoretic cybersafety was used to analyze and secure an example generator turbine's fuel industrial control system. It is important to note that when features were considered for supporting systems theoretic cybersafety analysis, it was assumed the user of the software tools would already have at least some familiarity with STAMP/STPA, specifically the introduction to STPA found in Chapters 1 and 2 of the 2018 STPA Handbook [17]. However, in this paper, we assume the reader does not have any STAMP/STPA methodology experience, and we provide some background on the major steps required to perform systems theoretic cybersafety analysis using STAMP.

3.0.1 What Steps Are Needed?

STAMP software tools offer initial advantages over general purpose document creation tools by being able to provide an overall methodology guidance by showing an organized view of all the steps found in the process, with navigational links to views of each. SafetyHAT uses a main view that allows simple, organized access to the various areas of STAMP/STPA information used for systems theoretic cybersafety analysis.

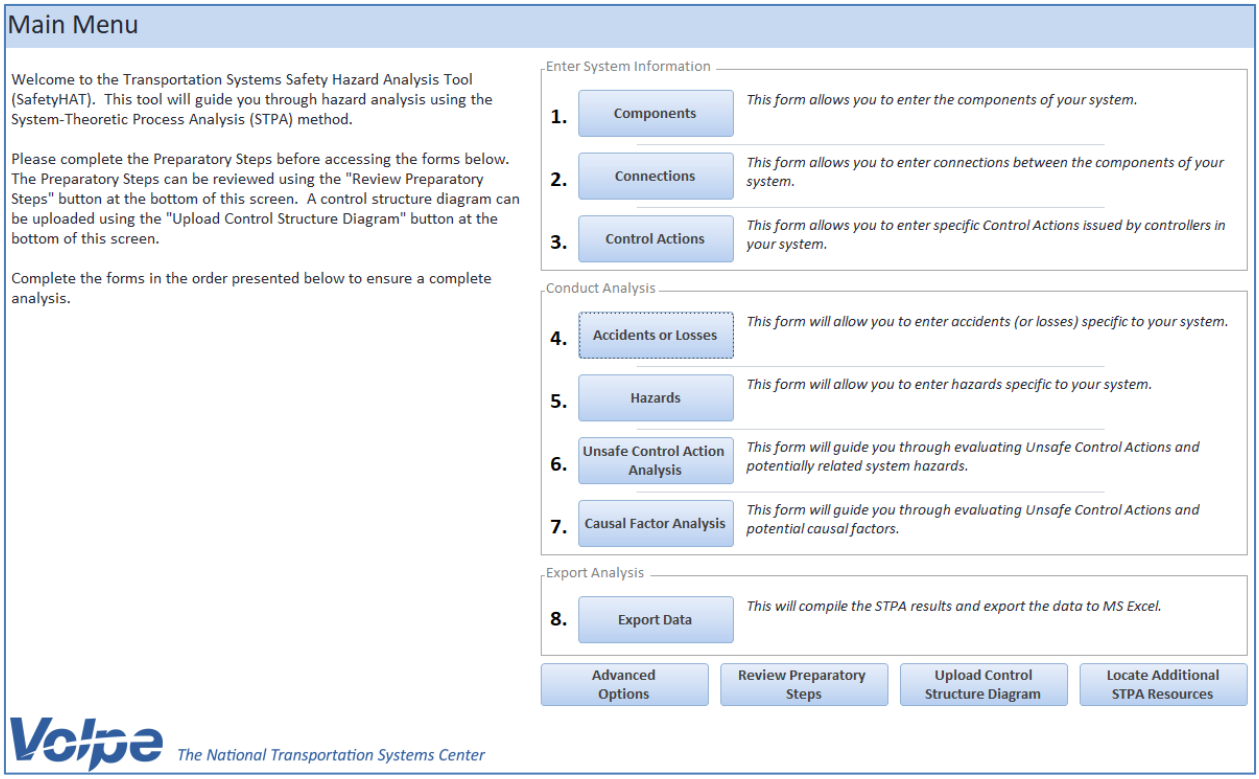


Figure 2. U.S. Dept. of Transportation Volpe's SafetyHAT MS Access Runtime System - Main View

XSTAMPP and STAMP Workbench also provide a structured, organized approach to conveying how to progress through the analysis methodology, highlighting access to the major steps a hierarchical rooted tree view. Figure 3 shows the user interface view that XSTAMPP uses to provide a navigational map through the given STAMP methodology, and Figure 4 shows STAMP Workbench's version. The workflow viewing structure is arranged in a rooted tree structure, allowing for an iterative, non-linear progression through the steps in the analysis, specifically allowing the user to jump forwards and backwards to whichever step they desire to work on or revisit, and potentially can decouple information for collaboration optimization.

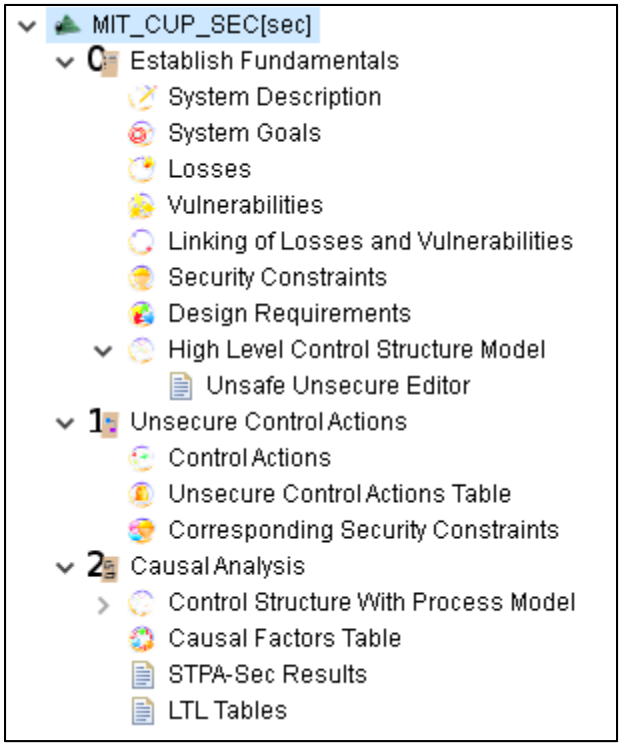


Figure 3. XSTAMPP Hierarchical View of STPA-Sec Analysis Items

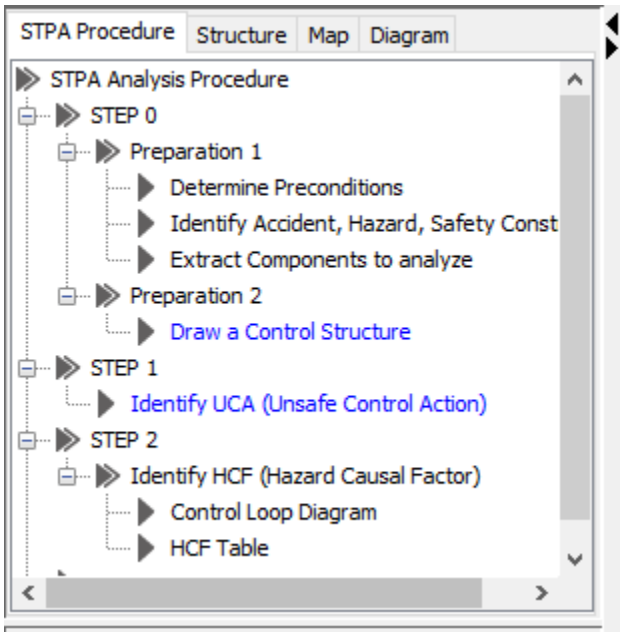


Figure 4. STAMP Workbench Hierarchical View of STPA Analysis Items

A hierarchical view to navigate through analysis provides flexibility and potentially enables collaboration through multiple separate views. Conveying the overall workflow of a STAMP methodology in such a way provides a concise summary of the areas of work required to progress through the analysis, an area where general purpose tools like Microsoft Office (MS Office) and Draw.IO rely on external documentation to guide the creation and structuring of the information. XSTAMPP and STAMP

Workbench both concisely provide a customized hierarchical workflow view of each STAMP methodology that an analysis works through (such as STAMP/CAST, STAMP/STPA, and STAMP/STPA-Sec), but software tool considerations for methodologies other than STAMP/STPA-Sec and STAMP/STPA-SafeSec such as STAMP/CAST (post mortem and root cause Causal Analysis Based on STAMP) are outside the scope of this paper.

We will discuss requirements formally in future work, but we note that systems theoretic cybersafety navigation views should leverage Dr. Nancy Leveson's and Dr. John Thomas's March 2018 STPA Handbook Chapter 2 overview of STPA analysis in a complimenting way, ideally where the handbook could provide guidance when using systems theoretic cybersafety software tools [17].

3.1 STPA-Sec Methodology Step 1.1: Define the Purpose of the Analysis

Step 1.1 of STPA-Sec is to *define a system purpose statement.* An example robust structure for a system purpose statement is recommended as follows [19]:

"A system to do {what = purpose} by means of {how = method} in order to contribute to {why = goals} while {constraints / restraints}."

The system purpose statement is analogous to a short yet complete thesis statement for the system purpose. Using rich text in MS Office, one can summarize systems in a concise, yet thorough way using a simple to-by-using structure. Framing the system purpose statement up sets the stage for the rests of the analysis and sets alignment for the top areas of consideration when considering what primary value function the system performs, what functional process the system uses to do it, and what physical form the system implements to provide such functionality.

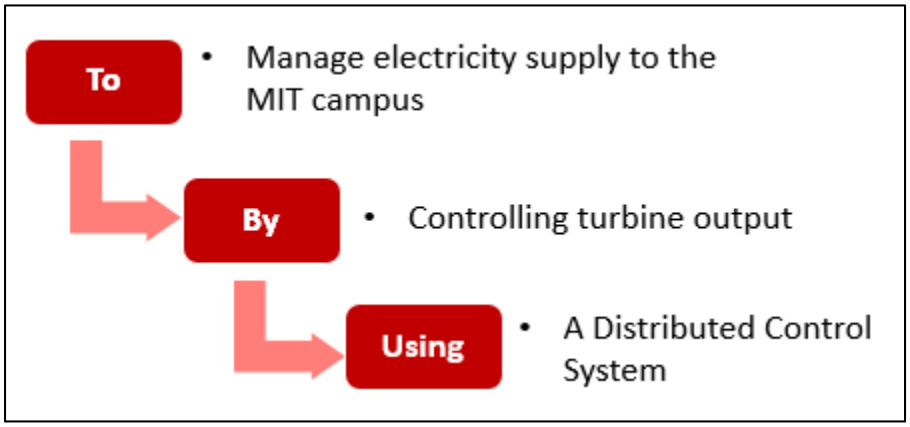


Figure 5. MS Office Example Rich Canvas Functionality for System Purpose

The system purpose definition also consists of a more in-depth description of the system, sometimes including more contextual information in the form of diagrams or tables. In this respect, Microsoft Office (MS Office) provides more flexibility than XSTAMPP in that it provides an opportunity to include a rich format for describing the high-level purpose and surrounding landscape of a given system under analysis, including links and images. Figure 7 shows the use of MS Office to provide this type of functionality by including an image showing where the system fits into an external surrounding context, while Figure 6 shows a plaintext implementation in XSTAMPP.

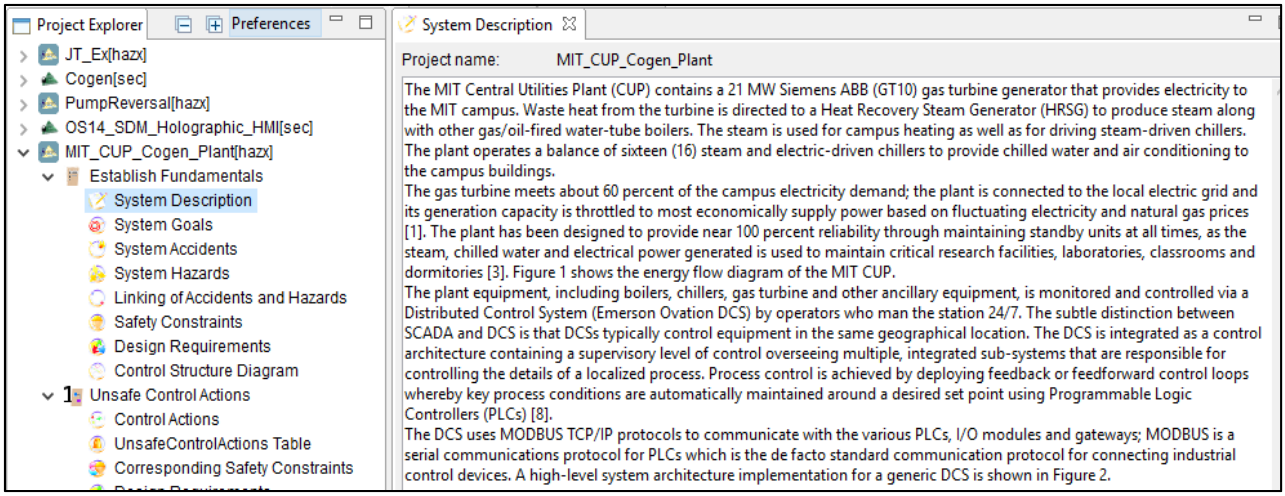


Figure 6. XSTAMPP Example Plaintext Functionality for System Purpose and Description

Part of the system purpose statement defines high-level goals of the system. In a general-purpose document-editing tool such as MS Office, this is most productively specified implicitly within rich text descriptions of the system (Figure 7).

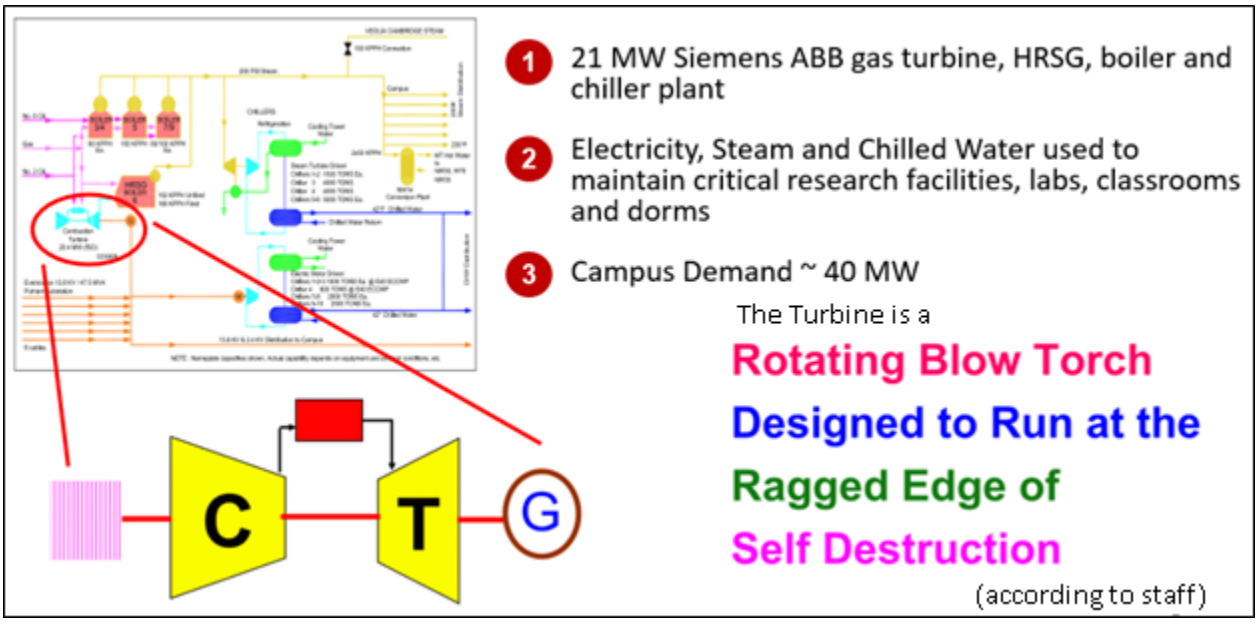


Figure 7. MS Office Rich Text Description of System and Goals

XSTAMPP claims to provide STAMP functionality to organize and identify system goals in a way a general-purpose document editor such as MS Office simply cannot. Figure 8 shows the goal manager in XSTAMPP. Although its description is limited to plaintext, the software allows for more organized storage of goal information that can be uniquely identified, linked to, and navigated, which supports traceability. When these types of features are fully functional, they allow for the rapid reordering, reprioritization, and automatic cross-referencing of the list items that are tracked with the system, such as losses, vulnerabilities, constraints, control actions, requirements, and possibly others, allowing for the enrichment of report information through software automation tools.

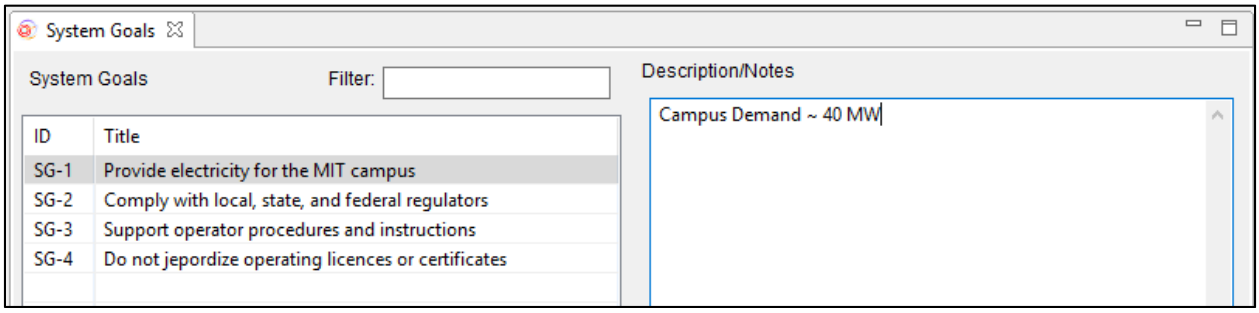


Figure 8. XSTAMPP Example Goals Management with Extended Plaintext Descriptions

3.1.1 STPA-Sec Methodology Step 1.2: Identify Losses

Step 1.2 of STPA-Sec is to *identify losses (Accidents in STAMP/STPA).* The STPA Handbook defines losses as:

"A loss involves something of value to stakeholders. Losses may include a loss of human life or human injury, property damage, environmental pollution, loss of mission, loss of reputation, loss or leak of sensitive information, or any other loss that is unacceptable to the Stakeholders" [2].

The example systems theoretic cybersafety analysis on the MIT CUP Cogen Power Plant used to evaluate STAMP software tools specifies these losses in plaintext table form. Figure 9 shows MS Office's functionality for quickly specifying losses, although the indexing and unique identification must be performed and managed by the user, a noted tedious and time consuming step identified by the author of the example MIT CUP Cogen cybersafety analysis, S. Khan. We believe these types of opportunities present themselves as areas of further formalization for the purpose of software automation, discussed further in 4. Conclusions and Appendix A – Detailed Categories of Software Features Supporting STAMP Analyst Items.

L-1: Death, dismemberment or injury to plant personnel
L-2: Loss of electrical power
L-3: Loss of equipment (financial/operational)
L-4: Loss of Revenue

Figure 9. MS Office Example Table for Specifying Losses

XSTAMPP automatically indexes the losses similarly to goals, allowing the user to enter each one, and provides a feature for an extended plaintext description. When user feedback ensures their

effectiveness, natural keyboard shortcuts can allow much more rapid entry and refinement of ordered lists such as losses and goals, and tools can help manage reprioritization of ordered list items by managing unique identifiers and data structures internally. The XSTAMPP loss manager also provides linking to vulnerabilities and safety constraints, which will be detailed in subsequent steps showing how linking is important for traceability.

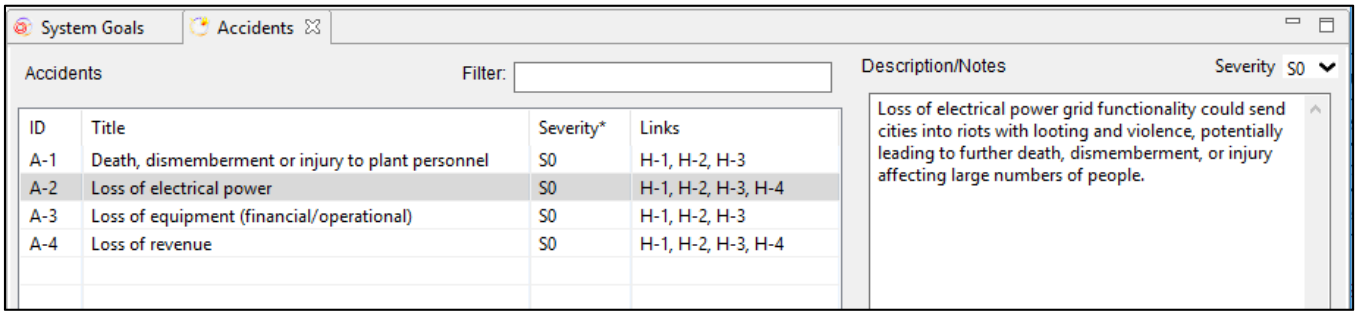


Figure 10. XSTAMPP Example Loss Manager with Extended Plaintext Descriptions

STAMP Workbench similarly identifies and allows the rapid addition of losses in the form of accidents (Figure 11). Although the losses are uniquely identified, like XSTAMPP, reordering and reprioritizing losses is not yet implemented in STAMP Workbench.

Accident ID	Accident
A1	Death, dismemberment or injury to plant personnel
A2	Loss of electrical power
A3	Loss of equipment (financial/operational)
A4	Loss of revenue

Figure 11. STAMP Workbench Specifying Losses

3.1.2 STPA-Sec Methodology Step 1.3: Identify System-Level Hazards (Vulnerabilities)

STPA-Sec Step 1.3 is to *identify system-level hazards (or vulnerabilities in STPA-Sec).* The STPA handbook provides the following definitions.

"Definition: A hazard [or vulnerability] is a system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to a loss" [17].

"Definition: A system is a set of components that act together as a whole to achieve some common goal, objective, or end. A system may contain subsystems and may also be part of a larger system" [17].

Similar to the previous steps, hazards/vulnerabilities can be specified rapidly in MS Office with a table, but requires tedious manual management of the unique identifiers, something that could be automated with software tools when internal data structure architecture allows for easy implementation of automation.

Hazards	**Related Losses**
H-1: CUP is operating beyond normal operational limits	L-1, L-2, L-3, L-4
H-2: CUP violates correct sequence of operations	L-1, L-2, L-3, L-4
H-3: CUP is unable to provide accurate feedback about equipment status	L-1, L-2, L-3, L-4
H-4: CUP does not respond to local (electricity) demand	L-2, L-4

Figure 12. MS Office Example Table Format Specifying Hazards (Vulnerabilities)

XSTAMPP again provides a manager for hazards/vulnerabilities, which features automatic indexing, and the ability to link back to specific losses. Figure 13 shows XSTAMPP's hazard/vulnerability manager, with extended plaintext description and loss linking functionality.

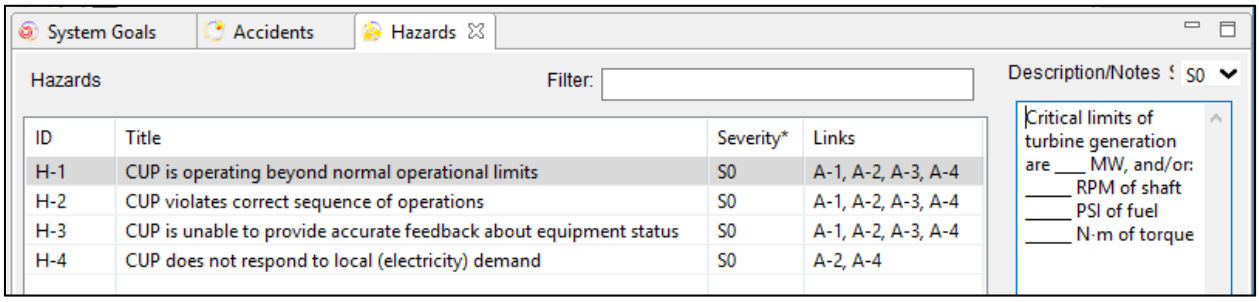


Figure 13. XSTAMPP Example Hazard/Vulnerability Manager with Extended Description & Linking

XSTAMPP and SafetyHAT include features that focuses directly on the linking of losses and vulnerabilities that aims to make the process much quicker and more organized than in a general editor such as MS Office. When doing example systems theoretic cybersafety analysis manually by hand, CAMS analysts had to reference and link to losses and vulnerabilities implicitly in the text, updating them as they changed, a time consuming and tedious task. MS Office (MS Word) provides a general cross-reference feature set, which may be able to be adapted as an initial way to reduce this workload within MS Office. Further formalized structure for systems theoretic cybersafety data structures could allow guidance for how the informational structure could be formed in software, which further increases the chances that new algorithms and external tools could then provide further automation and leverage the data later in the systems engineering process. An example time-saving improvement that standardizing

vulnerability parsing could enable is the auto generation of system level constraint syntax, saving the analyst or engineer from having to do simple, yet manual conversion (see Section 1.4).

Figure 15 shows XSTAMPP's hazard vulnerability and loss linking functionality, Figure 16 shows SafetyHAT's equivalent linking functionality, while MS Office only supports general purpose management of these ordered lists and each item's identifier using a traditional table (Figure 14).

Hazard	L-1	L-2	L-3
H-1	x	x	x
H-2	x	x	x
H-3	x	x	x
H-4	x	x	
H-5		x	

Figure 14. MS Office Example Hazard/Vulnerability & Loss Linking Management Functionality)

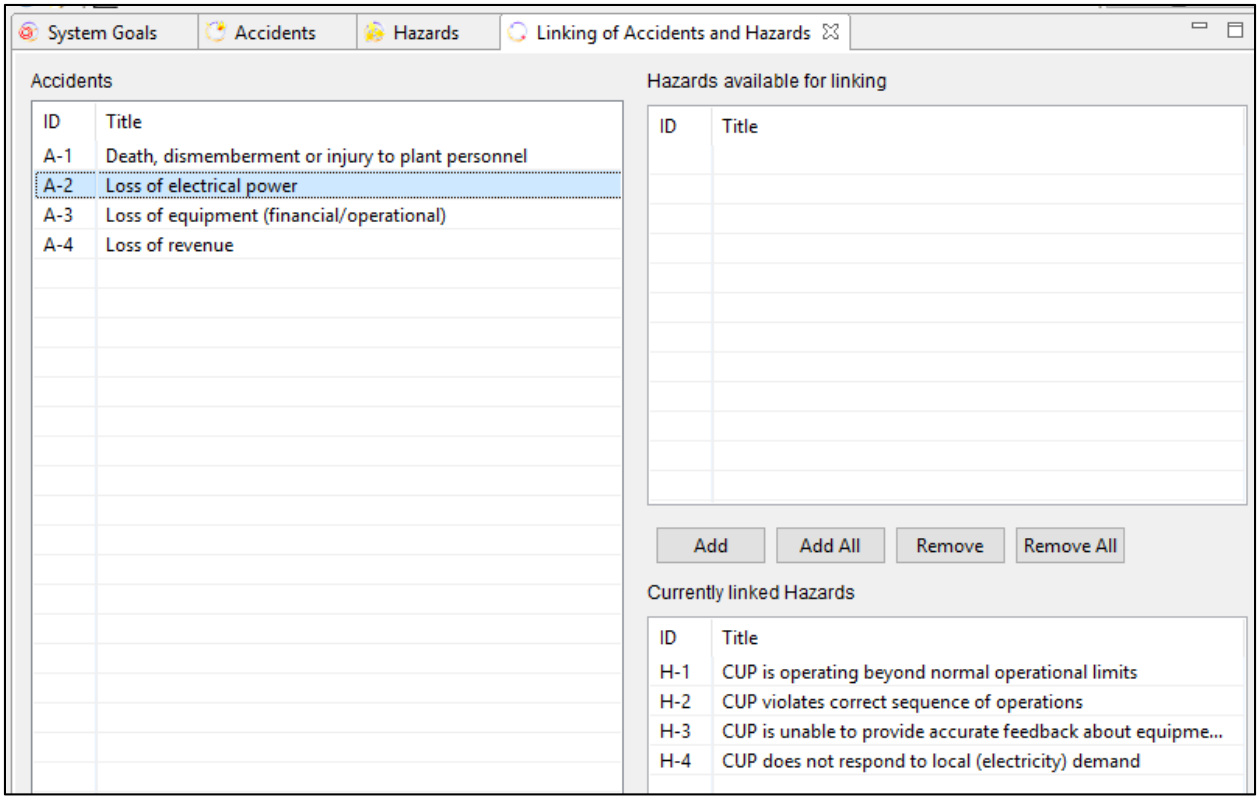


Figure 15. XSTAMPP Example Hazard/Vulnerability & Loss Linking Management Functionality

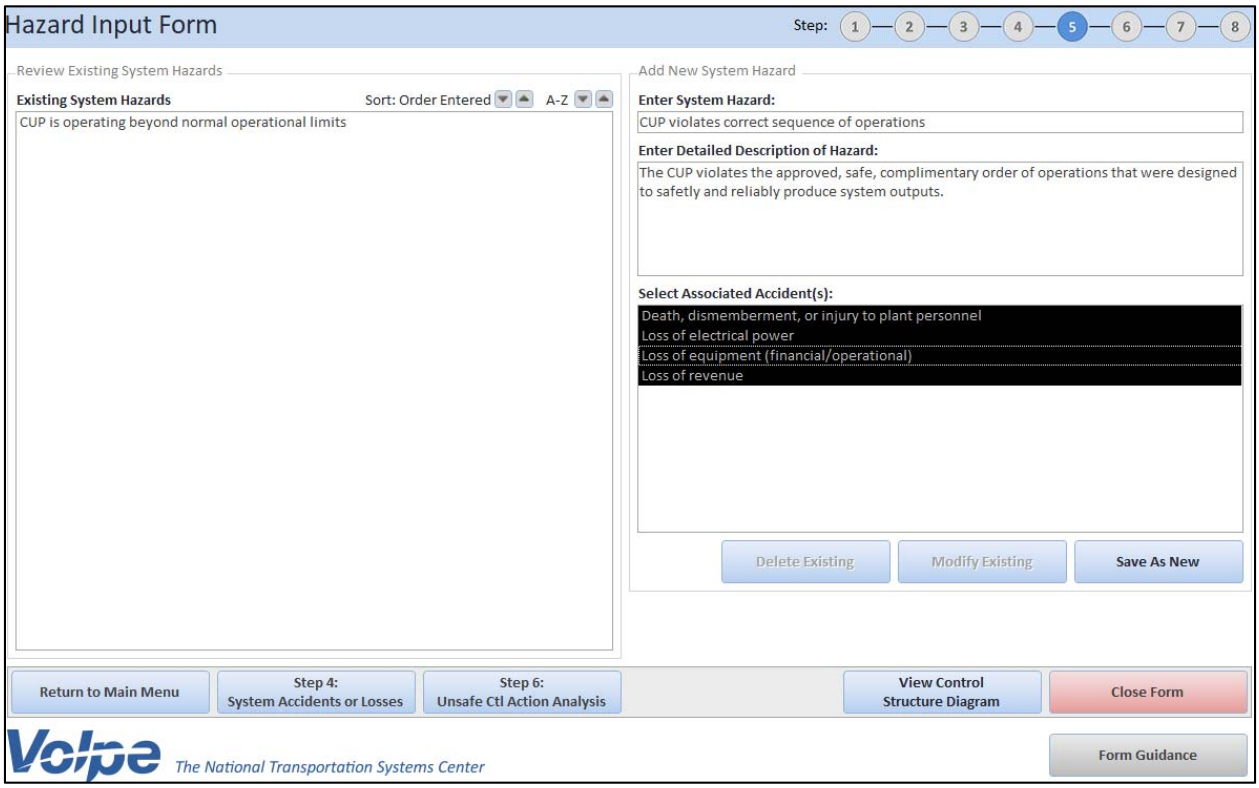


Figure 16. SafetyHAT's Hazard Input Form and Loss/Accident Linking Features

3.1.3 STPA-Sec Methodology Step 1.4: Identify System-Level Constraints

STPA-Sec 1.4 is to *identify system-level constraints.* The STPA Handbook provides the following definition:

"Definition: A system-level constraint specifies system conditions or behaviors that need to be satisfied to prevent hazards (and ultimately prevent losses)" [17].

Figure 17 shows an example of manual specification of these system-level constraints using a standard MS Office table.

Hazards	**Related Losses**	**System-Level Constraints**
H-1: CUP is operating beyond normal operational limits	L-1, L-2, L-3, L-4	SC-1: CUP must not operate beyond normal operational limits
H-2: CUP violates correct sequence of operations	L-1, L-2, L-3, L-4	SC-2: CUP must not violate the correct sequence of operations
H-3: CUP is unable to provide accurate feedback about equipment status	L-1, L-2, L-3, L-4	SC-3: CUP must not fail to provide accurate feedback about equipment status
H-4: CUP does not respond to local (electricity) demand	L-2, L-4	SC-3: CUP must not fail to respond to local (electricty) demand

Figure 17. MS Office Example Table for Safety Constraints

These system-level constraints are directly derived from the hazardous vulnerabilities in a form specifying what the system "must" do to remain in a safe operating state. Safety constraints must be specified using "must not" statements, to facilitate proving during verification and validation due to the ease of disproving "must not" statements compared with proving "must" statements.

Figure 18 shows XSTAMPP's system-level constraint manager providing extended description functionality as well as linking to accidents (losses) and design requirements.

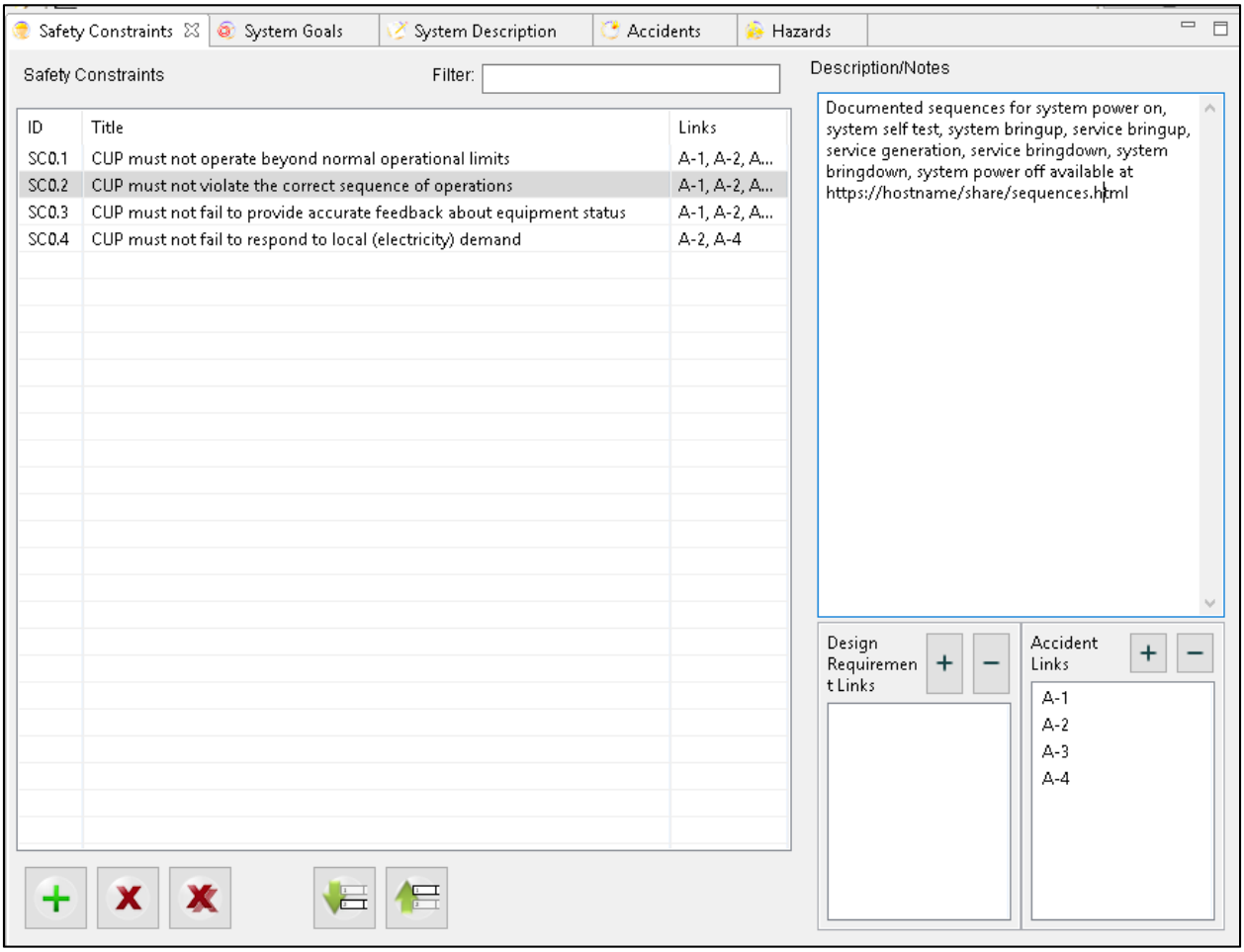


Figure 18 XSTAMPP Example Safety Constraint Manager and Linking Functionality

Analogous to the previously described linking functionality, in a general purpose document editor like MS Office, these links are specified and managed manually by the user in plaintext (Figure 19), which is noted to be a potentially time consuming process to refine and iterate.

from PLC. This scenario could be prevented by constraining the selection of VFDs to only those VFD types that do not allow reverse rotation via the PLC (as shown by component-level constraints CC-1.0 and CC-1.1 in Table 11).

Figure 19. MS Office Example Safety Constraint Linking Using Rich Text

The various structured linking that XSTAMPP provides for these various ordered lists is a noted improvement over managing the references between the various systems theoretic cybersafety artifacts and the management of their unique identifiers using general-purpose document editors such as MS Office. Linking of information allows computers to navigate the structure, potentially assisting in new ways. These features also allow for the dynamic re-identification of the listed items (goals, design requirements, losses, hazardous vulnerabilities, system-level constraints, etc.). Re-identification allows

cross-references to be updated for list items due to managing identifiers for each item, and allows the notion of priority in the lists (if deemed necessary). XSTAMPP offers an aesthetic level example of such functionality, although the order of identifiers remains unchanged**.** A function similar to a software development IDE's "refactor" global renaming functionality could likewise update all links to the item's new identifier as part of the reprioritization similar to updating a cross-reference in MS Word, saving time when revisiting these ordered items elsewhere.

Figure 20 demonstrates some first steps towards this functionality.

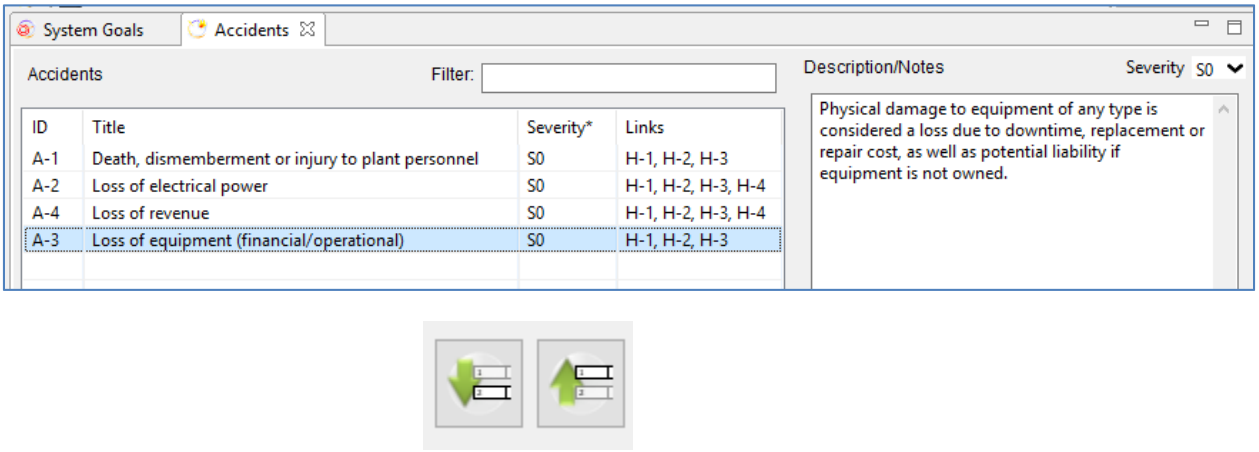


Figure 20. XSTAMPP's Preliminary Features for Ordered List Support

3.2 STPA-Sec Methodology Step 2: Model the Control Structures

After the purpose of the analysis is specified, the next step is to *build the control structure*, leading to the overall "Step 2" of STPA-Sec analysis. The STPA Handbook provides the following definition:

"Definition: A hierarchical control structure is a system model that is composed of feedback control loops. An effective control structure will enforce constraints on the behavior of the overall system" [17].

Figure 21 shows the general purpose drawing tool Draw.IO providing a general purpose graphical drawing canvas tool to facilitate creating a view of the functional control structure from the example systems theoretic cybersafety analysis, as well as contextual, rich information and images to help describe what the user is seeing.

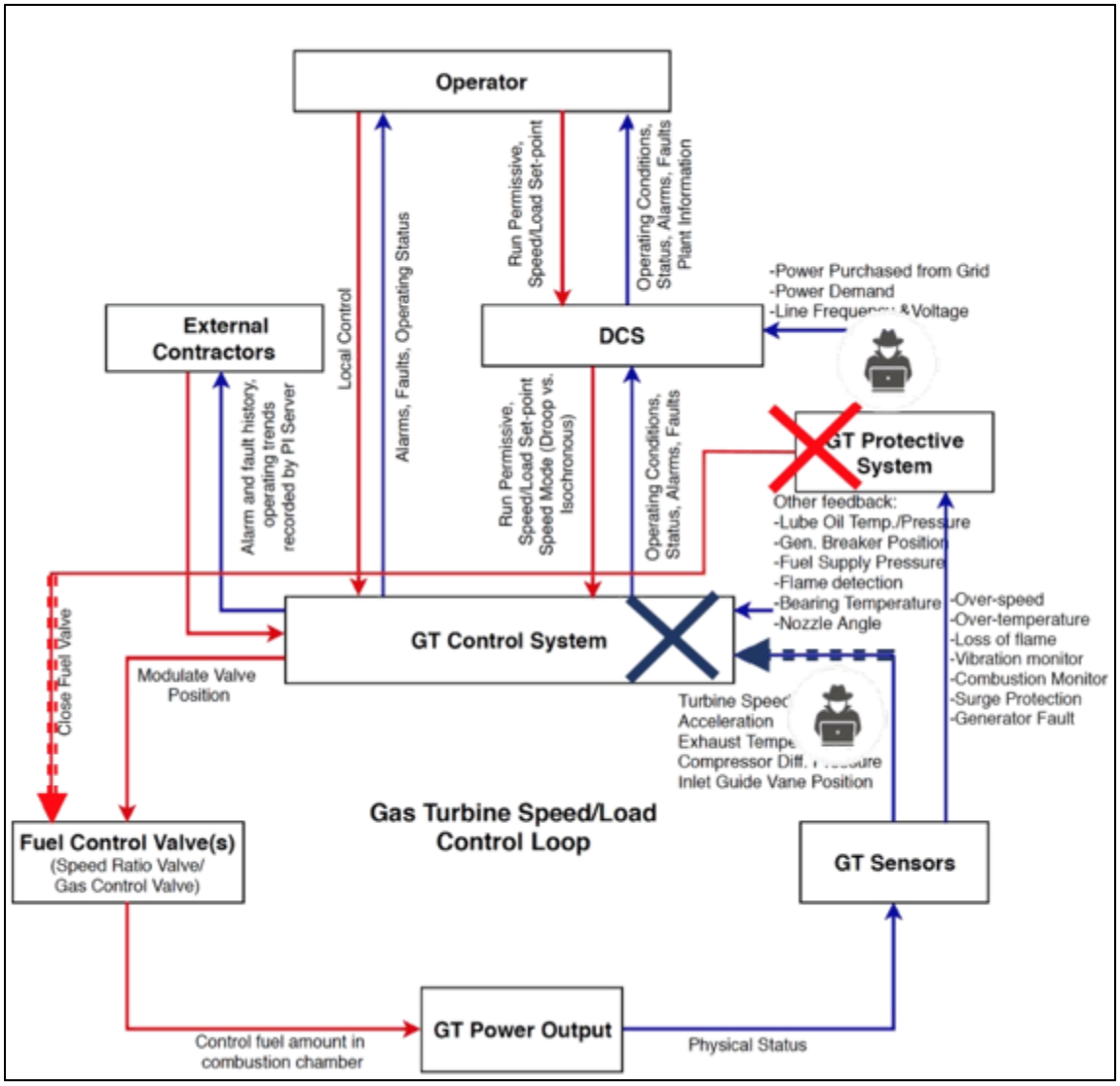
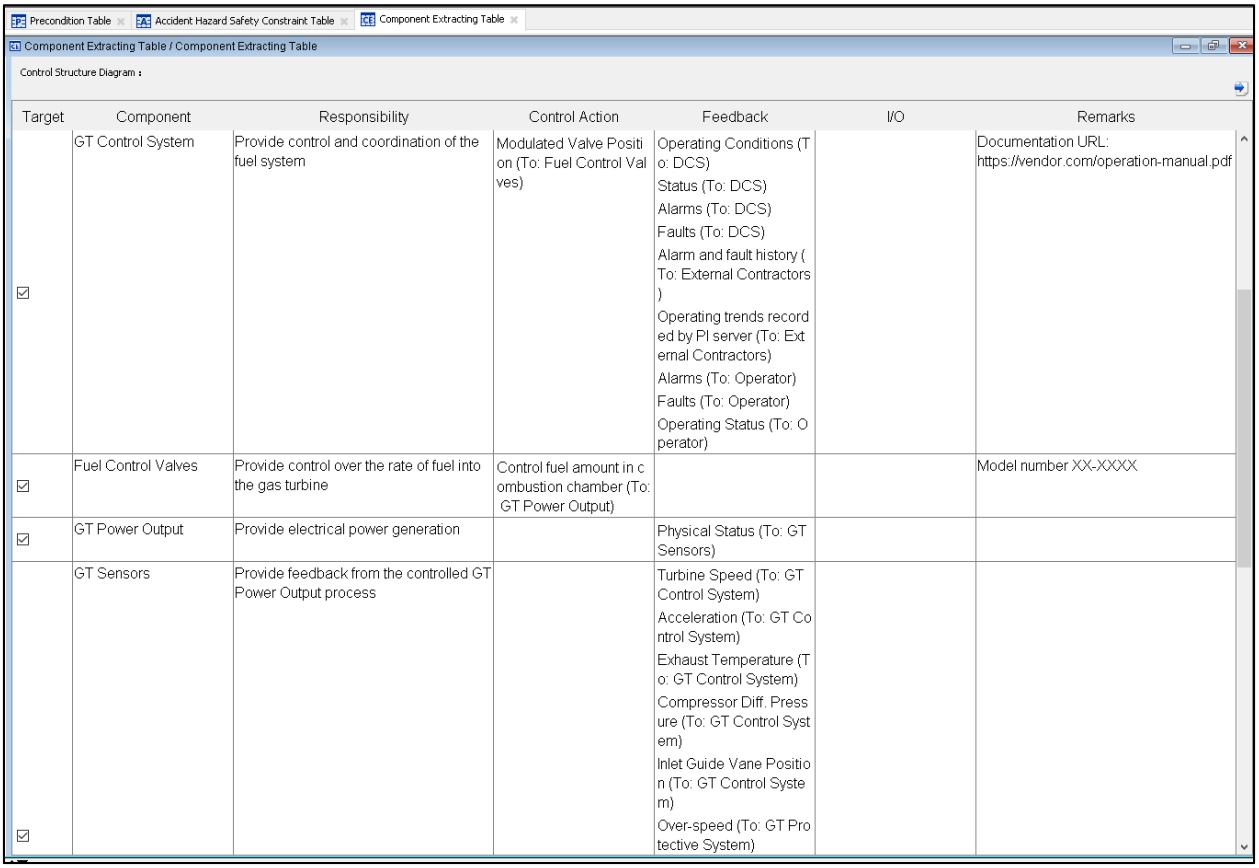


Figure 21. Draw.IO Example General Purpose Canvas Control Structure Drawing

MS Office and Draw.IO provides robust features for aligning and connecting objects on their well-funded, proprietary, mature, and general-purpose canvas. The auto alignment works very well and resizing and moving objects on the general-purpose canvas can produce simple diagrams quickly that are extremely customizable.

STAMP Workbench takes a unique approach to specifying the functional control structure by first allowing the specification of all components in a matrix view. This can be helpful once a preliminary control structure has been brainstormed on scrap paper or notes, and prevents the drawing canvas from becoming unruly as more functional components, control actions, and feedbacks are specified. STAMP

Workbench also supports the explicit enumeration of inputs and outputs for each controller, as well as extended plaintext descriptions of all functional components.



Target	Component	Responsibility	Control Action	Feedback	I/O	Remarks
☑	GT Control System	Provide control and coordination of the fuel system	Modulated Valve Position (To: Fuel Control Valves)	Operating Conditions (To: DCS) Status (To: DCS) Alarms (To: DCS) Faults (To: DCS) Alarm and fault history (To: External Contractors) Operating trends recorded by PI server (To: External Contractors) Alarms (To: Operator) Faults (To: Operator) Operating Status (To: Operator)		Documentation URL: https://vendor.com/operation-manual.pdf
☑	Fuel Control Valves	Provide control over the rate of fuel into the gas turbine	Control fuel amount in combustion chamber (To: GT Power Output)			Model number XX-XXXX
☑	GT Power Output	Provide electrical power generation		Physical Status (To: GT Sensors)		
☑	GT Sensors	Provide feedback from the controlled GT Power Output process		Turbine Speed (To: GT Control System) Acceleration (To: GT Control System) Exhaust Temperature (To: GT Control System) Compressor Diff. Pressure (To: GT Control System) Inlet Guide Vane Position (To: GT Control System) Over-speed (To: GT Protective System)		

Figure 22. STAMP Workbench's Matrix Specification for Functional Control Structure Generation

Listing and maintaining all control actions and feedbacks on drawing canvases can become tedious and time consuming, so STAMP Workshop includes a batch editor of control action and feedback lists, with “to target component” list boxes populated with the components currently loaded into the matrix. Figure 23 shows the control action manager and editor and Figure 24 shows the feedback manager and editor for specifying the control action and feedback structure of the functional control structure without having to drag, drop, position, and resize each on a drawing canvas.

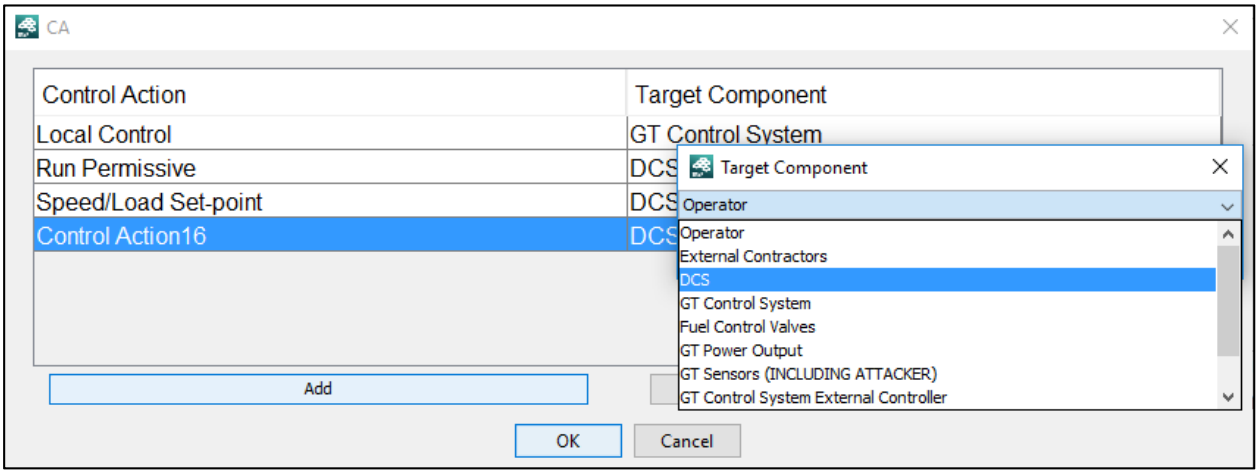


Figure 23. STAMP Workbench Control Action Manager and Editor

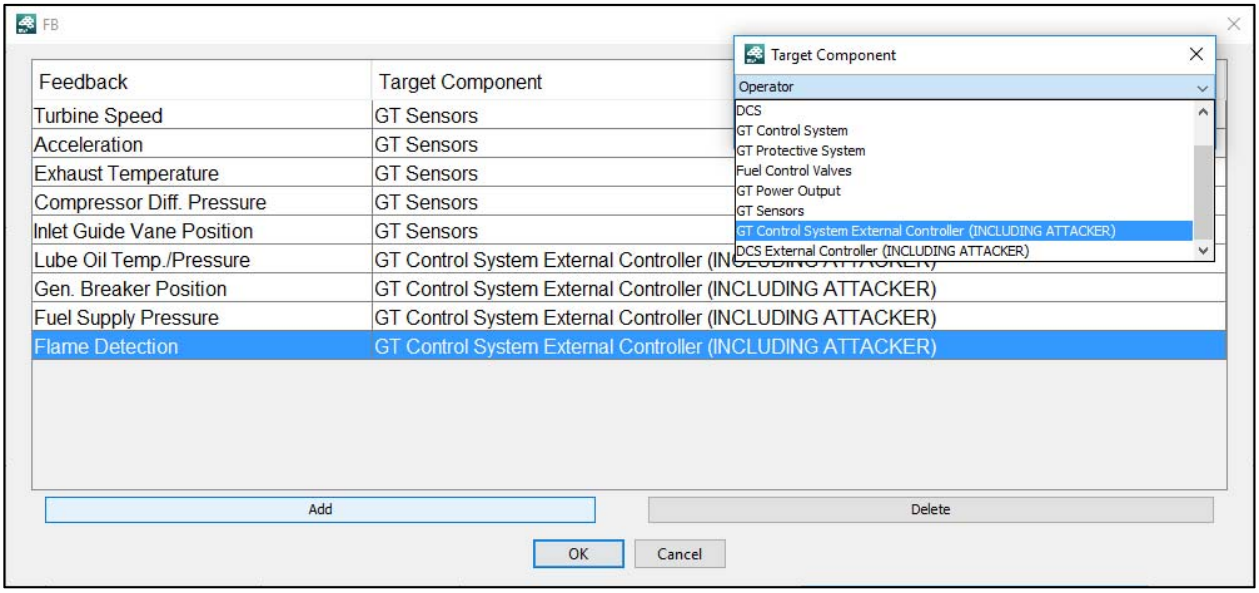


Figure 24. STAMP Workbench Feedback Manager and Editor

After generation, the controllers, controlled processes, actuators, sensors, control actions, and feedback are used to generate the relationships on a drawing canvas in STAMP Workbench. After a brief session of rearranging, editing, and refining connections, a control structure diagram is produced and shown in Figure 25.

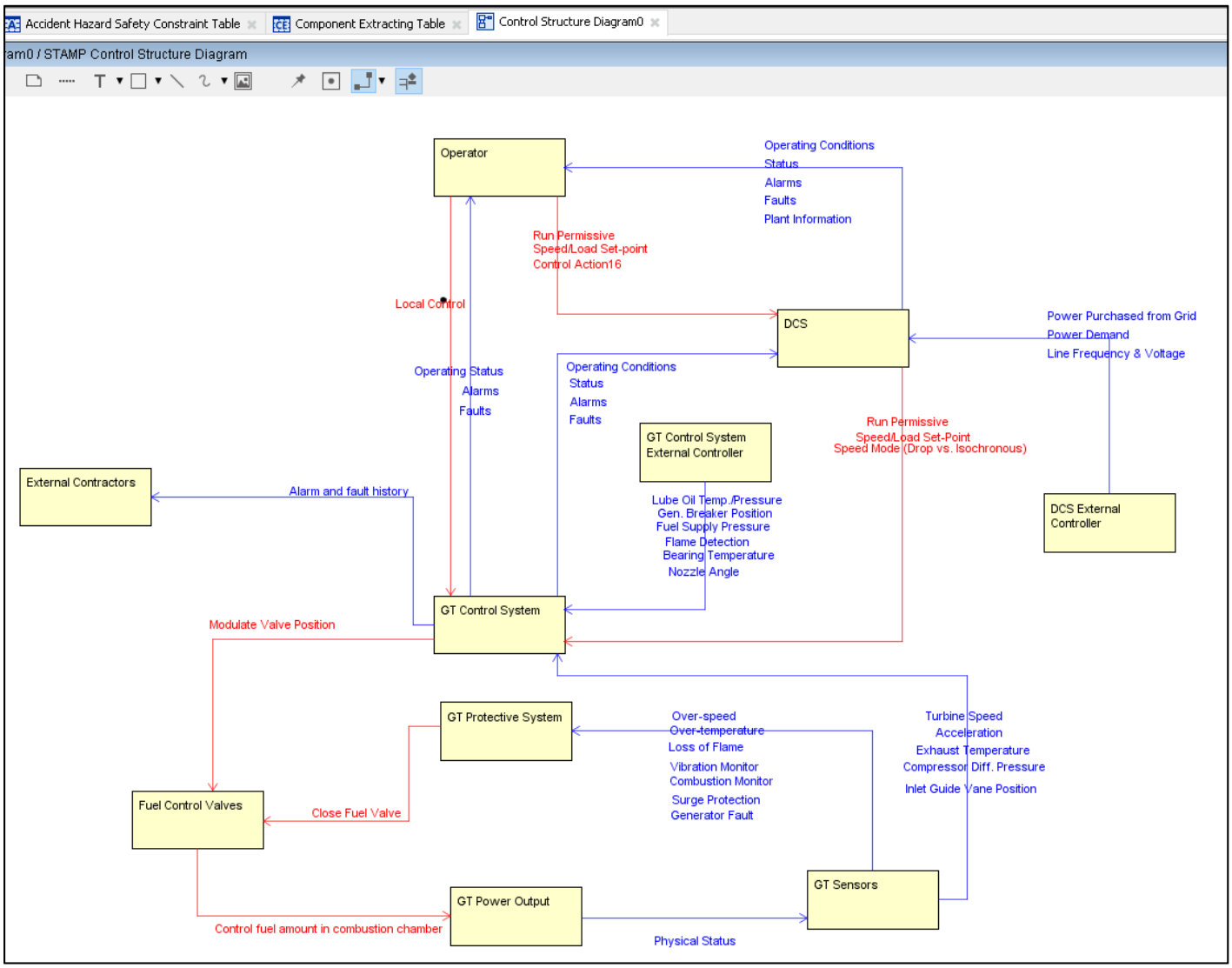


Figure 25. STAMP Workbench Control Structure Model After Generation and Brief Manual Rearrangement

XSTAMPP's free form drawing canvas has some improvements when compared to general purpose canvas tools and tools that aim to generate the control structure from a specification such as STAMP Workbench. XSTAMPP already boasts favorable graphical distinctions between Controllers, Control Actions, Feedbacks, Controlled Processes, as well as Actuators and Sensors, which greatly aid in the categorization and recognition of various control structure objects.

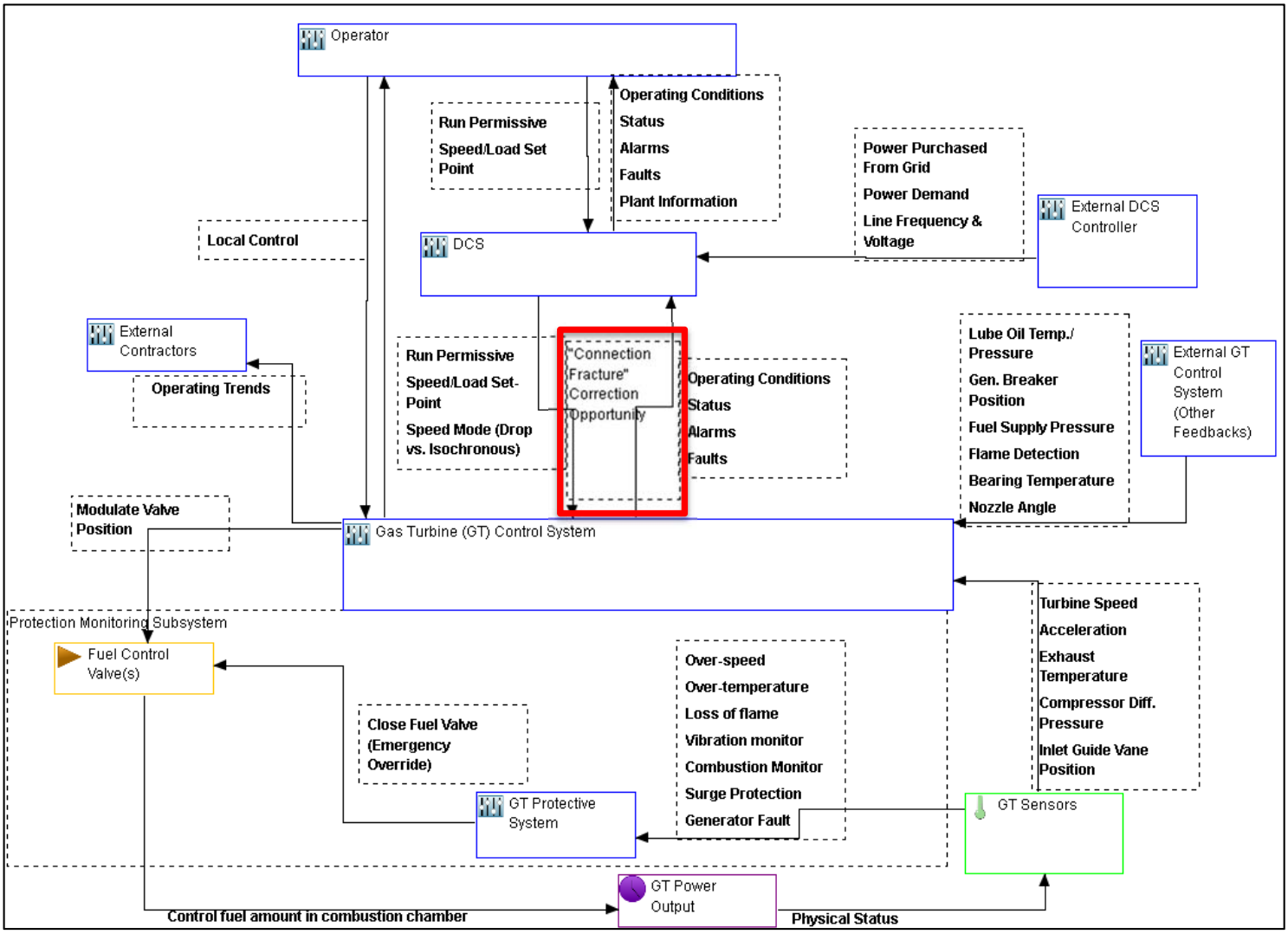


Figure 26. Using XSTAMPP's Drawing Canvas to Create a Functional Control Structure Drawing

Figure 26 and Figure 27 show a common two-dimensional canvas behavior in a red box when moving connected items – we will refer to them here as "connection fractures" (highlighted in red boxes). "Connection fractures" require a large overhead for aesthetic realignment, and reduce usefulness and efficiency of using drawing canvases if they appear too often, or are too time consuming to correct. Fortunately, mature and robust software drawing canvas tools can geometrically calculate when adjustments to the drawing objects can retain existing connections without introducing connection fractures, or even offer features to possibly fix existing ones, greatly speeding up the control structure creation process. Keyboard shortcuts for batch input of control actions and feedbacks further speed the

process up and organize the information in STAMP-specific forms internally that general purpose tools cannot.

Fortunately, some standard drawing canvas features already exist that aim to assist in preventing connection fractures when rearranging or fine-tuning drawing canvas object positions. These include intelligently rooting connections on object edges during small movements where the canvas calculates that geometrically, the connection is possible to remain intact (with no connection fractures) while the selected canvas object(s) move and or resize. This is a time-saving feature that allows analysts to rapidly create control structure drawings in a timely, organized, and professional manner and perfect the layout without inducing large amounts of rework. Figure 27 aims to demonstrate a typical user experience example when leveraging these types of drawing canvas features.

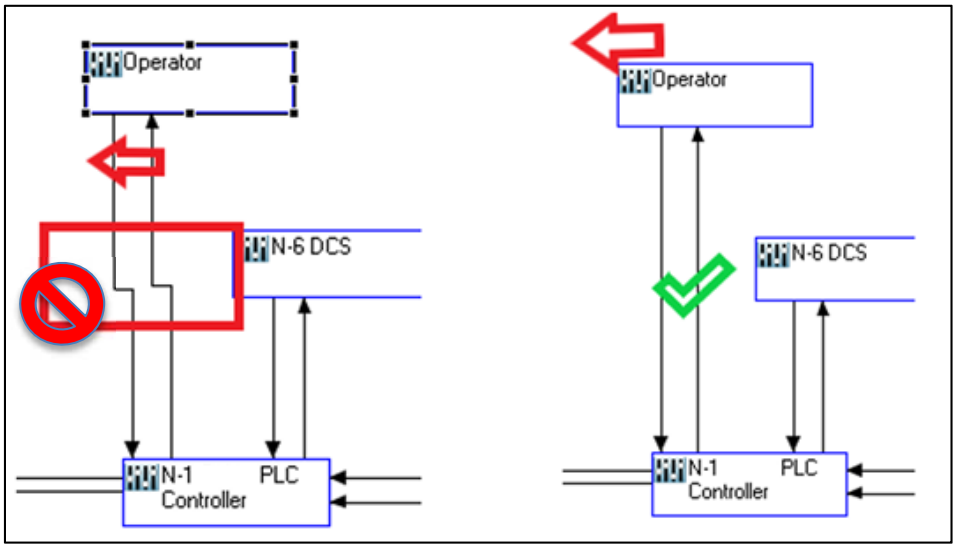


Figure 27. "Connection Fracture" Intelligent Drawing Canvas Optimizations

Drawing canvases aim to allow for the rapid iterative reformatting and expansion of drawings, as more objects are connected and subsystems are identified. Subsystem boundary identification is supported in some form in both the general purpose tools as well as XSTAMPP, however both of which offer no specific sub-system navigational functionality at this time. Another general-purpose drawing canvas tool, Draw.IO, can also produce clean control structure diagrams, although its functionality is comparable to a cloud-based version of MS Office's general purpose drawing features. It should be noted that all canvas type drawing tools evaluated all had some unique benefits, and software tools that provide accelerated and optimized control structure diagram design are believed to greatly aid in the rapid organization and automation of systems theoretic cybersafety reports. Some areas of improvement would include environmental input and output support, with features more powerful than dotted lines to imply system boundaries, such as support for multiple control structure drawings that provide different "views" of the model showing different levels of abstraction. XSTAMPP and general-purpose tools do not explicitly support assigning responsibilities for controllers. XSTAMPP and general-purpose tools do not explicitly support assigning unique identifiers to label/tag control actions, feedbacks, or other control structure objects for quick reference from elsewhere in the documentation or information. Human and technical controller distinction is not explicitly supported. The newer, more socio-technical general causal control model for casual scenario generation is not yet explicitly supported with specific features by XSTAMPP or general purpose tools [17]. Figure 18a shows the STPA Handbook's socio-technical causal control model.

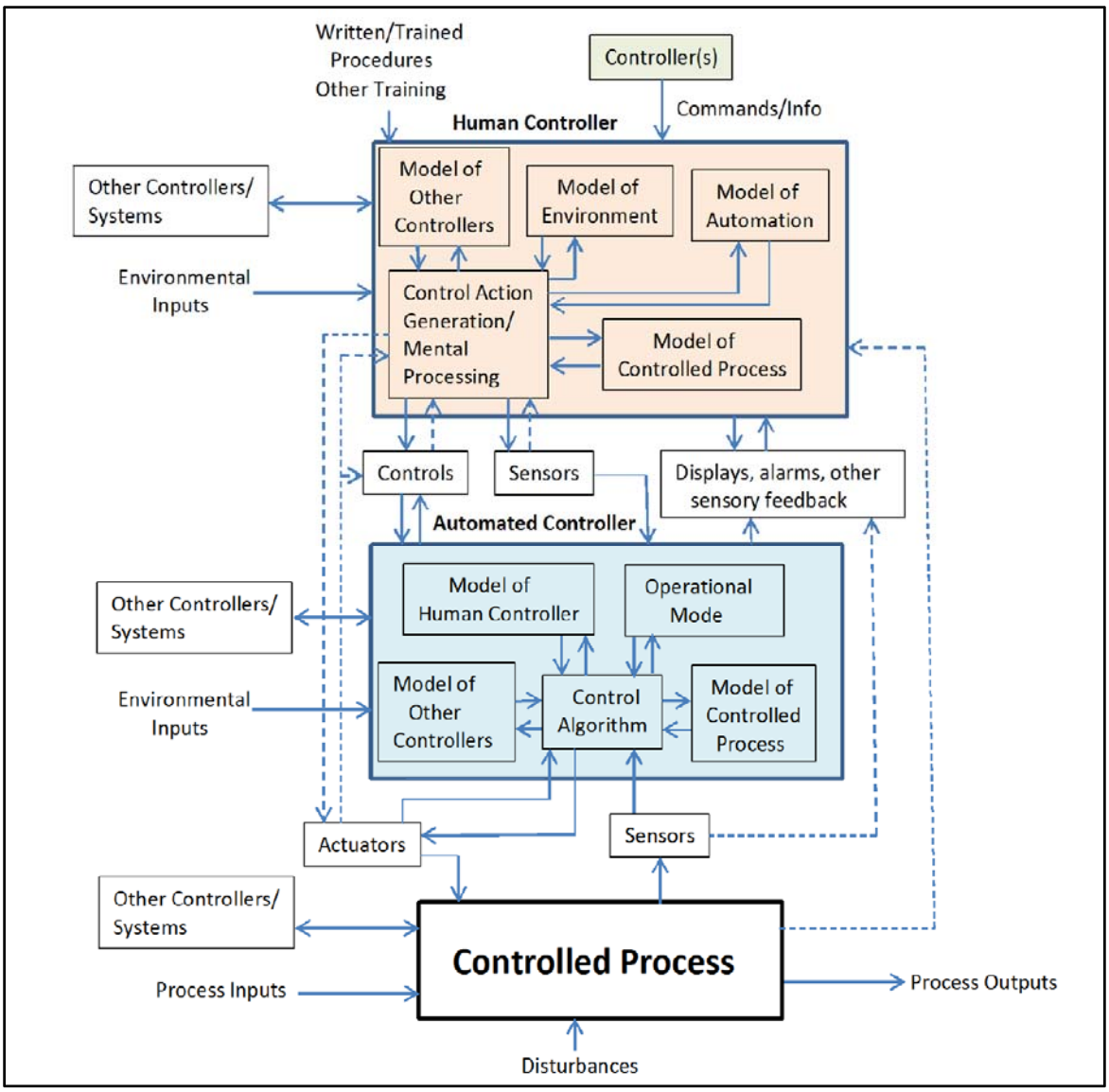


Figure 28. [17, Figure G-2]. Detailed Socio-Technical Causal Control Model

Where MS Office and general-purpose tools lack features, process models are explicitly supported in XSTAMPP. This enables the organization of a fair number of process model variables and their values, with enough freedom to customize the organization of the information. Physical controlled processes are also not currently differentiated. In some cases, tools offer no specific functionality to add detailed information about the process model elements for assumptions, notes, or to include URL references. There is currently no notion of variable types in STAMP software tools that would aid in software performing automation. Intelligent navigation of system states has already been shown to allow more flexibility and generate optimal sets of values to consider rather than requiring a human to enumerating all possible system states (discussed further in 3.3). Similar optimizations may be possible by first further formalizing specifications for a set of typed data structures that describe STPA for cybersafety and cybersecurity for use in interfacing between different system engineering tools and processes (see Appendix C – Preliminary Systems Theoretic Cybersafety Data Structure Organization).

After control structure creation, the next step is to design each controller's process model. Figure 29 shows XSTAMPP's features for process models. Support for extended process models in multiple controllers currently is necessary for software drawing canvases to allow organization of the process model variables and values.

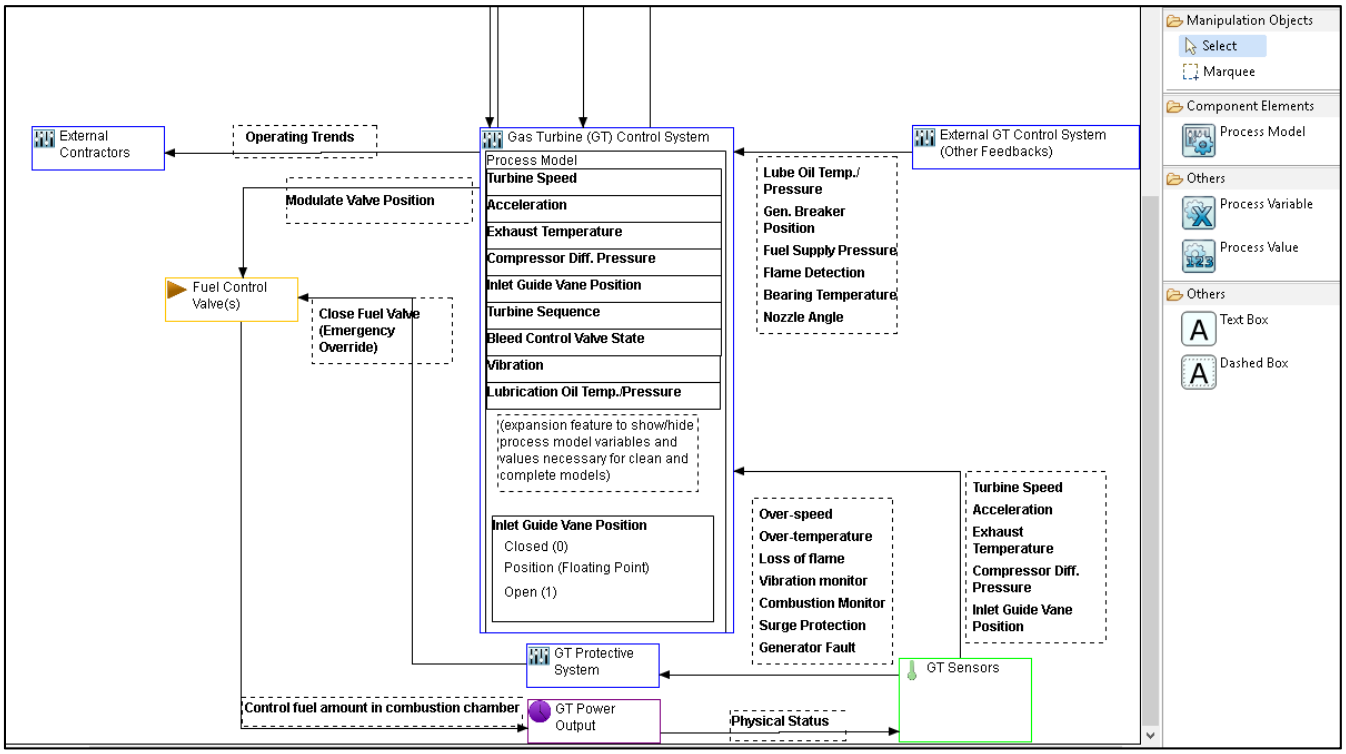


Figure 29. XSTAMPP Example Process Model Specific Features

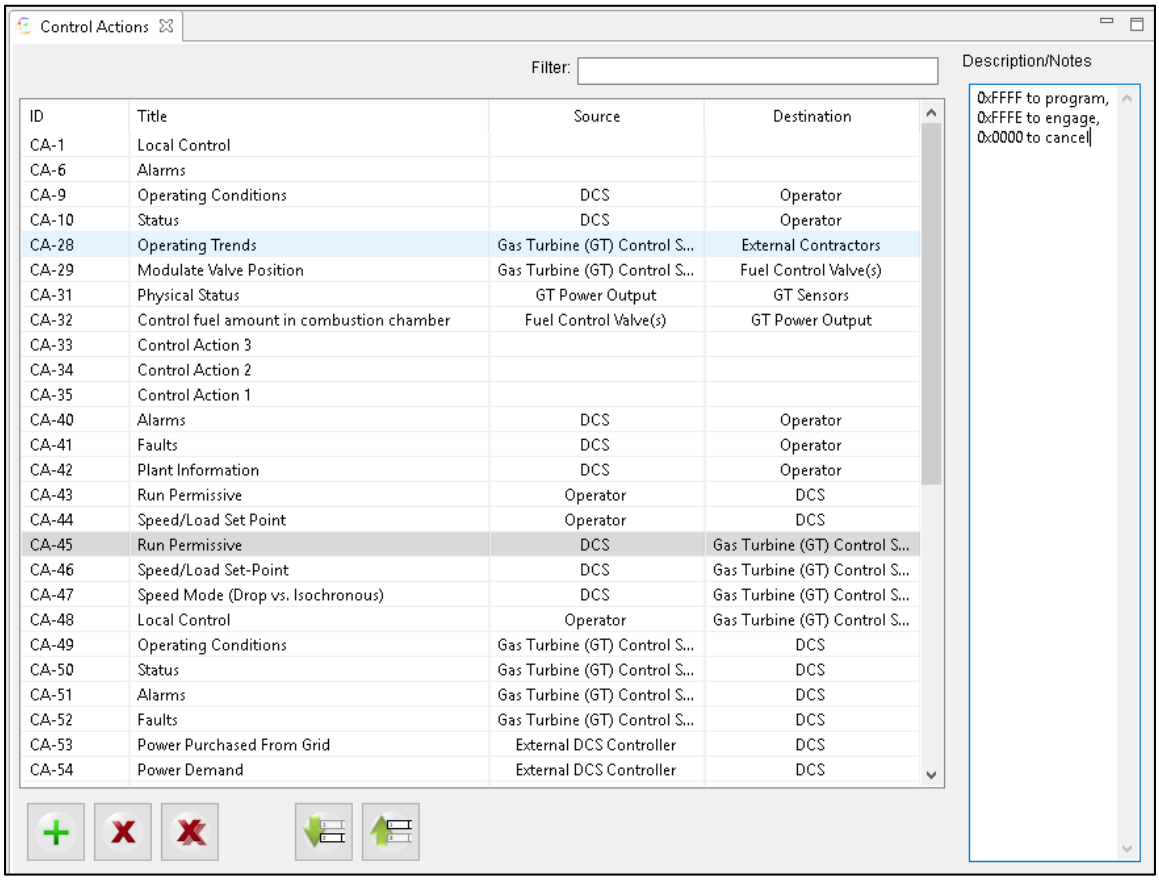
In conclusion of our exploration of control structures and process models, XSTAMPP leads with a free form drawing canvas implementation that includes graphical controls that correspond nicely to STAMP control structure objects (controllers, controlled processes, actuators, and sensors). Drawing features to mitigate connection fractures (see 3.2) can greatly drive drawing canvas usefulness when creating coherent control structure diagram(s) rapidly. STAMP Workbench leads in importing structured forms of control structures and their control action and feedback interactions, which is then used to generate control structures; a likely helpful feature if a crude control structure has been first created external to the tool, or perhaps first sketched manually by hand.

3.3 STPA-Sec Methodology Step 3: Identify Unsafe Control Actions (UCAs)

The next step in the analysis is to proceed through the control actions in an effort to *locate unsafe and insecure control actions*. The STPA Handbook provides the following definition:

"Definition: An Unsafe Control Action (UCA) is a control action that, in a particular context and worst-case environment, will lead to a hazard" [17].

XSTAMPP leads general purpose tools by allowing the functional control structure to specify and link control actions and feedbacks from and to controllers, actuators, controlled processes, and sensors. XSTAMPP generates a view of control actions once they are successfully added to the functional control structure; capturing structured information STAMP optimized ways can produce time saving collection of cybersafety analysis information for further software automation.



ID	Title	Source	Destination
CA-1	Local Control		
CA-6	Alarms		
CA-9	Operating Conditions	DCS	Operator
CA-10	Status	DCS	Operator
CA-28	Operating Trends	Gas Turbine (GT) Control S...	External Contractors
CA-29	Modulate Valve Position	Gas Turbine (GT) Control S...	Fuel Control Valve(s)
CA-31	Physical Status	GT Power Output	GT Sensors
CA-32	Control fuel amount in combustion chamber	Fuel Control Valve(s)	GT Power Output
CA-33	Control Action 3		
CA-34	Control Action 2		
CA-35	Control Action 1		
CA-40	Alarms	DCS	Operator
CA-41	Faults	DCS	Operator
CA-42	Plant Information	DCS	Operator
CA-43	Run Permissive	Operator	DCS
CA-44	Speed/Load Set Point	Operator	DCS
CA-45	Run Permissive	DCS	Gas Turbine (GT) Control S...
CA-46	Speed/Load Set-Point	DCS	Gas Turbine (GT) Control S...
CA-47	Speed Mode (Drop vs. Isochronous)	DCS	Gas Turbine (GT) Control S...
CA-48	Local Control	Operator	Gas Turbine (GT) Control S...
CA-49	Operating Conditions	Gas Turbine (GT) Control S...	DCS
CA-50	Status	Gas Turbine (GT) Control S...	DCS
CA-51	Alarms	Gas Turbine (GT) Control S...	DCS
CA-52	Faults	Gas Turbine (GT) Control S...	DCS
CA-53	Power Purchased From Grid	External DCS Controller	DCS
CA-54	Power Demand	External DCS Controller	DCS

Figure 30. XSTAMPP Control Actions Summary Table

3.3.1 STPA Handbook Hazardous Four Control Action Possibility Support

An effective process for identifying UCAs involves mapping the control actions against a set of standard four possibilities that may lead to losses when the system is in a hazardous vulnerable system state. The collection of hazardous process model variable values that create the system state conditions are specified in what is known as a "context table" (discussed further in Section 3.3.2). These four possibilities for when control actions may become unsafe are defined in the STPA Handbook, and are shown in Figure 31.

Control Action	Not providing causes hazard	Providing causes hazard	Too early, too late, out of order	Stopped too soon, applied too long

Figure 31. STPA's Hazardous Four Control Action Possibilities to Consider

XSTAMPP offers a helpful feature for these common hazardous four control manipulation possibilities by providing a matrix to evaluate all the control actions that have been captured until this point, as well as being able to link to previously captured hazards. Figure 32 shows a manually managed matrix using MS Office while Figure 33 shows XSTAMPP's control action matrix functionality. Research has shown that completing the evaluation of this table without specialized software tools is a significant, time-consuming process and software proof of concepts were built [11] [12]. STAMP-specific software tools are able to help facilitate this step much more so than a standard general purpose tool.

Action By	Control Action	Not Providing Causes Hazard	Providing Causes Hazard	Too soon, Too late, Out of order	Stopped too soon, Applied too long
GT Control System	Close Fuel Control Valve	**UCA-1:** GT Controller does not close *Fuel Control Valve* during startup or shutdown sequence (uncontrolled ignition) --> [H-2]	**UCA-4:** GT Controller closes *Fuel Control Valve* during *firing sequency* (no ignition) --> [H-2]	**UCA-6:** GT Controller close *Fuel Control Valve too late after* trip condition occurs (Surge, Loss of Flame, Overspeed, Over-temperature, vibration, combustion monitoring, Lube oil) --> [H-1; H-2]	**UCA-7:** GT Controller closes *Fuel Control Valve* for too long after achieving firing speed resulting in repeated purge cycles --> [H-2; H-4]
		UCA-2: GT Controller does not close *Fuel Control Valve* when trip condition occurs (Surge, Loss of Flame, Overspeed, Over-temperature, vibration, combustion monitoring, Lube oil) --> [H-1; H-2]	**UCA-5:** GT Controller closes *Fuel Control Valve* when power demand is higher than delivered by generator (generator breaker opens due to under-frequency protection) --> [H-4]	N/A	N/A
		UCA-3: GT Controller does not close *Fuel Control Valve* when power demand is low (generator breaker opens on over-frequency limit protection) --> [H-4]	N/A	N/A	N/A

Figure 32 MS Office Table Evaluating Manually Evaluating Control Actions

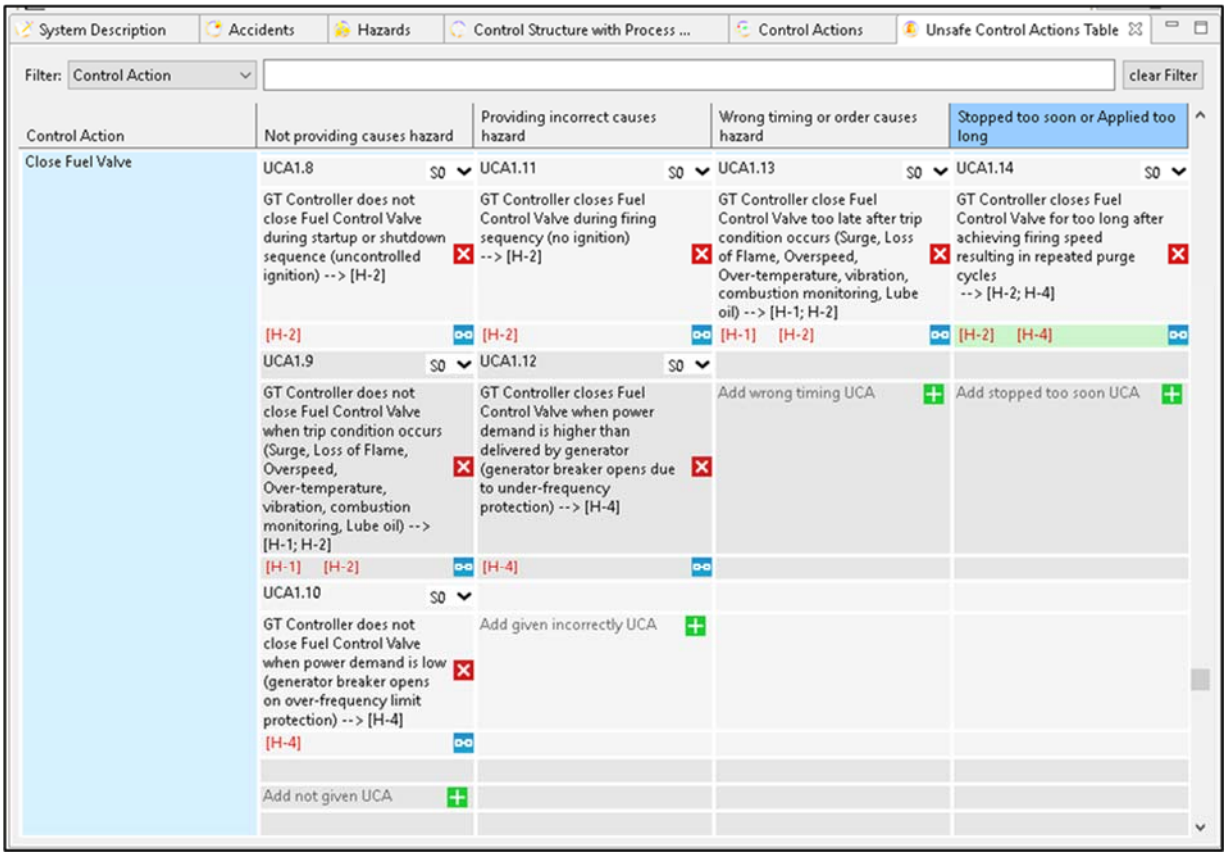


Figure 33. XSTAMPP Example Unsafe & Unsecure Control Action Matrix Evaluation Functionality

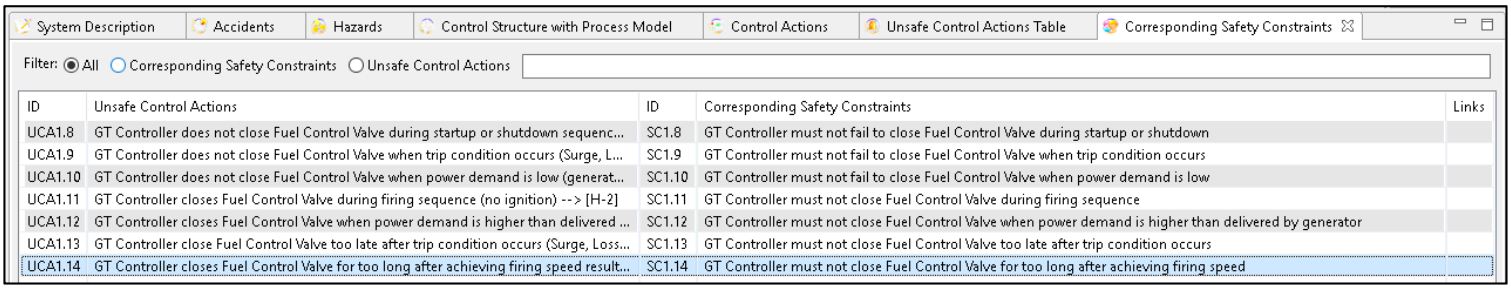
When control actions are imported from the control structure, software can produce these types of features to save time and cognitive effort on behalf of engineers and analysists performing systems theoretic cybersafety analysis. Logical simplification was proposed by John Thomas in an effort to eliminate duplicate rows in the matrix, and observations on the cognitive patterns people exhibit when filling out the matrix were highlighted [10]. Features that guide the user through an intuitive path through the matrix as they consider the hazardous possibilities would likely reduce the time required for this combinatorically challenging step of STAMP analysis. A more immersive view of the matrix, perhaps with another layer of UI depth such as pop-up or right click menus, may help to organize the massive amount of information in new ways.

Although it contains more information, and the ability to import and export the information to other software modules, the editing of UCAs in the XSTAMPP generated matrix is not as rapid as using a standard MS Office table. However, the UCA linking functionality realized by XSTAMPP is a promising improvement over standard general-purpose tables, and the ability to encapsulate dynamics links to hazards and control actions would likely gain analyst and engineer time back. Once the UCAs are identified, they must be organized and uniquely indexed. In a general-purpose tool, again, tedious manual indexing of the items in a traditional table does this, and consumes significant time as the number of UCAs scale. (See Figure 32).

Neither XSTAMPP nor general-purpose tools provide specific features for identifying and differentiating human and technical control actions. The distinct differences between human and

technologic behavior is extremely important to keep separate in the control structure and should be visually apparent, as they exhibit different capabilities and potential vulnerabilities.

From the UCAs, safety constraints are derived to prevent the vulnerable system states to be ever (including intentionally) achieved. XSTAMPP offers line-by-line editing of safety constraints on the list of UCAs derived from the matrix generation evaluation, although edits to UCAs require returning to the matrix-editing mode (see Figure 34). This can become more complicated than returning to a general purpose table, but the UCA matrix feature does offer the additional functionality of containing and working with much larger matrices than a standard table, and more potential to embed and encapsulate more information.



ID	Unsafe Control Actions	ID	Corresponding Safety Constraints	Links
UCA1.8	GT Controller does not close Fuel Control Valve during startup or shutdown sequenc...	SC1.8	GT Controller must not fail to close Fuel Control Valve during startup or shutdown	
UCA1.9	GT Controller does not close Fuel Control Valve when trip condition occurs (Surge, L...	SC1.9	GT Controller must not fail to close Fuel Control Valve when trip condition occurs	
UCA1.10	GT Controller does not close Fuel Control Valve when power demand is low (generat...	SC1.10	GT Controller must not fail to close Fuel Control Valve when power demand is low	
UCA1.11	GT Controller closes Fuel Control Valve during firing sequence (no ignition) --> [H-2]	SC1.11	GT Controller must not close Fuel Control Valve during firing sequence	
UCA1.12	GT Controller closes Fuel Control Valve when power demand is higher than delivered ...	SC1.12	GT Controller must not close Fuel Control Valve when power demand is higher than delivered by generator	
UCA1.13	GT Controller close Fuel Control Valve too late after trip condition occurs (Surge, Loss...	SC1.13	GT Controller must not close Fuel Control Valve too late after trip condition occurs	
UCA1.14	GT Controller closes Fuel Control Valve for too long after achieving firing speed result...	SC1.14	GT Controller must not close Fuel Control Valve for too long after achieving firing speed	

Figure 34. XSTAMPP Example Allowing New Security and Safety Constraints Based On Unsafe Control Actions

3.3.2 Hazardous System State Context Table Support

Included with the unsafe control action is the "context" that makes it unsafe. The context is a collection of process model variable values that put the system in an unsafe or insecure state, where then interactions with the surrounding external environment (including remote attackers) can cause losses when the unsafe control action(s) take place. This step has already been shown to have the potential to be improved with logic, natural language processing, and software tools [12]. In a general-purpose tool such as MS Office, this analysis is performed using a standard general purpose table as shown in Figure 35.

		Turbine Sequence	Turbine Speed	Exhaust Temperature	Flame Detected	Compressor Differential Pressure	Inlet Guide Vane Position	Vibration	Lubrication Oil Temperature/Pressure	Load Demand					
	System Variables	#1	#2	#3	#4	#6	#7	#8	#9	#10	Providing Causes Hazard	Not Providing Causes Hazard	Too Early, Too Late, or Out-of-Order	Applied too long, Stopped too soon	Hazards
Close Fuel Control Valve	CA-1	Start	-	-	-	-	-	-	-	-	0	1	0	0	H-2
	CA-2	Firing	-	-	-	-	-	-	-	-	1	0	0	1	H-2, H-4
	CA-3	S/Down	-	-	-	-	-	-	-	-	0	1	0	0	H-2
	CA-4	Oper.	Out	-	-	-	-	-	-	-	0	1	1	0	H-1, H-2, H-4
	CA-6	Oper.	-	Out	-	-	-	-	-	-	0	1	1	0	H-1, H-2, H-4
	CA-7	-	-	-	No	-	-	-	-	-	0	1	1	0	H-1, H-2, H-4
	CA-8	-	-	-	-	Out	-	-	-	-	0	1	1	0	H-1, H-2, H-4
	CA-9	Oper.	-	In	-	-	Close	-	-	-	0	1	1	0	H-1, H-2, H-4
	CA-10	-	-	-	-	-	-	Out	-	-	0	1	1	0	H-1, H-2, H-4
	CA-11	-	-	-	-	-	-	-	Out	-	0	1	1	0	H-1, H-2, H-4
	CA-12	Oper.	-	-	-	-	-	-	-	Pos	1	0	0	0	H-1, H-2, H-4
	CA-13	Oper.	-	-	-	-	-	-	-	Neg	0	1	0	0	H-1, H-2, H-4

Figure 35. MS Office Table Evaluating and Linking Control Actions with Hazardous Process Model Variable Values (Context)

In XSTAMPP, there is a context table manager with various features related to the examination of the combinations of process model variable values (system states) when considering control actions that could result in damages if they are tampered with at certain times. The first view lists all control actions, with the option to flag them as security critical. Figure 36 shows this additional helpful functionality to highlight systems theoretic cybersafety related unsafe control actions.

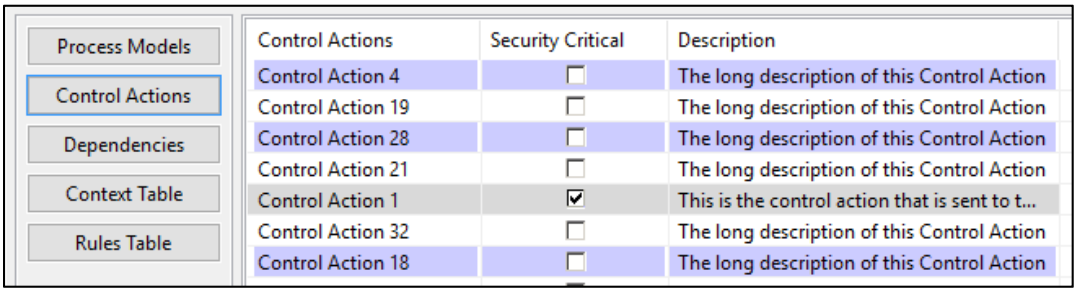


Figure 36. XSTAMPP Context Table Manger: Security Critical Control Action Review

The next set of context table specific features XSTAMPP offers have to do with the management of relevant process model variables that contribute to the control action becoming unsafe. This is where STAMP-specific tools take a lead over manual general purpose tools such as MS Office in terms of productivity, which can't offer such optimized linking of information with general purpose features. When the system is in such a vulnerable state, control actions can then be considered unsafe or insecure if the system encounters environmental conditions that cause a hazard. In the case of cyberattacks, the environment is considered potentially malicious rather than simply providing external interactions and disturbances to the system, and when the system has process model variables values that put the system in a hazardous state, environment interactions can become disastrous. Combinations of vulnerable process model variable values can be specified in a dynamic interface after the relevant process model variable values are listed as dependencies for the control action under consideration.

3.1.1 NIST's ACTS for Optimized T-Way Interaction Context Table Generation

Context table generation features are supported by XSTAMPP in a way that general purpose tools simply cannot compete with. Context tables can be generated with ACTS, NIST Computer Security Resource Center (CSRC) government website's "Automated Combinatorial Testing for Software," which supports multiple t-way combinatorial test set generation algorithms to produce an optimized number of interaction combinations in the system likely to uncover unexpected interactions [24] [37] [38]. This type of optimized system test set generation comes from NIST software testing theory [37].

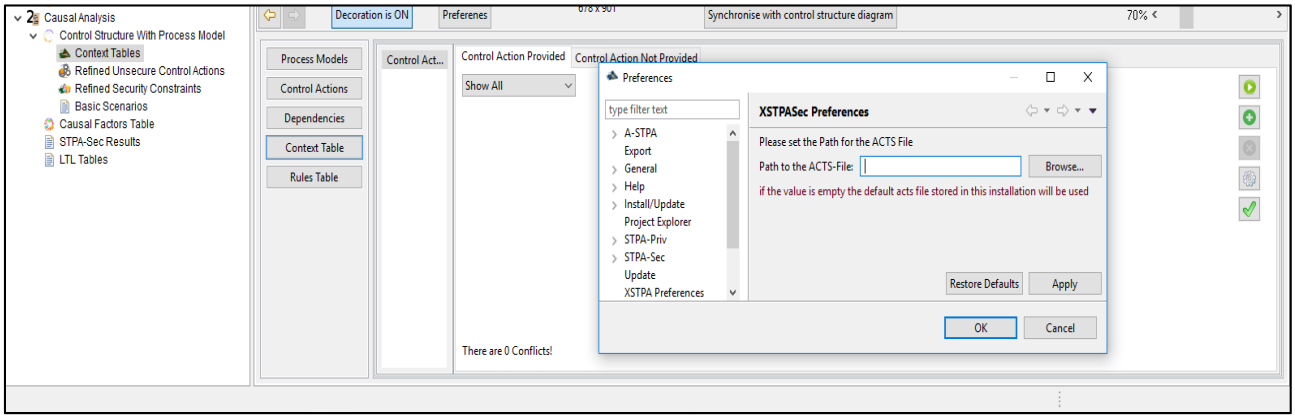


Figure 37. XSTAMPP Context Table Generation: NIST CSRC's ACTS Tool Interface

The tool runs on a generic Java interface, and offers both command line and graphical user interface options. XSTAMPP currently supports ACTS configuration file input to intelligently control, direct, and optimize the generation of context tables to consider. NIST's Automated Combinatorial Testing for Software (ACTS) is a "generation tool for constructing t-way combinatorial test tests," and is commonly used to efficiently create exhaustive software tests that are very effective at detecting faults that come from "unexpected interactions between different contributing factors" [37]. This is to support efficient creation of system tests that require combinatorial and pairwise testing, sometimes for regulatory compliance (citation pending). By levering some of the features of the generation engine such as constraint rules, the ACTS framework can be utilized to manage large-scale context table variable combinations by pruning invalid process model variable combinations from further consideration using constraint rules specified in restricted first-order logic expressions. Another feature allows the generation

of combinations given subsets of variables that the analyst is interested in exploring; the ACTS framework supports this notion of "related" variables (parameters), which can be used to select subsets of process model variable values, allowing targeted generation of combinations involving a respective set of variables and their respective values. For example, if an analyst suspected a variable frequency drive may be attacked, they may group a variable frequency drive speed variable with many other process model variables using a numerical relationship strength to limit the generation of context tables to combinations that involve the variable frequency drive speed variable.

As the control structure diagrams scale to support larger systems, the number of variable value combinations currently results in time-consuming cognitive and manual tasks, as noted by S. Khan during his systems theoretic cybersafety analysis of MIT's CUP Cogen Facility. The ACTS framework generation engine will likely aid in enforcing constraints to allow the analyst to prevent themselves from being distracted from considering invalid combinations of process model variable values. The ACTS framework can also allow the analyst the benefits of focusing solely on interesting combinations of variables using the relationship feature's subset grouping functionality. Examples of constraints illustrated in the ACTS User Guide are shown below in Table 2 [Page 9, 24].

Constraint 1: (OS = "Windows") => (Browser = "IE" || Browser = "FireFox" || Browser = "Netscape"), where OS and Browser are two parameters of type Enum.	This constraint specifies that if OS is Windows, then Browser has to be IE, FireFox,or Netscape.
Constraint 2: (P1 > 100) || (P2 > 100), where P1 and P2 are two parameters of type Number or Range.	This constraint specifies that P1 or P2 must be greater than 100.
Constraint 3: (P1 > P2) => (P3 > P4), where P1, P2, P3, and P4 are parameters of type Number or Range.	This constraint specifies that if P1 is greater than P2, then P3 must be greater than P4.
Constraint 4: (P1 = true || P2 >= 100) => (P3 = "ABC"), where P1 is a Boolean parameter, P2 is a parameter of type Number or Ranger, and P3 is of type Enum.	This constraint specifies that if P1 is true and P2 is greater than or equal to 100, then P3 must be "ABC".

Table 2: ACTS Constraint Logic Examples [Page 9, 37]

ACTS supports multiple generation algorithms for creating the combinations of variable values, but two offer support for "constraint" expressions using first order propositional logic: IPOG and IPOG-F, which generate variable (parameter) value combinations for "systems of moderate size (less than 20 parameters and 10 values per parameter on average)" [24]. If the system can be abstracted to less than 20 process model variables, with up to 10 values per process model variable, first order propositional logic rules can be applied to eliminate undesirable combinations. The "Forbidden Tuples" constraint option is recommended for larger numbers of variables and values with many constraints, and relationship strengths greater than or equal to 2, while "CSP Solver" constraint option is faster for small parameter numbers with few constraints, and relationship strengths less than or equal to 2. Evaluating the specified constraints and relationships could likely produce a best-guess default recommendation from the software tool automatically. A third algorithm called IPOG-D supports "larger systems" and appears to scale with larger combinations of process model variable values, but support for constraints using first order propositional logic is not currently supported, but is "planned in a future release" [24].

An optimal process model variable value combination generation tool would aim to support larger systems as well as logical expression based constraints, as well as relationships and grouping to govern and target variable value combination generation to where the analyst chooses to look. The format of the ACTS configuration is shown in the tool's documentation and shows the current syntax for defining constraints and relationships using ACTS to run the various generation algorithms and generate customized, optimized variable value combination sets [24].

The generated variable combinations can be exported into STAMP tools like XSTAMPP to support specific features that provide an efficient navigation and analysis mechanism for the oftentimes numerous context tables generated from an exhaustive consideration of all combinations of the process model variables. Features like this are game-changers for STAMP tools and have the potential to enable large time savings (when leveraged effectively) compared to STPA-Sec analysis using traditional general purpose editors such as MS Office or Draw.IO. By eliminating invalid combinations of process model variable values, considering sets of values that at minimum cover t-way interactions, as well as explicitly listing groups of variables of interest, STAMP software can enable the analyst to consider a more targeted set of context tables and process model variable value combinations instead of trying to exhaustively manage the combinations manually. These types of context table navigation features allow more efficient and flexible use of analyst time when considering what could potentially contribute to factors that may place the system in an unsafe or insecure state where losses could occur, and help mitigate the burden of having to exhaustively consider (sometimes extremely numerous) combinations of process model variable values.

3.4 STPA-Sec Methodology Step 4: Identify Scenarios

The last step of STPA-Sec analysis is to *identify scenarios* that could lead to the previously determined unsafe and insecure control actions, from a perspective theorizing a possible attack. The STPA Handbook provides the following definition:

"Definition: A loss scenario describes the causal factors that can lead to the unsafe control actions and to hazards" [17].

The process and work involved in this step is currently left almost completely to the analyst, as they work backwards from UCAs in an effort to theorize how someone may attack the system, in an intentional effort to cause losses.

The inputs and outputs for the STPA-Sec Step 4 are shown in Figure 26. Two main categories of strategy for forming loss scenarios are described in Figure 38 below.

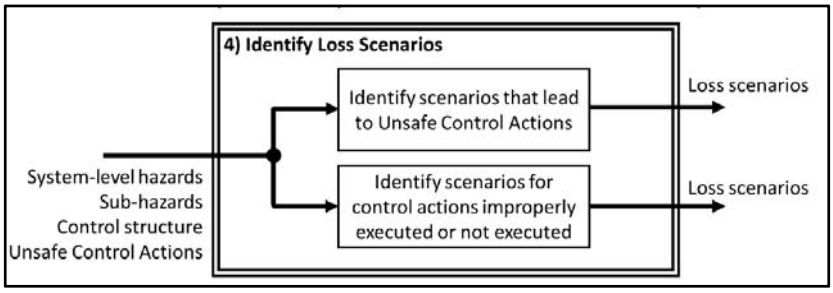


Figure 38. [17, Figure 2.20]. STPA Handbook: Overview of Scenario Identification

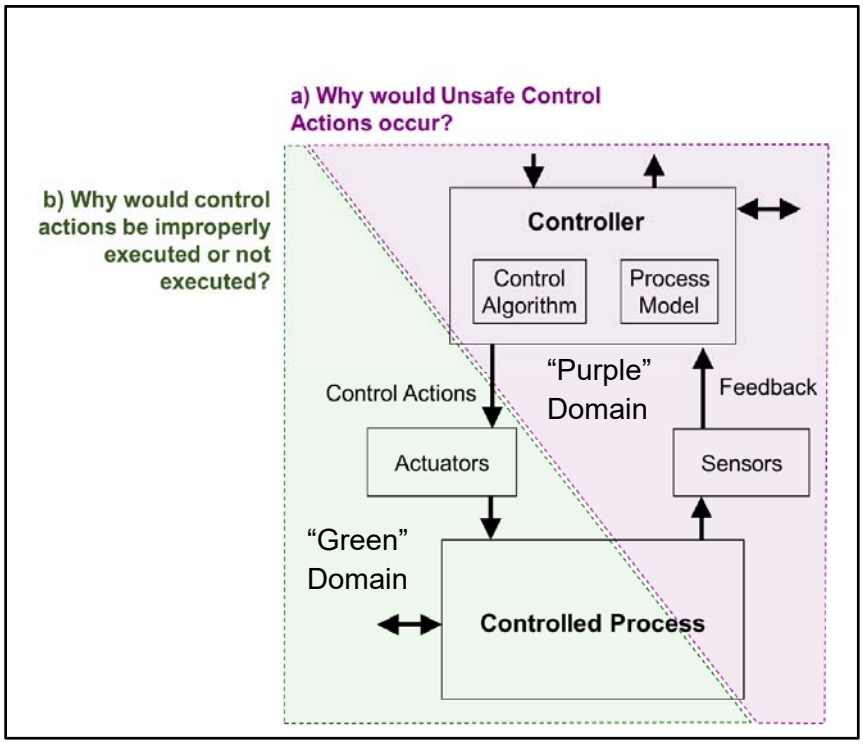


Figure 39. [17, Figure 2.17]. STPA Handbook: Two Types of Scenarios That Must Be Considered

STPA-Sec Step 4.1 consists of *enumerating loss scenarios that lead to unsafe control actions,* shown above in Figure 39 when focusing in the purple region of the control structure. Figure 40 below shows two main causes of unsafe control actions that can help guide the analyst's focus and creativity during this step.

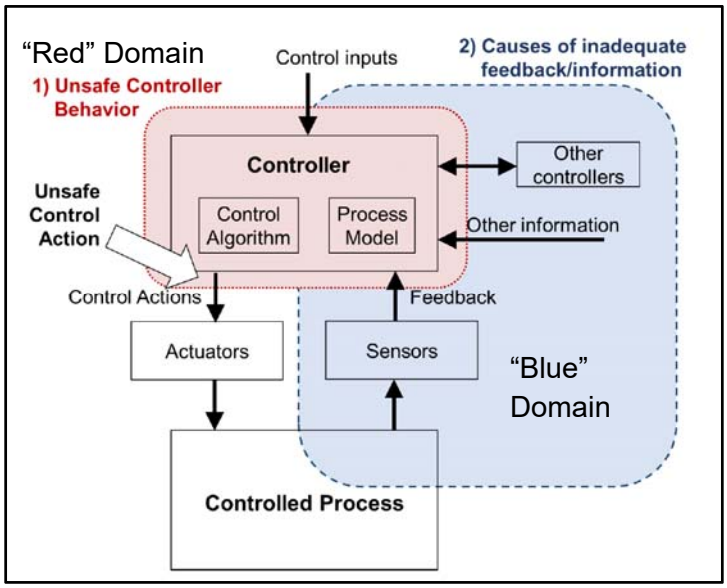


Figure 40. [17, Figure 2.18]. STPA Handbook: Two Causes of UCAs Scenarios

STPA-Sec Step 4.2 consists of *enumerating loss scenarios that result from control actions being improperly executed or not executed,* shown above in Figure 39 when focusing in the green region of the control structure. In addition to considering UCAs, the STPA Handbook suggests it is helpful to focus on the control path and other factors related to the controlled process, with example regions of each shown below in Figure 41 in red and blue, respectively.

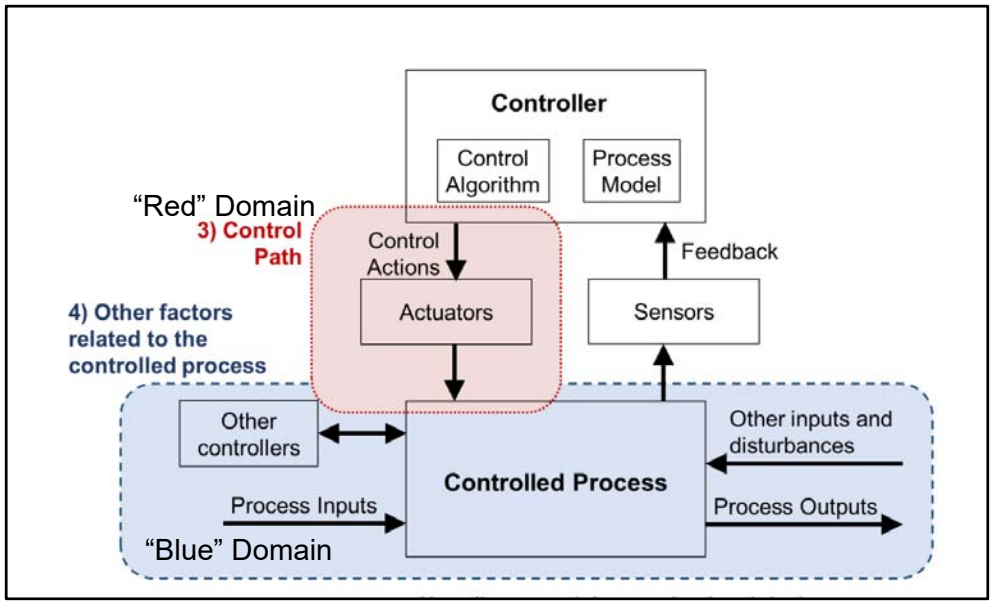


Figure 41. [17, Figure 2.19]. STPA Handbook's Generic Control Loop, Control Path, and Other Controlled Process Factors

Both of these sub-steps involved in the creation of STPA-Sec loss scenarios require cognitive creativity on the part of the analyst to identify the loss scenarios and both general purpose tools as well as XSTAMPP have a significant challenge supporting these open-ended steps. The STPA practice analysis performed in the SDM core project team used a brainstorming process of hypothesizing together to arrive at scenarios through games of "what if" postulates. It is assumed it would require significant application of synthetic knowledge organization for a tool to suggest somewhat helpful scenarios to consider based on the surrounding environment and external system inputs in a very natural way.

XSTAMPP takes the approach of providing the analyst with an organized view of as much of the important factors to help the analyst leverage their creativity to quickly identify loss (and potential attack) scenarios. XSTAMPP provides a matrix view where components can be linked with UCAs and vulnerabilities, scenarios can be described, and corresponding security constraints can be entered along with notes. The XSTAMPP matrix-based features for organizing large amounts of UCAs, vulnerabilities, and security constraints around scenarios provides a level of organization for the scenario creation process not natively found in general purpose editor tools such as MS Office, but is awkwardly implemented. The ability to focus on components, UCAs, and vulnerabilities in a single view shows great promise for providing a rich thought environment to seed creative combinations of external and environmental effects. This enables the identification of scenarios that could lead to losses, and appears to help when comparing with starting from a general purpose document with information dispersed such as in MS Office. Figure 30 shows XSTAMPPs causal factor scenario matrix-based management features.

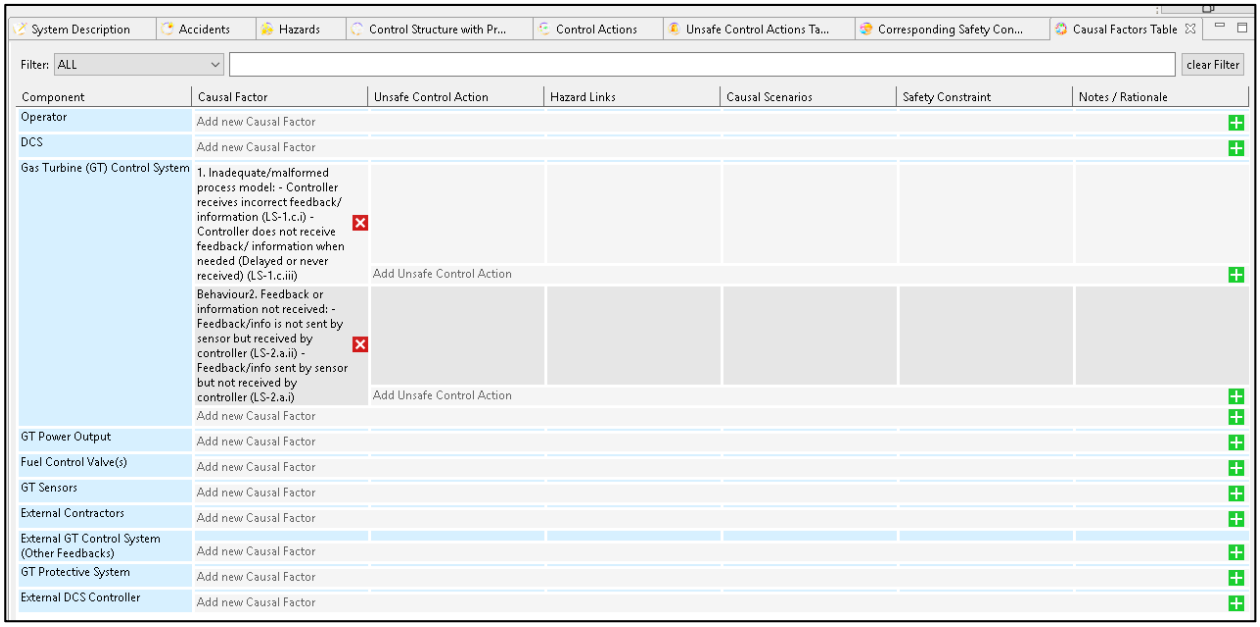


Figure 42. XSTAMPP Example Matrix Based Scenario Management

In contrast, generic drawing tools allow for a less structured, perhaps more artistic organization of the information involved in each scenario, at the expense of more analyst time. Figure 43 shows an example of creating a MS Office based visualization of a scenario that could lead to loss in the example analysis.

UCA-2: GT Controller does not close *Fuel Control Valve* when trip condition occurs (Surge, Loss of Flame, Overspeed, Over-temperature, vibration,				
	Loss Scenario	**Associated Causal Factors**	**Rationale**	
2.0	During a controlled (fired) shutdown of the gas turbine, the *shutdown* sequence is modified such that after the flame is extinguished, the Fuel Control Valve instead of being closed shut, is opened. A flammable mixture of gas accumulates resulting in an internal explosion.	1. Inadequate/malformed control algorithm: - Flawed implementation of the specified control algorithm (LS-1.b.i)	-Malicious command manipulation on GT controller (N-1) modifies the controllers auto shutdown sequence to undertake incorrect actions i.e. open fuel control valve when the opposite is required	Unsafe Controller Behaviour
		2. Inadequate process model: - Protection system (controller) receives correct feedback (i.e. loss of flame) but ignores it or interprets it incorrectly (LS-1.c.ii)	-The protection system is programmed to issue a trip command for flameout condition during operation. However, during *shutdown sequence, loss of flame* condition is expected and hence the protection system considers this as routine occurance	
2.1	During operation, the GT controller incorrectly modulates the inlet guide vanes resulting in excessive exhaust temperatures. Despite the existence of the trip conditions, the Protection System does not intervene to shut off the fuel control valve.	1. Inadequate/malformed process model: - GT Controller receives correct feedback/ information but interprets it incorrectly or ignores it (LS-1.c.ii), thus incorrectly modulating the inlet guide vanes	-Malicious feedback manipulation at controller (N-1) from sensors (C-4) causes the controller to assume incorrect state and hence incorrectly modulate inlet guide vanes	Unsafe Controller Behaviour
			-The protection system controller is also simultaneously compromised; 'trip' values are maliciously modified to prevent safety-critical control action when trip conditions occur. This is possible because the specific safety controller used in this application allows reprogramming of ROM values	

Figure 43. MS Office Example General Purpose Tool Visual Scenario Diagram Example

In general purpose tools, rich text is available to document the scenario in a wide array of custom formats. Using existing software features, scenario generation could potentially be expedited through the creation of thoughtful and robust template documents (such as the format in Figure 43) to capture structured scenario information such as attack descriptions, causal factors, and rationale notes.

3.5 Final Steps: Outputs, Reports, and Traceability

The final steps of systems theoretic cybersafety analysis, and potentially the most important and influential for implementing security changes, is a tool's ability to compile the analysis information into a manageable, organized, and neat report for information dissemination and discussion. XSTAMPP aims to have a simple export feature where a final PDF report is created. It is highly likely STPA-specific tools

like XSTAMPP, with a more structured input and organization of STAMP analysis data, are more easily able to produce standardized and more compact reports compared with general-purpose tools such as MS Office and Draw.IO. General-purpose templates could guide the creation of critical aspects of the report and may increase analyst efficiency, but it is likely that specialized report generation capabilities from STAMP specific tools would have significant advantage when compiling the final analysis report.

4 Conclusions

Software has been helping humans visualize, organize, and navigate information for as long as computing machines have existed. It is believed there is opportunity for new software features to reduce the manual workloads placed on engineers or analysts who want to perform systems theoretic cybersafety analysis, especially to support the rapid analysis of OT systems found in energy distribution systems in response to recent destructive cyberattacks on similar cyber-physical energy distribution systems [21].

Various opportunities exist to utilize existing STAMP tool features for supporting systems theoretic cybersafety analysis; Section 3 documents the evaluation of some leading STAMP tool features as well as existing general purpose document creation features to demonstrate using software to assist in an example analysis of energy distribution systems. Along the way, an introduction to STAMP analysis steps was included to provide background on the systems theoretic methodologies that can be applied to cybersafety.

Tools that aim to support systems theoretic cybersafety should be evaluated using general software attributes covering general software features such as architecture, auto-save and auto-recovery, standard drawing canvas features, undo and redo, etc. (see Appendix B – STAMP Software Tool General Attribute). Future systems theoretic cybersafety tools should be designed with a detailed list of feature areas in mind (see Appendix A – Detailed Categories of Software Features Supporting STAMP Analyst Items). Software tools aimed at supporting STAMP methodologies should also match the 2018 STPA Handbook terminology and structure to facilitate standardizing training program development and educational sustainability.

Leading STAMP software features have already been realized by software, especially in the area of linking various pieces of information in the analysis. Linking allows the tool to understand the structure of the information, and navigate it, which is required to enable further automation features. XSTAMPP offers some features that bring together information in rich, visual ways that can help analysts view the relevant information when considering attack scenarios. We believe finding more visual organizations of the data will continue to enable the engineer or analyst to spend their time efficiently when performing systems theoretic cybersafety analysis. The hierarchical view and guidance through the method's progression can help analyst learn quicker, and is powerful for navigating through the analysis information quickly. Finally, XSTAMPP and STAMP Workbench run on the Eclipse platform, which offer the benefits of being cross platform, free, and open source, all noted positives for software tools. Both projects are considerable candidates for implementing new features.

The context table management was observed to be a large source of tedious edits and manipulations when performing alterations in MS Office both by an MIT SDM class team as well as the MIT Cogen Analysis [18] [S. Khan working paper]. Fundamentally the analysis process aims to have the analyst or engineer mentally enumerate all the state combinations and consider if they leave the state in a hazardous/vulnerable state. The context table is a collection of hazardous process model and environment values that lead to a vulnerable system state, and XSTAMPP claims to offer it can be auto-generated with

ACTS-based automation optimizations to save human time [24]. The context feature set is forming but is still considered somewhat in-development at this time.

In existing functional control structure diagrams, controllers, sensors, and actuators currently do not have software features that support showing information about external I/O. Sensors do not currently have software features that support showing information about input, or feedback output. Actuators do not currently have software features that support showing information about control input, or output. The ability to refer to a physical control structure allows analysts another view of the system, and to allocate implementations from the functional control structure, which is currently not supported by software features [23]. When linked with the functional control structure, the two control structure views would likely aid in the faster navigation of the system in the pursuit of uncovering vulnerabilities, unsafe control actions, and attack scenarios. Multiple control structures were observed to be necessary in practice to capture the larger context by S. Khan's MIT CUP Cogen cybersafety analysis experience, but is not currently supported in XSTAMPP.

A final set of missing features for supporting systems theoretic analysis on industrial control systems are related to supporting a physical control structure. Formalizing the creation of a physical component layer editor to show network devices, networks, and network connections would provide foundational physical control structure objects for industrial control systems that use networks to communicate aimed at cybersecurity and cyber safety [23]. Software features to integrate the possibilities of general network integrity threats, highlight known vulnerabilities from databases, and integrating STPA-SafeSec's general availability threats could further formalize and expand the process of evaluating control actions by considering the form that the control functionality maps to [23].

Additional initial future work and research that remain after this examination are as follows:

- How can standardization of STPA-Sec data allow interfacing with other software tools and processes in a systems engineering toolchain?

- How could a client-server software model provide additional computational resources, security, as well as portability and availability for STPA-Sec software tools?

- Where can additional logical structure or propositional logic be formalized in STPA-Sec to enable software automation?

- How could hardware accelerated graphical control structure design features enable more rapid and detailed control structure design? What new organization, visualizations, navigation, and automation options would next-generation drawing and simulation software features enable?

- What would a more formalized subsystem mechanism enabling a notion of zooming, abstracting, or aggregation in hierarchical control structures look like?

- Would further integration of process model and process model variables into the control structure drawing features allow for more organization of the hazardous/vulnerable system states? In addition, how could one view the information while preserving diagrammatic simplicity?

- Would the addition of features to support the management and potentially simulation of a physical control structure aid in the discovery of unsafe control actions and attack scenarios?

Longer term future research areas:

- What generation or automation opportunity would be enabled through specifying systems theoretic information through an optimized structured input mechanism using a domain specific language to specify data structures found in Appendix C – Preliminary Systems Theoretic Cybersafety Data Structure Organization

- Could artificial intelligence assist in software automation of the capture of data structures found in Appendix C – Preliminary Systems Theoretic Cybersafety Data Structure Organization?

Appendix A – Detailed Categories of Software Features Supporting STAMP Analyst Items

STPA HANDBOOK STEP REFERENCE (And Local Reference)	Areas of Software Feature Requirements Supporting Systems Theoretic Cybersecurity and Cybersafety
STEP 1 (3.1)	**STPA-Sec Methodology Step 1.1: Define the Purpose of the Analysis**
(3.1)	Support identifying rich system purpose statement
(3.1)	Support identifying rich system description
(3.1)	Support identifying rich system goals
(3.1.1)	**STPA-Sec Methodology Step 1.2: Identify Losses**
(3.1.1)	Support unique identifiers on losses
(3.1.2)	**STPA-Sec Methodology Step 1.3: Identify System-Level Hazards (Vulnerabilities)**
(3.1.2)	Support unique identifiers on hazards
(3.1.3)	**STPA-Sec Methodology Step 1.4: Identify System-Level Constraints**
(3.1.3)	Support refining hazards (optional)
STEP 2 (3.2)	**STPA-Sec Methodology Step 2: Model the Control Structures**
(3.2)	Support unique identifiers for all control structure objects
(3.2)	Support reference label identifiers for all control structure objects [23]
(3.2)	Support subsystem identification
(3.2)	Support multiple levels of abstraction and multiple control structure views
(3.2)	Support environmental inputs
(3.2)	Support written/trained procedures & other training inputs
(3.2)	Support external controllers
(3.2)	Support human controller model of environment
(3.2)	Support human controller model of automation
(3.2)	Support human controller model of controlled process
(3.2)	Support human controller model of other controllers
(3.2)	Support human controller control action generation / mental processing
(3.2)	Support controllers (general)
(3.2)	Support controller control and feedback [23]
(3.2)	Support sensors (general)
(3.2)	Support sensor input and feedback output [23]

(3.2)	Support actuators
(3.2)	Support actuators output and control input [23]
(3.2)	Support controller, sensors, and actuators external I/O [23]
(3.2)	Support sensor displays, alarms, and other sensory feedback
(3.2)	Support automated controller control algorithm
(3.2)	Support automated controller model of human controller
(3.2)	Support automated controller model of other controllers
(3.2)	Support automated controller operational mode
(3.2)	Support automated controller model of controlled process
(3.2)	Support other controllers and other systems (external to control structure system boundary)
(3.2)	Support process model creation and encapsulation inside controller
(3.2)	Support process model variables
(3.2)	Support process model variable values (PMVVs)
(3.2)	Support process model variable types
(3.2)	Support assigning responsibilities to functional control structure (FCS)
(3.2)	Support controlled process creation
(3.2)	Support controlled process inputs and outputs
(3.2)	Support controlled process disturbances
(3.2)	Support controlled process operational mode
(3.2)	Support control action creation
(3.2)	Support differentiation between technical and human control actions
(3.2)	Support feedback creation
(3.2)	Support hierarchical functional control structure
(3.2)	Support iterative control structure expansion and refinement
(3.2)	Support functional relationships and functional interactions
(3.2)	Support physical processes creation [23]
(3.2)	Support physical component structure mapping to functional control structure(s)
(3.2)	Support physical process inputs and outputs (state) [23]
(3.2)	Support multiple levels of functional control structure hierarchy (abstraction levels)
(3.2)	Support documenting additional information about controllers
(3.2)	Support documenting additional information about control actions and feedbacks
(3.2)	Support documenting additional information about controlled processes

(3.2)	Support common physical control structure network (devices, network connections, networks, network devices, and standard network integrity/availability threats [23])
(3.2)	Support control action unique identifiers
(3.2)	Support control action organized appearance on functional control structure (FCS)
(3.2)	Support control action expand and collapse functionality to save space on FCS
(3.2)	Support documenting additional information about all control structure objects (both physical & functional)
STEP 3 (3.3)	STPA-Sec Methodology Step 3: Identify Unsafe Control Actions (UCAs)
(3.3)	Support STPA Handbook Four UCA Type Matrix Generation [10, Page 38, 80-87]
(3.3)	Support linking UCAs to hazards
(3.3)	Support identifying human UCAs
(3.3)	Support identifying technical UCAs
(3.3)	Support differentiating between human and technical UCAs
(3.3)	Support defining controller constraints
(3.3)	**Support evaluating UCAs by providing CA** [10, Page 38, 80-87]
(3.3)	Support control source providing control action ID
(3.3)	Support context for control action (with vulnerable PMVVs)
(3.3)	**Support evaluating UCAs by not providing CA** [10, Page 38, 80-87]
(3.3)	Support control source providing control action ID
(3.3)	Support context for control action (with vulnerable PMVVs)
(3.3)	**Support evaluating UCAs by considering providing potentially safe CA too early, too late, or in wrong order** [10, Page 38, 80-87]
(3.3)	Support control source providing control action ID
(3.3)	Support context for control action (with vulnerable PMVVs)
(3.3)	**Support evaluating UCAs by considering if CA lasts too long or is stopped to soon** [10, Page 38, 80-87]
(3.3)	Support control source providing control action ID
(3.3)	Support context for unsafe control action (with vulnerable PMVs / PMVVs)
STEP 4 (3.4)	STPA-Sec Methodology Step 4: Identify Scenarios
(3.4)	Support hazardous control actions information flow identification
(3.4)	Support unsafe CA scenarios ID (using NL STPA Step 2 safety control flaw analysis)
(3.4)	Support unsecure CA scenarios ID (using Young's expanded control flaw analysis)
(3.4) STPA Handbook a) “Purple Domain”	**Support scenarios by identifying why UCA(s) occur & scenarios that lead to UCA(s)**

[17, Page 52, Figure 2.17 & Figure 2.20]	
(3.4)	Support scenarios by identifying UCAs and working backwards
(3.4)	Support scenarios by identifying UCAs from physical failures involving controller
(3.4)	Support scenarios by identifying UCAs from inadequate control algorithm
(3.4)	Support scenarios by identifying UCAs from unsafe or insecure control input from another controller
(3.4)	Support scenarios by identifying UCAs from inadequate process model
(3.4) STPA Handbook 1) “Red” Domain [17, Page 44, Figure 2.18]	**Support scenarios by identifying UCAs from unsafe controller behavior**
(3.4) STPA Handbook 2) “Blue” Domain [17, Page 44, Figure 2.18]	**Support scenarios by identifying UCAs from causes of inadequate feedback and information**
(3.4)	Support scenarios by identifying UCAs from feedback or information not received
(3.4)	Support scenarios by identifying UCAs from inadequate feedback received
(3.4) STPA Handbook b.) “Green Domain” [17, Page 52, Figure 2.17 and Figure 2.20]	**Support scenarios by identifying why control actions (CAs) would be improperly executed or not executed**
(3.4) STPA Handbook 3) “Red” Domain [17, Page 49, Figure 2.19]	**Support functional control structure control path identification and linking to the control path**
(3.4)	Support scenarios by identifying from control action not executed
(3.4)	Support scenarios by identifying from control action improperly executed
(3.4) STPA Handbook 4) “Blue” Domain [17, Page 49, Figure 2.19]	**Support scenarios by identifying other factors related to the controlled process**
FINAL STEPS	**OUTPUTS**
	Report Generation
	Support existing system architecture design feature identification and alteration identification
	Support new executable requirements creation

	Support identification of design requirements
	Support identification of necessary mitigations and safeguards
	Support defining test cases and creation of test plans
	Support new design decisions (during development)
	Support evaluation of existing design decisions for gaps and changes (after development)
	Support development of leading indicators of risk
	Support designing more effective safety management systems
	Support generation of organized views of STAMP analysis information
	Support traceability through intelligent automated information linking [17, Page 52, Figure 2.21]

Appendix B – STAMP Software Tool General Attribute Framework

Software Tool General Attributes		**XSTAMPP**
General Software Attribute		**State Explanation**
Safety Standard Qualifications [7]		
	IEC 61508 - E/E/PE	Yes. Tool T2 category with generation of any test cases, T1 otherwise. (pending vendor confirmation)
	EN50128 - Railway	Yes. Tool T2 category with generation of any test cases, T1 otherwise. (pending vendor confirmation)
	D0-178C/D0-330 - Aerospace & Defense	Yes. Can only be classified as TQL-5 (likely software Level B-D) by an end user, other levels of support require developer involvement providing qualification data. (pending vendor confirmation)
	ISO26262 - Automotive	Yes. Tool Impact Level 2 (TI2). Tool Confidence Level 3 (TCL3). Possibly TCL1-2 if LSA was implemented. Presentation shows ISO26262 limitations for AVs, compliments to (and differences from) HARA, details in [15]. (pending vendor confirmation)
Open Source Compliance (from opensource.org)		
	Apache License 2.0	No
	BSD 3-Clause "New" or "Revised" license	No
	BSD 2-Clause "Simplified" or "FreeBSD" license	No
	GNU General Public License (GPL)	No
	GNU Library or "Lesser" General Public License (LGPL)	No
	MIT license	No
	Mozilla Public License 2.0	No
	Common Development and Distribution License	No
	Eclipse Public License	Yes. "XSTAMPP is published under the Eclipse public license. All software is provided free of charge and will remain free in the future."
General Availability		
	Immediate	Yes. Easily downloadable.
	On Request	N/A
	Other (Including Future / Planned / In Development)	N/A
Cross Platform Compatibility (Runtime required)		
	Linux (PC)	Yes. (Java 8 JRE)
	Apple Mac OS (PC)	Yes. (Java 8 JRE)
	Microsoft Windows OS x64 (PC)	Yes. (Java 8 JRE)
	Apple iOS (Mobile)	No. Not at this time.
	Android OS (Mobile)	No. Not at this time.
Device Compatibility		
	Desktop PC Workstation Compatible	Yes.
	Mobile Laptop Workstation Compatible	Yes.

	Mobile Tablet Compatible	Depends; Not natively on mobile OS at this time, but possibly functional with remote desktop connection.
	Mobile Smartphone Compatible	No. Not at this time.
Architecture		
	Standalone Application	Yes. Wrapper of Eclipse Platform.
	OnPrem Client-Server	Partially. App is standalone but shared storage could provide shared workspace and collaboration. No real time collaboration functionality.
	3rd Party Cloud-Based	No. Not at this time.
Dependencies		
	Interfaces with and requires larger software platform product(s)	Yes. Requires Eclipse Platform.
	Interfaces with and requires larger software runtime(s)	Yes. Requires Java 8+ Runtime Environment (JRE 8+)
	Standalone or all-inclusive product	Somewhat. Eclipse Platform included, which offers to help install JRE as well.
System Versioning		
	Numerical	Yes. Currently Version 2.5 released Sep 29, 2017
	Version Control System Based	Yes. Freely and immediately available at https://github.com/asimabdulkhaleq/XSTAMPP.git
Content Versioning		
	Numerical	Yes. Manual version management.
	Version Control System Based	No. Supports static saving of project and no integrated versioning control system.
Reliability		
	Automatic Save	No. Not at this time.
	Automatic Project Recovery (in the event of a crash)	No. Not at this time.
	Automatic Crash Reporting	No. Not at this time.
	Maintains Backwards Compatibility	Partially. Requires JRE8+. Unclear if previous Workspaces and Projects are guaranteed to be backwards compatible in future XSTAMPP versions.
	Telemetry	No. Not at this time.
	User Experience Improvement Program	No. Not at this time.
Help and Support		
	Integrated Help Content	Partially. Implemented but 500 error accessing integrated help on Linux. Not 4K screen compatible.
	Online Help Content	Yes. Tutorials as well as Video Demos.
	"Contact Us" Direct Support	Yes. Accessible online help form provided.
	User Community Forums	No. Not at this time.
Monetary Cost		
	Free	Yes. All software is provided free of charge and will remain free in the future.
	One Time Purchase	No.
	Service Purchase	No.
Deployment Time Cost		

	Low (user manages independently)	Yes. Application and source code can be acquired and deployed in <5 minutes.
	Medium (considerable setup time)	No.
	High (requires staff allocation to maintain)	No.
Collaboration	Does the software support multiple users working together on projects in real-time?	No.
Drawing Canvas	Supports drag and drop functionality?	Yes.
	Supports drag multi-object selection?	Yes. (but only after clicking Marquee button)
	Supports keyboard shortcuts to duplicate controls?	Yes.
	Supports connection fracture mitigation optimizations?	Yes.
	Supports auto-alignment of objects?	Yes.
	Supports auto-resize of objects?	Partially. Control actions can be added to control action lists.
	Supports drawing generation from structured user input data?	No.
	Supports undo and redo features?	Yes.
	Supports batch entry of objects?	Yes.
Textual Input	Supports rich text?	No.
	Supports undo and redo features?	Partially. Some text boxes support common Ctrl-Z and Ctrl-Y keyboard shortcuts to undo and redo, however other text boxes are limited.

Appendix C – Preliminary Systems Theoretic Cybersafety Data Structure Organization

OBJECT	ATTRIBUTES	TYPE	Notes
(General supporting abstract objects below)			
UNIQUE EXPORTABLE OBJECT			A general object for each object to extend that requires a unique identifier and rich description for all important relevant information
	Identifier	UUID	
	Title	Plain Text	string
	Description	Rich Text	
	Browser Exporter	HTML Serializer	Necessary to interface with other tools, toolchains, or services
	Service Exporter	JSON Serializer (and or XML)	Necessary to interface with other tools, toolchains, or services
	Implements / Enforces	(some serializable interface, JSON and maybe XML)	An interface to export the object to a report form or other tool
ORDERED LIST ITEM			
	Item	Unique Exportable Object	
	List Index	Integer	Position in list
	Links	Links	[ ] set
TOOL INPUT			
	Identity	Unique Exportable Object	
	Browser Importer	HTML Deserializer	Necessary to interface with other tools, toolchains, or services
	Service Importer	JSON Deserializer (and or XML)	Necessary to interface with other tools, toolchains, or services
(STPA-SafeSec specific objects below [23])			
NETWORK			From SafeSec paper
	Identity	Unique Exportable Object	
	Subnet	Network Subnet	
DEVICE			
	Extends	Controller	
LINK			

	Label	Object (UUID)	
	From	Object (UUID)	
	To	Object (UUID)	
NETWORK CONNECTION			From SafeSec paper
	Extends	Link	
	Identity	Unique Exportable Object	
NETWORK DEVICE			From SafeSec paper
	Extends	Device	
	Network Address	Network Address	IP or other network visible address
	Network ID	UUID	A unique fingerprint differentiating the device. Potentially could use MAC address fingerprint, though they can be easily spoofed.
	Services	Software Service	Port and the version of the services listening on them
NETWORK CONNECTION			
	Extends	Link	From SafeSec paper
(Step 1 information below)			
SYSTEM PURPOSE			
	Purpose of Analysis	Rich Text	
	System Description	Rich Text	
	System Goals	Goals	[ ] set
GOAL			
	Extends	Ordered List Item	
LOSS			
	Extends	Ordered List Item	
HAZARD / VULNERABILITY			
	Extends	Ordered List Item	
CONSTRAINT			
	Extends	Ordered List Item	
	Applicable Process Model Variables	Process Model Variables	[ ] set
	Applicable Process Model Var. Value Limits	Process Model Variable Values	[ ] set
	Applicable Objects	Link	[ ] set
SYSTEM-LEVEL SAFETY CONSTRAINT			
	Extends	Constraint	
	Boundary	Graph? Tree? Points in space?	Something is necessary to show this type of constraint applies to the system level
(Step 2 below)			

OUTPUT			
	Identity	Unique Exportable Object	
	Links	Links	[ ] set
INPUT			
	Identity	Unique Exportable Object	
	Links	Links	[ ] set
ENVIRONMENTAL OUTPUT			
	Extends	Output	
	Links	Links	[ ] set
ENVIRONMENTAL INPUT			
	Extends	Input	
	Links	Links	[ ] set
ENVIRONMENT			
	Identity	Unique Exportable Object	
	Inputs	Environmental Inputs	[ ] set
	Outputs	Environmental Outputs	[ ] set
TRAINING			
	Identity	Unique Exportable Object	
CONTROLLER			
	Identity	Unique Exportable Object	
	Inputs	Inputs	[ ] set
	Outputs	Outputs	[ ] set
CONTROLLED PROCESS			
	Identity	Unique Exportable Object	
	Inputs	Inputs	[ ] set
	Outputs	Outputs	[ ] set
	Disturbances	Disturbances	[ ] set
	Operational Mode	Operational Mode	[ ] set
EXTERNAL CONTROLLERS			
	Extends	Controller	
	Boundary	Graph? DAG? Tree? Points in space?	Specifies the boundary that the controller is external to.
AUTOMATION			
	Identity	Unique Exportable Object	
DISTURBANCE			
	Identity	Unique Exportable Object	
OPERATIONAL MODE			
	Identity	Unique Exportable Object	
MODEL			
	Identity	Unique Exportable Object	
PROCESS MODEL			
	Extends	Model	

	Controlled Process (Perceived)	Controlled Process	
PROCESS MODEL VARIABLE			
	Identity	Unique Exportable Object	
	Value	Process Model Variable Value	
PROCESS MODEL VARIABLE VALUE			
	Identity	Unique Exportable Object	
	Value	Measurable	
	Type	Classifiable	If strong static typing is required, type must be captured and checked
	Hazardous or Unsafe	Boolean (True or False)	
ENVIRONMENT MODEL			
	Extends	Model	
	Environment (Perceived)	Environment	
OTHER CONTROLLER MODEL			
	Extends	Model	
	Other Controller (Perceived)	Controller	
CONTROL ALGORITHM			
	Identity	Unique Exportable Object	
HUMAN CONTROLLER			
	Name	Text	
	Title	Text	
	Identity	Unique Exportable Object	
	Extends	Controller	
	Env. Model	Environment Model	
	Automation Model	Automation Model	
	Controlled Process Model	Controlled Process Model	
	Other Controller Model	Other Controllers Model	
	Control Action Generation / Mental Processing	Control Algorithm	
HUMAN CONTROLLER MODEL			
	Extends	Model	
	Human Controller (Perceived)	Human Controller	
AUTOMATED CONTROLLER			
	Extends	Controller	
	Control Algorithm	Contorl Algorithm	
	Human Controller Model	Human Controller Model	
	Other Controllers Model	Other Controller Model	
	Operational Mode	Operational Mode	
	Controlled Process Model	Controlled Process Model	
SENSOR			
	Identity	Unique Exportable Object	
ACTUATOR			
	Identity	Unique Exportable Object	

DISPLAY			
	Identity	Unique Exportable Object	
	Content	Rich Text	
ALARM			
	Identity	Unique Exportable Object	
	Description	Rich Text	
	Trigger Condition	Rich Text	
OTHER SENSORY FEEDBACK			
	Identity	Unique Exportable Object	
RESPONSIBILITIES			
	Identity	Unique Exportable Object	
	Relevant Object(s)	Link	[ ] set
FUNCTIONAL CONTROL STRUCTURE			
	Extends	Control Structure	The functional control structure is a functional implementation of a control structure
	Hierarchy	Directed Graph? Tree? Points in space?	The functional control structure must support the notion of a hierarchy of objects
	Boundary	Graph? Tree? Points in space?	The functional control structure must support the notion of a boundary for sub-system identification or description.
	Environmental Inputs	Environmental Input	[ ] set
	Environmental Outputs	Environmental Output	[ ] set
	Controllers	Controller	[ ] set
	Controlled Processes	Controlled Process	[ ] set
	Sensors	Sensor	[ ] set
	Actuators	Actuator	[ ] set
	Displays	Display	[ ] set
	Alarms	Alarm	[ ] set
	Other Sensory Feedbacks	Other Sensory Feedback	[ ] set
	Networks	Network	[ ] set
	Links	Link	[ ] set
	Control Actions	Control Action	[ ] set
CONTROL ACTION			
	Identity	Unique Exportable Object	
TECHNICAL CONTROL ACTION			
	Extends	Control Action	
HUMAN CONTROL ACTION			
	Extends	Control Action	

FEEDBACK			
	Identity	Unique Exportable Object	
FUNCTIONAL RELATIONSHIP			
	Identity	Unique Exportable Object	
	Link	Link	
FUNCTIONAL INTERACTION			
	Identity	Unique Exportable Object	
	Link	Link	
PHYSICAL INTERFACE			
	Identity	Unique Exportable Object	
PHYSICAL CONTROLLED PROCESS			
	Extends	Controlled Process	
	Physical Interface	Physical Interface	[ ] set
(Step 3 below)			
CONTEXT TABLE			
	System Variables	Process Model Variables	[ ] set
	System Variable Values	Process Model Variable Values	[ ] set
UNSAFE CONTROL ACTION (UCA)			
	Extends	Control Action	
	Source	Controller	
	Type	One of Handbook Common Four Possibilities	Enumeration {Provides, Not Provides, Wrong Timing, Wrong Duration}
	Context	Context Table	Matrix of Process Model Variable Values
	Linked Vulnerabilities	Link	[ ] set
HUMAN UCA			
	Extends	UCA	
	Linked Human Controllers	Link	[ ] set
TECHNICAL UCA			
	Extends	UCA	
	Linked Automated Controllers	Link	[ ] set
CONTROLLER CONSTRAINT			
	Extends	Constraint	
	Applicable Controller	Controller	
(Step 4 below)			
CONTROL ACTION INFORMATION LINK			
	Extends	Link	
	Hazardous	Boolean (True or False)	
CONTROL ACTION DISRUPTION			
	Control Path	Tree? DAG? Graph?	
	ID'd From Control Action Not Executed	Boolean	

	ID'd From Control Action Improperly Executed	Boolean	
CONTROL ACTION SCENARIO			
	ID'd From Failures Involving Controller (physical)	Boolean	part of a two-byte bitmap (for compression)
	ID'd From Inadequate Control Algorithm	Boolean	part of a two-byte bitmap (for compression)
	ID'd From Unsafe Control Input (from another controller)	Boolean	part of a two-byte bitmap (for compression)
	ID'd From Inadequate Process Model	Boolean	part of a two-byte bitmap (for compression)
	ID'd From Unsafe Controller Behavior	Boolean	part of a two-byte bitmap (for compression)
	ID'd From Causes Of Inadequate Feedback and Information	Boolean	part of a two-byte bitmap (for compression)
	ID'd From Feedback or Information Not Received	Boolean	part of a two-byte bitmap (for compression)
	ID'd From Inadequate Feedback Is Received	Boolean	part of a two-byte bitmap (for compression)
	ID'd From Control Path Examination	Boolean	part of a two-byte bitmap (for compression)
	ID'd From Controlled Process Examination	Boolean	part of a two-byte bitmap (for compression)
	Other Factors Related To Controlled Process Examination	Rich Text	
UNSAFE CONTROL ACTION SCENARIO			
	Identity	Unique Exportable Object	
	UCA Link	Link	
UNSECURE CONTROL ACTION SCENARIO			
	Identity	Unique Exportable Object	
	UCA Link	Link	
(Outputs below; informally referred to as Step 5, or Final Steps)			
TOOL OUTPUT			
	Identity	Unique Exportable Object	
	Relevant Objects	Unique Exportable Object	[ ] set

	Browser Standardizer	HTML Serializer/Deserializer	Necessary to interface with other tools, toolchains, or services
	Service Standardizer	JSON and XML Serializer/Desearalizer	Necessary to interface with other tools, toolchains, or services
REQUIREMENT (GENERATION)			
	Extends	Tool Output	
REQUIREMENT (ALTERATION)			
	Extends	Requirement	
	Change / Difference	Rich Text	
SYSTEM ARCHITECTURE ALTERATION			
	Extends	Tool Output	
	Change / Difference	Rich Text	
REPORT (COLLECTION OF TOOL OUTPUTS)			
	Outputs	Tool Output	[ ] set
	System Purpose	System Purpose	[ ] set
	System Description	Rich Text	
	Goals	Goal	[ ] set
	Losses	Loss	[ ] set
	Vulnerabilities	Vulnerability	[ ] set
	System Level Constraints	System Level Constraints	[ ] set
	Relevant Functional Control Structure Diagram(s)	Functional Control Structure (rendered)	[ ] set
	Unsafe Control Actions	UCA	[ ] set
	Relevant Context Tables	Context Table	[ ] set
	Relevant Environments	Environment	[ ] set
	Unsafe & Unsecure Control Action Scenarios	Control Action Scenarios	[ ] set

Citations

[1] IPA/SEC. “The 2nd Japanese STAMP Workshop Abstracts.” *2nd Japanese STAMP Workshop*, Japanese Information-Technology Promotion Agency (IPA), 2017, www.ipa.go.jp/english/sec/complex_systems/stamp_workshop-2.html.

[2] Stiki-Information Security. “STPA Module.” *Risk Management Studio*, RM Studio, 2017, www.riskmanagementstudio.com/features/stpa.

[3] Engineering, Safeware. “SpecTRM: Specification Tools and Requirements Methodology.” *Safeware Engineering Corporation: Products*, Safeware Engineering, 2 Feb. 2018, www.safeware-eng.com/software%20safety%20products/software%20safety%20products.htm.

[4] Safety-Critical Systems Research Lab Team of ZHAW Zurich University of Applied Sciences. “SAHRA - STPA Based Hazard and Risk Analysis.” *SAHRA Becomes ANSHIN*, SAHRA, 2 Feb. 2018, www.sahra.ch.

[5] “Transportation Systems Safety Hazard Analysis Tool (SafetyHAT) User Guide (Version 1.0).” *National Transportation Library*, U.S. Department of Transportation, 24 Mar. 2014, rosap.ntl.bts.gov/view/dot/12034.

[6] Abdulkhaleq, Asim, and Stefan Wagner. “XSTAMPP: An EXtensible STAMP Platform as Tool Support for Safety Engineering.” *OPUS Publication Server*, University of Stuttgart , 2015, elib.uni-stuttgart.de/handle/11682/3550.

[7] Krauss, Sven Stefan, et al. “Tool Qualification Considerations for Tools Supporting STPA.” *Tool Qualification Considerations for Tools Supporting STPA*, 3rd European STAMP Workshop, 2015.

[8] F. Asplund, Safety and Tool Integration: A System-Theoretic Process Analysis, Technical Report ISRN/KTH/MMK-R-12/01-SE,KTHRoyal Institute of Technology, Stockholm, Sweden,2012.

[9] Bandor, Michael S. “Quantitative Methods for Software Selection and Evaluation.” *Software Engineering Institute*, Carnegie Mellon University, 2006, www.sei.cmu.edu/reports/06tn026.pdf.

[10] Thomas, John. “Extending And Automating A Systems Theoretic Hazard Analysis For Requirements Generation and Analysis.” *MIT*, 2013.

[11] Thomas, John, and Dajing Suo. “A Tool-Based STPA Process.” *STPA with Tool Assistance*, MIT Partnership for a Systems Approach to Safety (PSAS), psas.scripts.mit.edu/home/wp-content/.../Thomas-Suo-Tool-based-STPA-process.pdf.

[12] Suo, Dajing, and John Thomas. *An STPA Tool*. MIT Partnership for a Systems Approach to Safety (PSAS), Mar. 2015 psas.scripts.mit.edu/home/wp-content/uploads/2014/03/STPA_ToolV1.4-san.pdf.

[13] Suo, Dajing, and John Thomas. “An STPA Tool.” *Vimeo*, Michael Stone, 6 Feb. 2015, vimeo.com/96580336.

[14] Gurgel, Danilo Lopes, et al. *A Rule-Based Approach for Safety Analysis Using STAMP/STPA* . IEEE Conference Publication, 2015, ieeexplore.ieee.org/document/7311464/.

[15] Abdulkhaleq, Asim, and Daniel Lammering. “Using STPA in Compliance with ISO26262 for Developing a Safe Architecture for Fully Automated Vehicles.” *Automotive-Safety and Security 2017*, www.automotive2017.de/programm/Vortraege/S4V2%20Lammering_Abdulhaleq%20Automotive_Presentation%202017_v5.pdf.

[16] "Russian Government Cyber Activity Targeting Energy and Other Critical Infrastructure Sectors | US-CERT". US-CERT.Gov, 2018, https://www.us-cert.gov/ncas/alerts/TA18-074A. Accessed 23 July 2018.

[17] Leveson, Nancy G, and John P Thomas. STPA Handbook, MIT, Mar. 2018, psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf.

[18] Andrews, T., Clohessy, A., English, E., Jindal, S., & Kher, P.*EM.413 OS14 System Reliability and Safety - Team 19 Project: Holographic Human Machine Interface (Holographic HMI)* [Scholarly project]. In *MIT SDM - System Design and Management*. Mar. 2017. https://sdm.mit.edu/ (available upon request)

[19] Dr. Bill Young’s 2018 STAMP Workshop STPA-Sec for Cybersecurity presentation (not yet publicly available).

[20] Friedberg, I., McLaughlin, K., Smith, P., Laverty, D., & Sezer, S. STPA-SafeSec: Safety and security analysis for cyber-physical systems. *Journal of Information Security and Applications*. 2016. http://www.sciencedirect.com/science/article/pii/S2214212616300850#15-STPA-SafeSec - 1-s2.0-S2214212616300850-main.pdf

[21] E-ISAC, and SANS ICS. “Analysis of the Cyber Attack on the Ukrainian Power Grid.” *Analysis of the Cyber Attack on the Ukrainian Power Grid*, 18 Mar. 2016.

[22] Thomas, J., deLemos, F., Leveson, N. (2012, Evaluating *the Safety of Digital Instrumentation and Control Systems in Nuclear Power Plants*. MIT, Nov. 2012, sunnyday.mit.edu/papers/MIT-Research-Report-NRC-7-28.pdf.

[23] Friedberg, I., McLaughlin, K., Smith, P., Laverty, D., & Sezer, S. (2016). STPA-SafeSec: Safety and security analysis for cyber-physical systems. *Journal of Information Security and Applications*. http://www.sciencedirect.com/science/article/pii/S2214212616300850#15-STPA-SafeSec - 1-s2.0-S2214212616300850-main.pdf

[24] “User Guide for ACTS.” Computer Security Resource Center (CSRC), June 2018, csrc.nist.gov/CSRC/media/Projects/Automated-Combinatorial-Testing-for-Software/documents/acts_user_guide_2_92.pdf.

[25] Leveson, N., Daouk, M., Dulac, N., & Marais, K. (2003, October 30). *A Systems Theoretic Approach to Safety Engineering* [Scholarly project]. In *MIT Aero/Astro System Safety and Software Engineering Research Papers*. Retrieved July, 2018, from http://sunnyday.mit.edu/accidents/external2.pdf

[26] "STAMP Tools | Partnership For Systems Approaches To Safety And Security (PSASS)". Psas.Scripts.Mit.Edu, 2018, https://psas.scripts.mit.edu/home/2016-2/. Accessed 20 Jan 2018.

[27] "Reference — STAMP Workbench 1.0.0 Documentation". Ipa.Go.Jp, 2018, https://www.ipa.go.jp/sec/stamp_wb/manual/reference.html. Accessed 23 June 2018.

[28] Leveson, Nancy G. A New Approach To Hazard Analysis For Complex Systems. Massachusetts Institute Of Technology, 2018, http://sunnyday.mit.edu/papers/issc03-stpa.doc. Accessed 23 July 2018.

[29] Nourian, Arash, and Stuart Madnick. A System Theoretic Approach To The Security Threats In Cyber Physical Systems: Applied To Stuxnet. Composite Information Systems Laboratory (CISL), 2014, https://ic3.mit.edu/wp-content/uploads/2014-13.pdf. Accessed 23 July 2018.

[30] Salim, Hamid, and Stuart Madnick. Cyber Safety: A Systems Thinking And Systems Theory Approach To Managing Cyber Security Risks. Composite Information Systems Laboratory (CISL), 2014, https://ic3.mit.edu/wp-content/uploads/2014-12.pdf. Accessed 23 July 2018.

[31] von Neumann, John, and Arthur W. Burks. Theory Of Self-Reproducing Automata. University of Illinois Press, 1966.

[32] Niazi, Muaz; Hussain, Amir (2011). "Agent-based Computing from Multi-agent Systems to Agent-Based Models: A Visual Survey" (PDF). Scientometrics. Springer. 89 (2): 479–499. doi:10.1007/s11192-011-0468-9. Archived from the original (PDF) on October 12, 2013.

[33] Sargent, R. G. (2000). "Verification, validation and accreditation of simulation models". 2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165). 1. pp. 50–59. doi:10.1109/WSC.2000.899697. ISBN 0-7803-6579-8.

[34] Niazi, Muaz; Hussain, Amir; Kolberg, Mario. "Verification and Validation of Agent-Based Simulations using the VOMAS approach" (PDF). Proceedings of the Third Workshop on Multi-Agent Systems and Simulation '09 (MASS '09), as part of MALLOW 09, Sep 7–11, 2009, Torino, Italy. Archived from the original (PDF) on June 14, 2011.

[35] "Agent-Based Model". En.Wikipedia.Org, 2018, https://en.wikipedia.org/wiki/Agent-based_model. Accessed 15 Aug 2018.

[36] Niazi, Muaz A. K. (June 11, 2011). "Towards A Novel Unified Framework for Developing Formal, Network and Validated Agent-Based Simulation Models of Complex Adaptive Systems". University of Stirling.

[37] Kuhn, D. Richard et al. "Advanced Combinatorial Test Methods For System Reliability | CSRC". Csrc.Nist.Gov, 2010, https://csrc.nist.gov/publications/detail/journal-article/2010/advanced-combinatorial-test-methods-for-system-reliability. Accessed 21 July 2018.

[38] Yu, Linbin et al. "Efficient Algorithms For T-Way Test Sequence Generation - IEEE Conference Publication". *Ieeexplore.Ieee.Org*, 2018, https://ieeexplore.ieee.org/document/6299217/. Accessed 21 July 2018.