WebSTAMP: a Web Application for STPA & STPA-Sec

Fellipe G. R. Souza[1*], Daniel P. Pereira[1], Rodrigo M. Pagliares[2], Simin Nadjm-Tehrani[3] and Celso M. Hirata[1]
[1] Instituto Tecnológico de Aeronáutica, Brazil
[2] Universidade Federal de Alfenas, Brazil
[3] Linköping University, Sweden

ABSTRACT

STAMP (System-Theoretic Accident Model and Processes) techniques such as STPA (System-Theoretic Process Analysis) and STPA-Sec (STPA for Security) have been applied only in an ad-hoc manner, without the aid of tools. More recently, tools have been proposed to help the application of STPA and STPA-Sec. Most of the tools focus on user experience issues and do not cover all the aspects of STPA and STPA-Sec. Three aspects of tools are systematization, automation and analysis completeness. Systematization allows the analysis to be performed in a more disciplined way while automation allows a more time efficient analysis. Analysis' completeness is the analysis coverage in a given domain. We identify the essential requirements supporting business and stakeholders' needs for a STAMP based tool. We propose a STAMP-compliant web application, named WebSTAMP, for STPA and STPA-Sec. WebSTAMP is intended to aid analysts throughout the analysis process in a more automated and comprehensive way, and it aims to be a collaborative tool. We illustrate how the requirements are implemented in the current version of WebSTAMP with an example of use. The results show that WebSTAMP assists analysts to conduct safety and security analyses in a more systematic, automated and comprehensive manner.

Keywords: STAMP; STPA; STPA-Sec; Web Application.

1. INTRODUCTION

STAMP (Systems-Theoretic Accident Model and Processes) (Leveson, 2011) is an accident causality model, based on system theory (Von Bertalanffy, 1968). The goal of STAMP is to understand why accidents occur and how to use that understanding to create new and better ways to prevent accidents. STAMP is based on three main concepts: (i) Safety Control Structure - a hierarchical representation of the system under analysis on which upper-level components impose constraints on lower level components; (ii) Process Model - a model of the process being controlled; and (iii) Safety Constraints – restrictions that the system components must satisfy to assure safety.

STPA (System-Theoretic Process Analysis) is a technique based on STAMP for accident analyses. STPA has two main steps. The first step is to identify how Control Actions (CAs) issued wrongly or not issued would lead the system to a hazardous scenario. A CA is a command sent to the controlled process or lower-level components. The analyst must identify cases of provision of control action and contexts where the CA can be hazardous. In general, there are four cases: (i) Providing CA causes hazard; (ii) Not providing CA causes hazard; (iii) Providing CA too early, too late or in wrong order causes hazard and; (iv) Stopping CA too soon or applying CA too long causes hazard (Leveson, 2011). Contexts are the states of the process model. Step 1 identifies the

[*] Corresponding author: +5535988114110, fellipeguilhermerey@gmail.com

potentially Unsafe Control Actions (UCAs) and their contexts and associated Safety Constraints (SC).

Step 2 reveals potential causes of issuing UCAs or not issuing safe CAs. The goal of Step 2 is twofold (Leveson 2011): (i) discover scenarios and associated causal factors for issuing UCAs or not issuing safe CAs and (ii) generate safety requirements. Safety requirements assist designers in eliminating or mitigating the potential causes of UCAs and the occurrence of hazards.

STPA-Sec (System-Theoretic Process Analysis for Security) is an extension of STPA that allows safety and security analysis (Young & Leveson, 2014). Instead of Safety Control Structure, STPA-Sec uses a more general model: Functional Control Structure (FCS). It uses the term Hazardous Control Action (HCA) extending the concept of UCA to encompass both unsafe and insecure CAs. The major difference between STPA and STPA-Sec (beyond the focus on Safety or Security) resides in Step 2. In addition to the STPA tasks performed in Step 2, in STPA-Sec, analysts must trace hazardous control actions through *Information Lifecycle*, place scenarios on a *D4 Matrix* and conduct *wargame* security scenarios to select a control strategy (Young & Porada, 2017).

Completing an STPA or STPA-Sec analysis is a complex and, often, repetitive task that demands expertise, time, and effort for elaboration and verification of safety and security analysts (Leveson & Thomas, 2013). The difficulties are related to finding all HCAs, scenarios, associated causal factors and deriving the safety and security requirements. It is common to miss cases (e.g. HCAs, scenarios, causal factors and requirements) when performing STPA or STPA-Sec.

There are approaches and tools that aim to help analysts to complete an STPA or STPA-Sec analysis; however, they focus mainly on Step 1 of STPA and user experience issues. Three capabilities of a tool with respect to STPA analysis are systematization, automation, and completeness. The tool systematization capability allows performing the analysis following a process, i.e. in a more disciplined way. The tool automation capability allows performing the analysis using some knowledge about the system that results in a more time-efficient analysis. It allows performing repetitive analysis of contexts automatically to explore the system's knowledge. The tool completeness capability is the ability of the tool to cover all states in a given domain. Completeness is relative to a domain of actions prescribed in a model, which is restricted by the assumptions made on it. The challenge for the tools is to help analysts to perform STPA following a systematic process, aiding in repetitive tasks, and resulting in a complete list of hazardous control actions and safety and security requirements.

We propose a STAMP compliant web application, named WebSTAMP, for STPA and STPA-Sec. WebSTAMP implements essential requirements for safety and security analyses and provides guidance, assisting analysts in a comprehensive way. The remainder of this paper is organized as follows: Section 2 presents approaches and tools that support or provide some sort of systematization, automation, and completeness to STPA/STPA-Sec. Section 3 presents the essential requirements for a software application aiming to support STPA and STPA-Sec analyses. Section 4 presents a system - glucose monitoring and insulin pumping system that is used to illustrate the use of WebSTAMP. We present the use of WebSTAMP for the system in Section 5. Section 6 presents the concluding remarks, with suggestions for future work.

2. RELATED WORK

The need for approaches and tools that provide some sort of systematization, automation, and analysis completeness for STPA and STPA-Sec analysts when conducting hazard analyses has been widely acknowledged (Leveson & Thomas, 2013; Thomas, 2013; Gurgel, Hirata, & Bezerra, 2015; Suo, 2016). There are approaches and tools to aid the automation of STPA Step 1, but there is almost no help, for instance, to aid systematization and/or automation of STPA Step 2. To date, to the best of our knowledge, there is no automation for STPA-Sec.

Thomas (2013) defines a procedure to perform Step 1 of STPA systematically. The procedure is based on a formal mathematical structure. Thomas defines four elements to construct a Control Action (CA): "Source", a component responsible to issue a control action; "Type", that informs whether the CA is provided or not provided; “Control action” representing the CA and; "Context", which is the system or environmental state to analyse the particular CA.

Gurgel et al. (2015) propose a rule-based approach to automate the detection of hazardous contexts in Step 1 of STPA. Their approach is based on the results presented by Thomas (2013). In the rule-based approach, a safety analyst defines a rule for a specific CA using AND expression of pairs of variable-value. In an expression, a specific value or ANY value must be assigned to each variable, characterizing a *context* or a set of contexts. They also show that the rules to define unsafe control actions are a powerful way to perform the analysis.

Suo (2016) presents a proof of concept for an STPA tool. The tool assists analysts when performing Step 1 by applying logical simplifications to the Step 1 results. It allows generating simplified requirements that address UCAs identified in the original STPA analysis. Our web-based tool (described in Section 5) adopts the rule-based approach and Thomas' procedure to automate STPA Step 1. Thomas (2013), Gurgel et al. (2015) and Suo (2016) focus on automation of Step 1. We focus on both steps of STPA.

There are three tools supporting STPA Step 2: STAMP Workbench (Information-Technology Promotion Agency - IPA, 2018), SafetyHAT (Becker & Van Eikema Hommes, 2014), and XSTAMPP (Abdulkhaleq & Wagner, 2015). The tools are available as desktop applications. STAMP Workbench (Information-Technology Promotion Agency - IPA, 2018) provides a tool with step-by-step guidance for beginners. The tool provides an environment for analysts to store an STPA analysis (STPA-Sec is not supported), but it does not provide automation for STPA Step 1 or Step 2.

SafetyHAT (Becker & Van Eikema Hommes, 2014) aims to help safety analysts by providing six guide phrases to aid the identification of UCAs and 26 causal factors to help the evaluation of the UCAs. A current restriction of SafetyHAT is the domain-specific characteristic of the tool. Nine of the causal factors are exclusive to the transport domain. Another disadvantage is that, although SafetyHAT provides ways to identify causal factors in STPA Step 2, the tool does not identify safety requirements (necessary to eliminate or mitigate potential causes of hazards). SafetyHAT does not support STPA-Sec.

Abdulkhaleq and Wagner's tool, named XSTAMPP (Abdulkhaleq & Wagner, 2015), is an open-source platform based on the Eclipse Plug-in Development Environment (PDE) (Steinberg, Budinsky, Merks, & Paternostro, 2008) and Rich Client Platform (RCP) (McAffer, Lemieux, & Aniszczyk, 2010) that supports STPA and STPA-Sec. XSTAMPP implements Thomas’s (2013) approach for Step 1. Although XSTAMPP supports documenting causal factors for Step 2, it neither supports a systematic way to identify them nor allows automatic generation of scenarios and safety requirements.

Thomas (2013), Gurgel et al. (2015), Suo (2016) and XSTAMPP (2015) provide a way to find unsafe control actions in Step 1 and formalize them to be used in Step 2. SafetyHAT (2014) is the only one work that partially addresses the difficulty related to Step 2.

3. ESSENTIAL REQUIREMENTS FOR STPA/STPA-SEC SOFTWARE APPLICATIONS

This section describes the requirements for a tool aiming to support safety and security analyses with STPA and STPA-Sec. A requirement analysis process is usually used to derive the essential requirements in the system from a set of business and stakeholders needs (Faulconbridge & Ryan, 2014). Needs are capabilities stated at the business operations levels. Requirements are formal and structured statements that can be validated.

Functional Requirement (FR) means a function, something that the system should do or provide to the stakeholders. Non-Functional Requirement (NFR) refers to properties, qualities or attributes that the system must possess. NFR may also refer to some condition that the system must meet or some constraint under which it must be developed or operate (Faulconbridge & Ryan, 2014). Based on our experience of analysis, we propose a set of essential FRs and NFRs for an STPA & STPA-Sec tool, as shown in Table 1. In this paper, we focus on the FRs, particularly on the systematization, automation, completeness aspects. Section 5 provides the details of the systematization, automation, and completeness aspects. Other requirements of Table 1 are briefly discussed in what follows.

Change management (FR05) allows tracing backward and forward any element of the analysis. Analysis traceability ensures that one knows where each analysis element comes from, which analysis element is related to it, and which analysis element comes from it. For instance, from an HCA we can navigate back to a system hazard (backward traceability) or to an associated causal factor (forward traceability). Traceability also supports configuration control. If one needs to change an analysis element for any reason, one can see where that analysis element comes from and what is the impact of the change.

Table 1 Essential requirements for an STPA & STPA-Sec tool
(FR - Functional Requirement. NFR - Non-Functional Requirement)

ID	Description
FR01	Create safety and/or security analyses based on STAMP. • Create a new safety/security analysis and retrieve, update, or remove an existing one at any time.
FR02	Systematize and automate Step 1 of STPA & STPA-Sec. • Systematize the generation of the Context Table from the FCS in a complete way. • Automate the fulfilment of the Context Table. • Systematize and automate the generation of the complete list of HCAs from the fulfilled Context Table.
FR03	Systematize Step 2 of STPA & STPA-Sec. • Systematize the analysis of HCAS striving for a clearer analysis. • Systematize the complete generation of scenarios and recommendations for each HCA.
FR04	Provide collaborative analysis. • Provide Web-based collaborative environment considering aspects of coordination, communication, cooperation and awareness.
FR05	Provide change management. • Provide traceability for any element in the analysis. • Provide visualization of the element change impact in the analysis.
FR06	Provide support for verification of the analysis. • Automate the verification of consistency of FCS. • Automate the verification of coverage of hazardous control actions, constraints, and recommendations.
NFR01	Provide rich user experience. • To build the FCS (Drag and drop of visual components desirable). • To interact using graphical user interfaces. • To generate reports.
NFR02	Provide analysis reusability • Reuse analyses of parts of a system.
NFR03	Provide secure environment • Provide access control to users and other security.
NFR04	Provide portability. • The tool must run on different hardware and software platforms.

Table 2 shows how the tools presented in Section 2 meet the requirements identified in Table 1. The cell values of the table include "Yes" (the tool fulfils the requirement) and "No" (the tool does not meet the requirement). We indicate when the tool partially fulfils the requirements (values “Only Safety” and “Only Windows”) and when the requirement is not described in the tool's documentation (value "Not described"). We discuss how WebSTAMP (second column) meets the requirements in Section 5.

Table 2 Comparison of WebSTAMP, STAMP Workbench, SafetyHAT and XSTAMPP taking into consideration FRs and NFRs

Requirements (FR and NFR)	**WebSTAMP**	**STAMP Workbench**	**SafetyHAT**	**XSTAMPP**
FR01 - Create safety and/or security analyses based on STAMP.	Yes	Only Safety	Only Safety	Yes
FR02 - Systematize and automate Step 1 of STPA & STPA-Sec.	Yes	No	No	Yes
FR03 - Systematize Step 2 of STPA & STPA-Sec.	Yes	No	No	Not described
FR04 - Provide collaborative analysis.	Yes	No	No	No
FR05 - Provide change management.	Yes	Not described	Yes	Yes
FR06 - Provide support for verification of the analysis.	No	Not described	Not described	Yes
NFR01 - Provide rich user experience.	No	Yes	No	Yes
NFR02 – Provides analysis reusability.	No	Not described	Not described	Yes
NFR03 - Provide security environment	No	No	No	No
NFR04 – Provide portability	Yes	Only Windows	Only Windows	Yes

4. USE CASE: GLUCOSE MONITORING AND INSULIN PUMPING SYSTEM

This section describes the glucose monitoring and insulin pumping system that is employed as an example of the use of WebSTAMP. People with diabetes may take 1-2 insulin injections of a long-acting insulin every day and three or more injections of rapid-acting insulin for meals and snacks. The typical person with Type 1 diabetes can take 4-7 injections a day. Many people currently receive insulin through an insulin pen or a syringe.

An insulin pump delivers rapid-acting insulin in two way: basal and bolus. Basal is the insulin a person needs even in the absence of food. The basal rate replaces the long-acting injection that a person takes. The insulin pump is programmed to give insulin every hour throughout the hour referred to as basal insulin. Bolus is the insulin a person takes for food or to correct a high blood sugar.

Once a person is using a pump, all insulin is delivered through the pump and shots are no longer necessary. In October 2016, Food and Drug Administration (FDA) approved the first pump of its kind (a hybrid closed loop pump). It is an insulin pump with an integrated continuous glucose monitor that can increase and decrease, within a range, the basal rate on its own, depending on what the monitor reads. It is referred to as a hybrid because it cannot give a bolus dosage for food; the user needs to enter carbs in order to obtain the bolus dosage.

We use a Glucose Monitoring and Insulin Pumping System (GMIPS) as the example throughout the remainder of this paper. GMIPS is an insulin pump, with a continuous glucose monitor, integrated with a mobile application to provide a smooth treatment of diabetes. Unlike the predecessors, the system here has a mobile app that controls the insulin pump. The system makes the life of patients easier, improving their health awareness, by means of monitoring the patient’s glucose, controlling the injection of insulin, and providing alerts about its operation. Although the system is a fictitious one, we strive to perform the analysis presented herein aligning it with real system specifications.

5. WEBSTAMP

WebSTAMP is a web application that supports STPA and STPA-Sec. The technologies used to build WebSTAMP are HyperText Markup Language (HTML), Cascade Style Sheets (CSS), Javascript, PHP (Nixon, 2012), and the Laravel framework (Bean, 2015). MySQL (MySQL, 2000) stores engineering data of safety analyses (as projects) created within the tool.

The application is partially collaborative, enabling analysts to work remotely on a shared analysis (FR04). Communication must be provided by a conference tool, such as Skype. Coordination must be provided by the users using social protocol. Synchronization is provided by reload of the Web page. There are advantages for the end user when using a web application, such as the possibility of access from anywhere and platform independence (NFR04). WebSTAMP treats the analyses as projects with associated roles and artefacts. It allows managing several projects; it keeps traceability of identified hazards, and currently it provides partial automation for some tasks of Steps 1 and 2.

5.1. Analysis of GMIPS with WebSTAMP

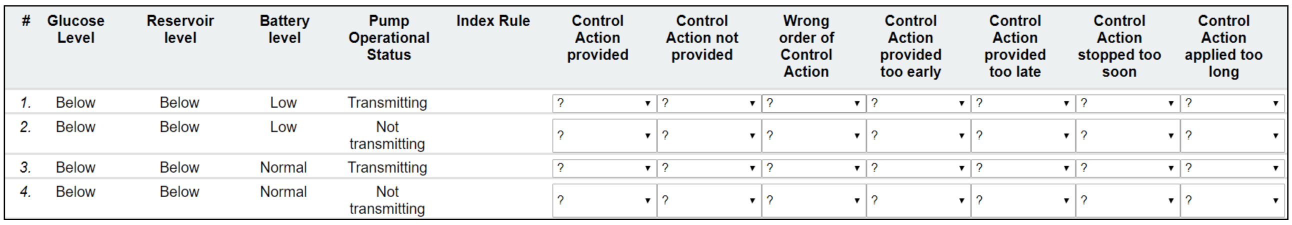
To illustrate how the tool supports STPA-Sec, we conduct an analysis of GMIPS using WebSTAMP. Sections 5.1.1. and 5.1.2. describe STPA Step 1 and STPA Step 2 respectively, assuming that the project is of type "Safety and Security" and the *Fundamentals* part - System goals and purpose, losses, hazards, system security constraints and FCS - is completed (FR05 is applicable in Fundamentals).

5.1.1. Step 1

As seen in Section 1, for each Control Action identified in the *Fundamentals* part, the analyst must discover cases when the control action is hazardous. To help the discovery of the cases, WebSTAMP employs the rule-based approach for STPA Step 1 (Gurgel et al., 2015).

Thomas (2013) proposes the creation of a Context Table to find HCAs. A Context Table can be defined as the combination of all process model variables and values. Therefore, the Context Table for the CA "Pumping Insulin" of *Insulin Pump* (controller) is the result of the combination of all states of the variables: *Glucose level*, *Reservoir level*, *Battery level* and *Pump operational status*. The number of rows of the Context Table is a function of the number of states – in this case, the multiplication between three values (Below, Normal, and Above) of *Glucose level*, two possible values (Below and Normal) of *Reservoir level*, two values (Low and Normal) of *Battery level* and two values (Transmitting and Not Transmitting) of *Pump operational status*, resulting in a table with 24 rows. WebSTAMP creates the Context Table automatically and systematically, based on the Process Model variables. This is the way that WebSTAMP addresses FR02.

Figure 1 illustrates the first four rows of the Context Table for *Pumping Insulin* CA. The column "#" is the number of the row. The columns 2 to 5 are the variables and states of the process model of "Insulin Pump" controller (in this case, we have four columns because we have four variables). The column "Index Rule" describes references of rules (we will explain this column later). The remaining columns (columns 7 to 13) are the cases that we must check if the control action is hazardous. Initially, each box has a question mark "?" – it means that the context is not checked. If the box has a space " " - it means that the context is not considered hazardous.



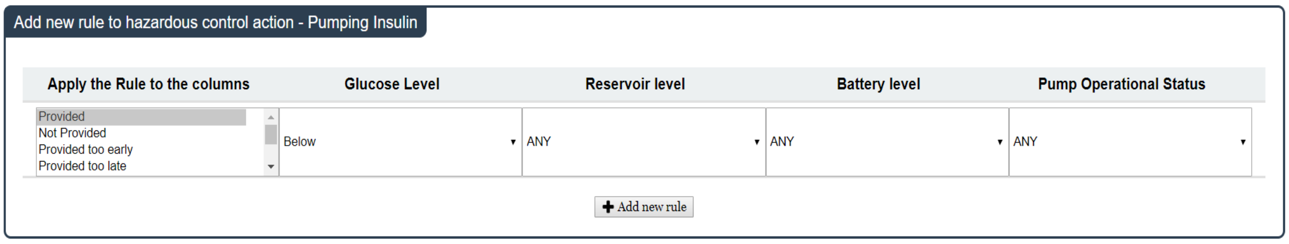
#	Glucose Level	Reservoir level	Battery level	Pump Operational Status	Index Rule	Control Action provided	Control Action not provided	Wrong order of Control Action	Control Action provided too early	Control Action provided too late	Control Action stopped too soon	Control Action applied too long
1.	Below	Below	Low	Transmitting		?	?	?	?	?	?	?
2.	Below	Below	Low	Not transmitting		?	?	?	?	?	?	?
3.	Below	Below	Normal	Transmitting		?	?	?	?	?	?	?
4.	Below	Below	Normal	Not transmitting		?	?	?	?	?	?	?

Figure 1: Context table for Pumping Insulin CA - excerpt with the first four rows of a total of twenty-four

The context table provides a comprehensive way to find hazardous control actions because the analyst has to verify every possibility. The number of possibilities can be very large. In the example depicted in Figure 1, the analyst must check 168 possibilities for “Pumping Insulin” control action, which requires a large amount of effort. By providing visibility of the context table and its cells, WebSTAMP helps addressing the completeness of the analysis.

In many cases, checking the possibilities is repetitive work, because the cases are similar. The rules proposed by Gurgel et al. (2015) aim to help the analyst to automate the context table, filling some columns (7 to 13) automatically as “Hazardous” (or “Unsafe”, in an STPA analysis).

Figure 2 illustrates an example of the rule: Whenever the variable *Glucose Level* has the value “Below”, providing a *Pumping Insulin* CA is hazardous. Therefore, we can translate the rule as When *Glucose level* is “Below”, and *Reservoir level* is “Normal” or “Below” and *Battery level* is “Low” or “Normal” and *Pump operational status* is “Transmitting” or “Not transmitting”, providing the control action “Pumping Insulin” is hazardous.

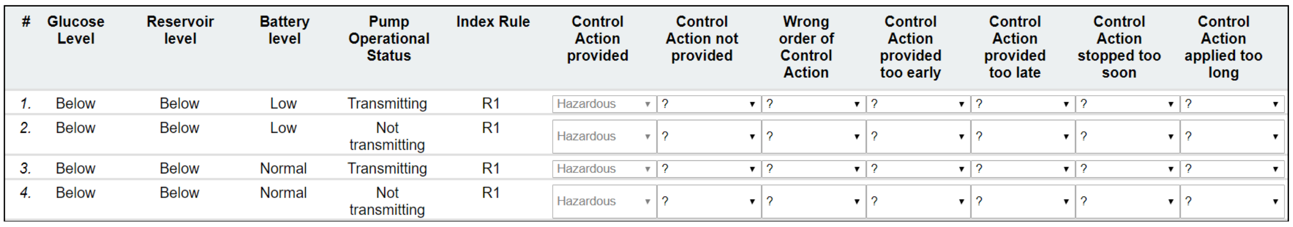

Add new rule to hazardous control action - Pumping Insulin

Apply the Rule to the columns	Glucose Level	Reservoir level	Battery level	Pump Operational Status
Provided Not Provided Provided too early Provided too late	Below	ANY	ANY	ANY

Figure 2: Creating a new rule for “Pumping Insulin” control action

When a rule is created, in every context where the rule applies, the column is marked as “Hazardous” (FR02). Figure 3 depicts the context table after the application of the rule created. The field “Index Rule” is filled with “R1”, where R means “Rule” and 1 is the index of the rule. The column “Control Action provided” is marked as “Hazardous”, because of the rule. For simplicity, we applied the rule only for the column “Provided”, but the rule could also be applied to columns “Provided too early”, “Provided too late”, “Stopped too soon” and “Applied too long”.

Rules are the functionality of WebSTAMP to automate the filling of the context table. In order to take advantage of the functionality, the analyst must have knowledge about the concepts of the system.



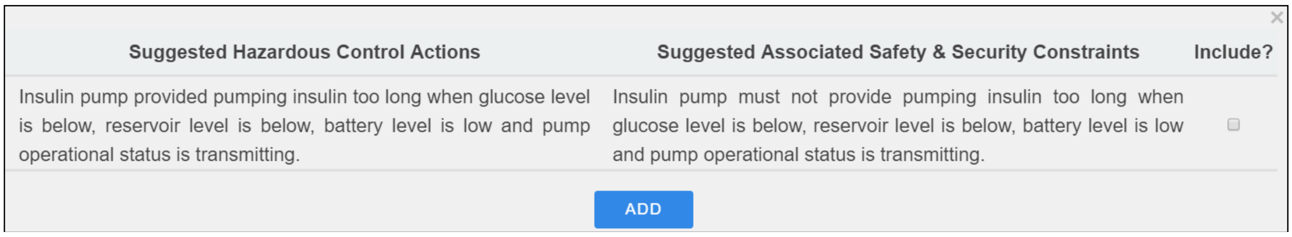
#	Glucose Level	Reservoir level	Battery level	Pump Operational Status	Index Rule	Control Action provided	Control Action not provided	Wrong order of Control Action	Control Action provided too early	Control Action provided too late	Control Action stopped too soon	Control Action applied too long
1.	Below	Below	Low	Transmitting	R1	Hazardous	?	?	?	?	?	?
2.	Below	Below	Low	Not transmitting	R1	Hazardous	?	?	?	?	?	?
3.	Below	Below	Normal	Transmitting	R1	Hazardous	?	?	?	?	?	?
4.	Below	Below	Normal	Not transmitting	R1	Hazardous	?	?	?	?	?	?

Figure 3: Context table for Pumping Insulin CA with rule “R1” – excerpt with the first four rows of a total of twenty-four

When the rule is created, the corresponding hazardous control action is generated. The hazardous control action “Insulin pump provided pumping insulin when glucose level is Below” is automatically created using the four elements to construct a CA (as seen in Section 2) – the controller is “Insulin Pump”, the type is “Provided”, the control action is “pumping insulin” and the context is “glucose level is Below”.

Besides using rules, there are two alternatives to identify hazardous control actions. In the first alternative, the analyst manually checks “Hazardous” for a cell in the context table. In the first alternative, the analyst must also confirm that the control action is hazardous for a selected context. In the second alternative, the analyst can manually choose a combination of context and a type of provision (provided, not provided, provided in wrong order, provided too early, provided too late, stopped too soon or applied too long). The reason for providing the second alternative is to allow performing the analysis without using the context table and rules. Based on our experience, the flexibility is a requirement that the analysts demand.

Figure 4 illustrates an example of the first alternative. In Figure 3, suppose that the cell row #1 and column “Control Action applied too long” is marked as “Hazardous” in the context table. The tool automatically asks for confirmation that the control action is hazardous for this case (FR02). In Figure 4, each marked checkbox “Include?” confirms that the CA is hazardous for that particular context.

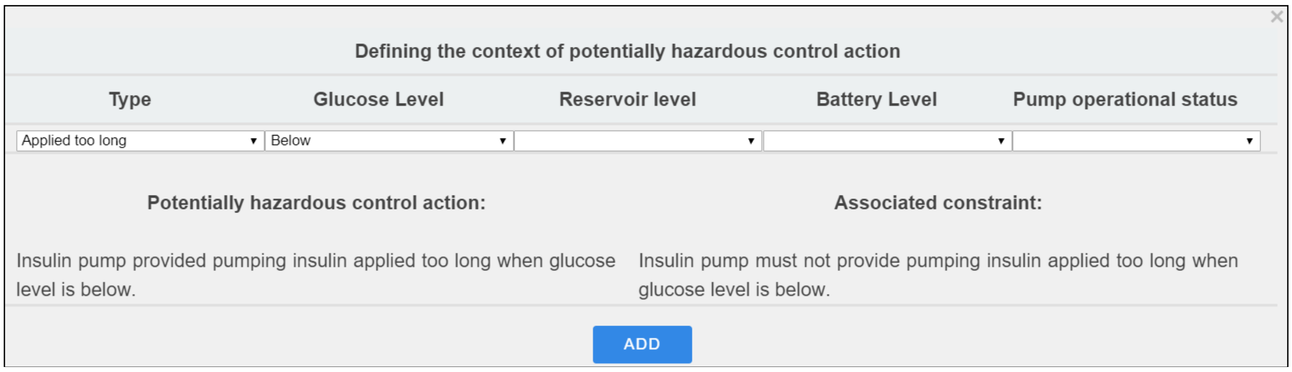


Suggested Hazardous Control Actions	Suggested Associated Safety & Security Constraints	Include?
Insulin pump provided pumping insulin too long when glucose level is below, reservoir level is below, battery level is low and pump operational status is transmitting.	Insulin pump must not provide pumping insulin too long when glucose level is below, reservoir level is below, battery level is low and pump operational status is transmitting.	☐

Figure 4: Including (confirming) a hazardous control action. Each cell manually checked as “Hazardous” in the context has to be confirmed by checking the “Include?” box

Figure 5 depicts the second alternative to create a hazardous control action. The analyst chooses the type of application of the CA and the context manually, using AND expression of pairs variable-value, and the text of hazardous control action is generated automatically. In Figure 5, the columns “Reservoir level”, “Battery level” and “Pump operational status” are blank because the analyst understands that their values can be any for this hazardous control action.

In WebSTAMP, the identified hazardous control actions are automatically translated to associated constraints. The translation is possible because the hazardous control actions are semi-formally described using Thomas’ approach (Thomas, 2013). This is the way that WebSTAMP addresses FR02.



Defining the context of potentially hazardous control action

Type	Glucose Level	Reservoir level	Battery Level	Pump operational status
Applied too long	Below			

Potentially hazardous control action:	Associated constraint:
Insulin pump provided pumping insulin applied too long when glucose level is below.	Insulin pump must not provide pumping insulin applied too long when glucose level is below.

Figure 5: Defining a hazardous control action manually, with text automatically generated by WebSTAMP

In WebSTAMP, the hazardous control actions identified in Step 1 are systematically used as input to complete the Step 2 analysis. This is the way that WebSTAMP addresses FR03. As seen in Section 1, in Step 2 the analyst must find scenarios that lead to hazardous contexts. While Step 1 finds the hazardous control actions, Step 2 discovers the reasons for the control action to be hazardous when provided or not provided (when it is required).

5.1.2. Step 2

WebSTAMP stores the Step 2 information in a tabular form. The table has four columns: “Scenario”, “Associated Causal Factors”, “Recommendations” and “Rationale”. The main goal of Step 2 is to check the generic control loop and find for hazardous scenarios and vulnerabilities. Figure 6 depicts the Generic Control Loop (GCL), based on the Leveson’s GCL for safety (Leveson & Thomas, 2011) with the guidewords for security proposed by Young and Porada

(2017). In the figure, we consider the right-side guidewords are on the right side of the diagonal of the dashed line whereas the left side guidewords are on the left side of the diagonal of the dashed line.

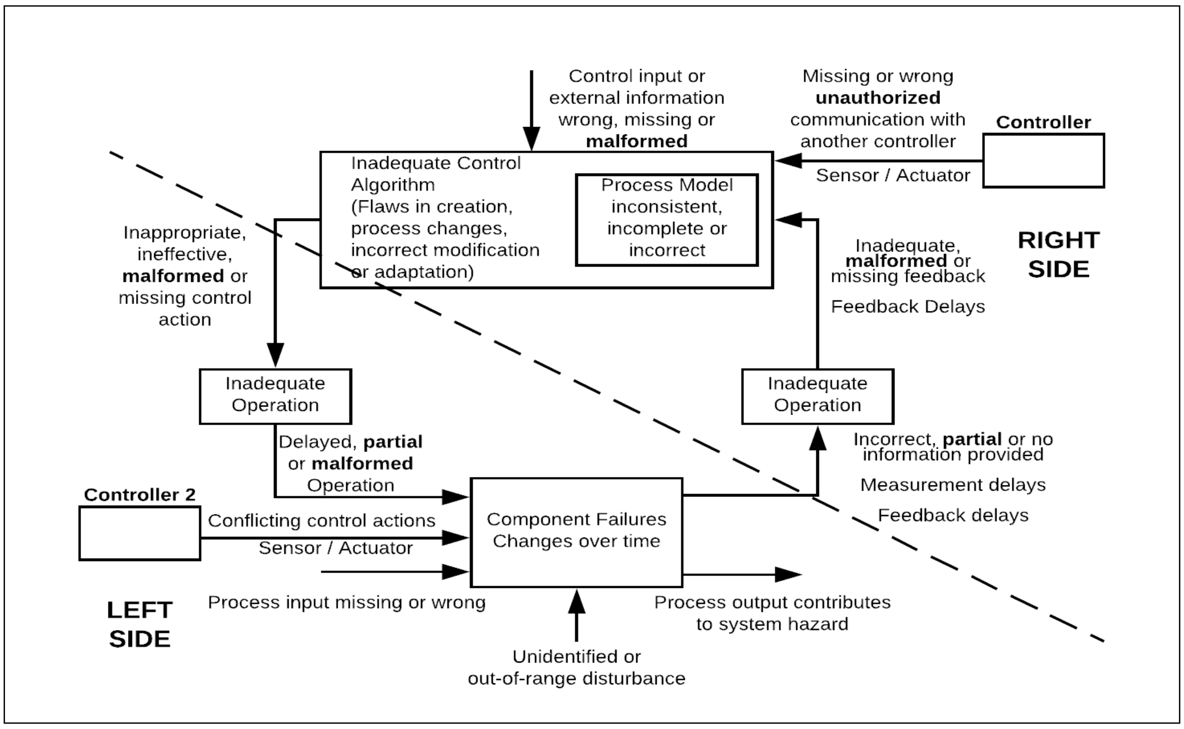

Control input or external information wrong, missing or **malformed**
Missing or wrong **unauthorized** communication with another controller
Controller
Inadequate Control Algorithm (Flaws in creation, process changes, incorrect modification or adaptation)
Process Model inconsistent, incomplete or incorrect
Sensor / Actuator
Inappropriate, ineffective, **malformed** or missing control action
Inadequate, **malformed** or missing feedback
Feedback Delays
RIGHT SIDE
Inadequate Operation
Inadequate Operation
Delayed, **partial** or **malformed** Operation
Incorrect, **partial** or no information provided
Measurement delays
Feedback delays
Controller 2
Conflicting control actions
Sensor / Actuator
Component Failures Changes over time
LEFT SIDE
Process input missing or wrong
Process output contributes to system hazard
Unidentified or out-of-range disturbance

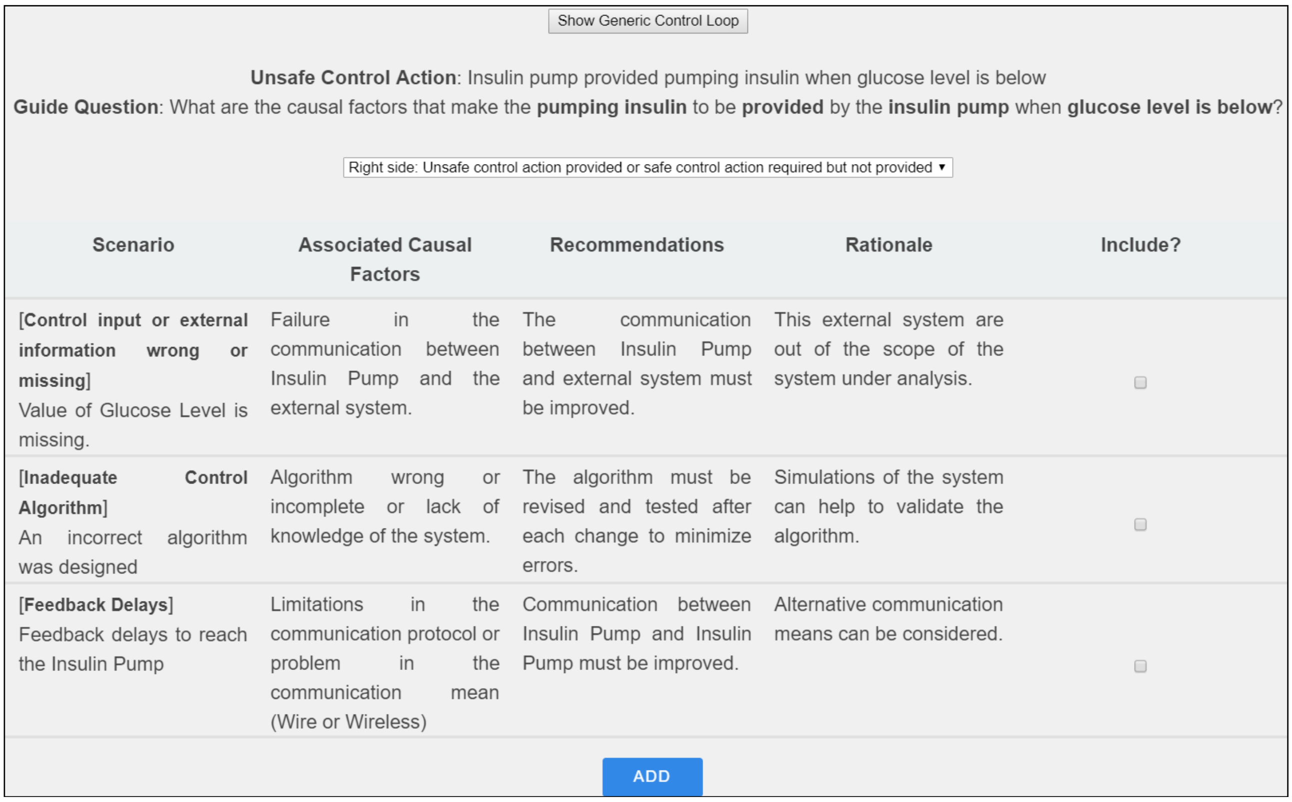
Figure 6: Generic Control Loop. Adapted from Leveson and Thomas (2011) and Young and Porada (2017)

In order to help the analyst, WebSTAMP generates a guide question for each hazardous control action. Figure 7 illustrates the guide question “What are the causal factors that make the pumping insulin to be provided by the insulin pump when the glucose level is below” for the hazardous control action “Insulin pump provided pumping insulin when the glucose level is below”. The guide question helps the analysis by providing all the information to be considered when finding the reasons of the hazardous control action (FR03).

Using WebSTAMP, the analyst can either create his/her own set of scenarios, associated causal factors, recommendations and rationale or use a set of generic scenarios provided by the tool. Figure 8 depicts how the analyst adds a new scenario in WebSTAMP. It is necessary to link the created scenario with a guideword. Although STPA is not a technique driven by guidewords, linking a guideword to the scenario helps to find and identify causal factors. WebSTAMP automatically suggests which side (left or right) to look for the guidewords based on the type of hazardous control action. The three types of hazardous control actions are: a (safe and secure) CA is provided, but not followed or executed adequately; a hazardous CA is provided; a (safe and secure) CA is not provided (but it is required) or issued inadequately.

By listing all the guide questions, WebSTAMP systematically requests that the analyst find causal factors and generate recommendations. This is the way that WebSTAMP addresses FR03.

Using WebSTAMP, the analyst can create his/her own set of guide words. In order to create guidewords related to security threats, based on Young and Porada work (2017), we suggest the usage of Information Life Cycle. For each hazardous control action (or guide question), the analyst must identify the information type (control action, physical energy transfer, sense (event), feedback data, communication data, process variables, code, process input, and process output) and the phase of information (generation, processing, storage, communication, consumption, and destruction) that can be an associated causal factor to address the particular guide question.



Show Generic Control Loop

Unsafe Control Action: Insulin pump provided pumping insulin when glucose level is below
Guide Question: What are the causal factors that make the **pumping insulin** to be **provided** by the **insulin pump** when **glucose level is below**?

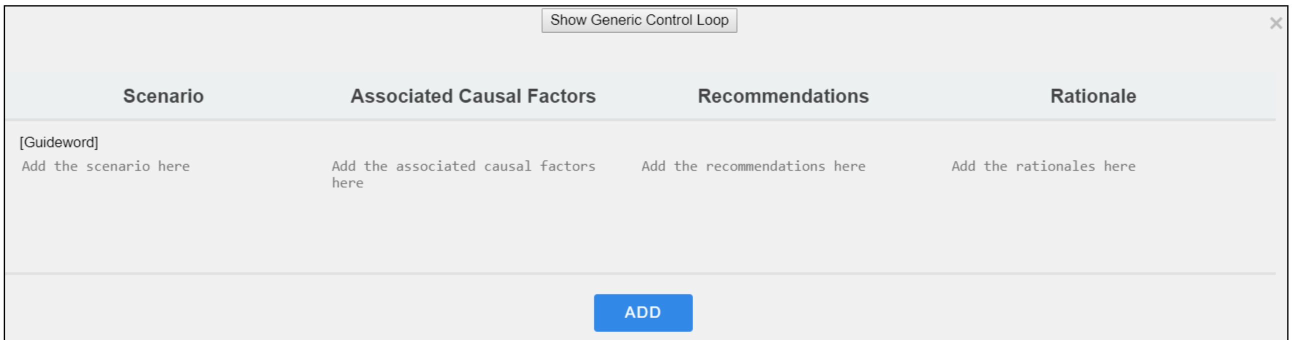
Right side: Unsafe control action provided or safe control action required but not provided ▾

Scenario	Associated Causal Factors	Recommendations	Rationale	Include?
[Control input or external information wrong or missing] Value of Glucose Level is missing.	Failure in the communication between Insulin Pump and the external system.	The communication between Insulin Pump and external system must be improved.	This external system are out of the scope of the system under analysis.	☐
[Inadequate Control Algorithm] An incorrect algorithm was designed	Algorithm wrong or incomplete or lack of knowledge of the system.	The algorithm must be revised and tested after each change to minimize errors.	Simulations of the system can help to validate the algorithm.	☐
[Feedback Delays] Feedback delays to reach the Insulin Pump	Limitations in the communication protocol or problem in the communication mean (Wire or Wireless)	Communication between Insulin Pump and Insulin Pump must be improved.	Alternative communication means can be considered.	☐

ADD

Figure 7: Selecting a generic set of scenarios, associated causal factors, recommendations and rationale using WebSTAMP. The row where the column "Include" is checked will be included in the analysis' output

For instance, for the guide question of Figure 7 ("What are the causal factors that make the pumping insulin to be provided by the insulin pump when the glucose level is below"), the guidewords to be created are related to the information on the right side. For example, the information types include feedback data, communication data, process variables, and the algorithm's code. For the algorithm's code, the analyst can determine that the critical phases of the information lifecycle are generation and storage, and two suggested security related guidewords can be the unauthorized installation of the executable code and unauthorized change of the executable code. For data feedback, the analyst can determine that the critical phase is communication, and two suggested security related guidewords can compromise integrity of data feedback and information disclosure of data feedback. The identification of the guidewords depends on the expertise of the analyst on the system's domain.



Show Generic Control Loop

Scenario	Associated Causal Factors	Recommendations	Rationale
[Guideword] Add the scenario here	Add the associated causal factors here	Add the recommendations here	Add the rationales here

ADD

Figure 8: Adding a new set of scenarios, associated causal factors, recommendations and rationale using WebSTAMP

In the case of the analyst opts to choose a set of generic scenarios, it is necessary to select what scenario(s) will be considered. The scenarios provided by the tool are as generic as possible to fit in different domains and different systems. The analyst must evaluate if the generic scenario fits in the system under analysis and tailor it, if necessary (FR03).

6. CONCLUDING REMARKS

Using WebSTAMP, we managed to obtain a complete analysis of Glucose Monitoring and Insulin Pumping System, considering the restrictions and assumptions of the Functional Control Structure, in an automatic manner (using rules and formalisms) and in a systematic way (outputs are systematically considered inputs). We met the functional requirements 02 and 03.

We verified if WebSTAMP provides systematization, automation and completeness in the analysis of other systems. We analysed the following systems: train door (Thomas, 2013), chemical reactor (Young & Porada, 2017) (for safety only) and level crossing (an intersection of railway line and road on the same level). The results are similar to those found for Glucose Monitoring and Insulin Pumping System. The results indicate that safety/security analysts can benefit from the support provided by WebSTAMP when organizing their analysis work. WebSTAMP helps in reducing the chances of missing hazardous control actions or neglecting combinations of values. We also found that WebSTAMP assists analysts to understand STPA and STPA-Sec in a more integrated way – the "rigour" of the step-by-step proposed by the tool helps the analyst to complete an analysis as a whole in a more traceable and guided way.

As for recommendations, we intend to extend the support for STPA-Sec, improving the existing features (expanded control flaws and information lifecycle) and adding new features, such as classifying hazardous scenarios using D4 Matrix and support to wargaming. Currently, we are developing a GUI to draw the control structure.

As future work, we intend to extend the tool to a STAMP platform, including other techniques based on STAMP, such as Causal Analysis based on STAMP (CAST) (Leveson, 2011) and leading indicators (Leveson, 2015). In the near future, we intend to make the source-code of WebSTAMP available using an open-source license. We also plan to extend WebSTAMP to be a fully collaborative platform. The motivation is to provide a way for the safety and security communities to contribute and allow analysts and organizations to use STAMP and its analysis techniques.

ACKNOWLEDGEMENTS

The first author was supported by the Coordination for the Improvement of Higher Education Personnel (CAPES). The fourth author was supported by the Swedish Governmental Agency for Innovation Systems under grant number NFFP7-04890. CNPq under grant number Universal 01/2016 403921/2016-3 and CISB SAAB 04/2016 supported the fifth author.

REFERENCES

Abdulkhaleq, A., & Wagner, S. (2015). XSTAMPP: an eXtensible STAMP platform as tool support for safety engineering.

Bean, M. (2015). Laravel 5 essentials. Packt Publishing Ltd.

Becker, C., & Van Eikema Hommes, Q. (2014). Transportation systems safety hazard analysis tool (SafetyHAT) user guide (version 1.0) (No. DOT-VNTSC-14-01). John A. Volpe National Transportation Systems Center (US).

Faulconbridge, R. I., & Ryan, M. J. (2014). Systems engineering practice. Argos Press Pty Limited.

Gurgel, D. L., Hirata, C. M., & Bezerra, J. D. M. (2015, September). A rule-based approach for safety analysis using STAMP/STPA. In Digital Avionics Systems Conference (DASC), 2015 IEEE/AIAA 34th (pp. 7B2-1). IEEE.

Information-Technology Promotion Agency - IPA. (2018). STAMP Workbench. Retrieved from https://www.ipa.go.jp/english/sec/reports/20180330.html

Leveson, N. (2011). Engineering a safer world: Systems thinking applied to safety. MIT Press.

Leveson, N. (2015). A systems approach to risk management through leading safety indicators. Reliability Engineering & System Safety, 136, 17-34.

Leveson, N., & Thomas, J. (2013). An STPA primer. Cambridge, MA.

McAffer, J., Lemieux, J. M., & Aniszczyk, C. (2010). Eclipse rich client platform. Addison-Wesley Professional.

MySQL, A. B. (2000). MySQL Documentation. Retrieved from http://www.mysql.com/doc.

Nixon, R. (2012). Learning PHP, MySQL, JavaScript, and CSS: A step-by-step guide to creating dynamic websites. " O'Reilly Media, Inc.".

Steinberg, D., Budinsky, F., Merks, E., & Paternostro, M. (2008). EMF: eclipse modeling framework. Pearson Education.

Suo, D. (2016). Tool-assisted hazard analysis and requirement generation based on STPA (Doctoral dissertation, Massachusetts Institute of Technology). Retrieved from https://dspace.mit.edu/handle/1721.1/105628#files-area

Thomas IV, J. P. (2013). Extending and automating a systems-theoretic hazard analysis for requirements generation and analysis (Doctoral dissertation, Massachusetts Institute of Technology).

Von Bertalanffy, L. (1968). General system theory. New York, 41973(1968), 40.

Young, W., & Porada, R. (2017). System-Theoretic Process Analysis for Security (STPA-SEC): Cyber Security and STPA. In: 2017 STAMP Workshop. Retrieved from http://psas.scripts.mit.edu/home/2017-stamp-presentations/.

Young, W., & Leveson, N. G. (2014). An integrated approach to safety and security based on systems theory. Communications of the ACM, 57(2), 31-35.