2018

System Theoretic Process Analysis: a literature survey on the approaches used for improving the safety in complex systems

Saulo Rodrigues e Silva
Universidade do Minho, saulo.r.silva@inesctec.pt

System Theoretic Process Analysis: a literature survey on the approaches used for improving the safety in complex systems

Saulo Rodrigues e Silva, Universidade do Minho, Portugal saulo.r.silva@inesctec.pt*

Abstract. Computer systems are becoming increasingly complex, specially interactive software systems, namely software user interfaces. The scientific community relies on different methods to assess their safety. This article provides an updated literature survey on hazard analysis approaches used to improve the safety of complex systems. To support the survey, we conceptualise complex systems, highlighting the challenge in terms of assessing their safety. We provide a brief overview on the approaches historically available to tackle issues in those systems, along with their most common methods. Finally, the article focuses in one method of a non-traditional approach, which is described in more details, along with some of its extensions, which seeks to improve the hazard analysis in complex systems.

Keywords: Complex systems; High-assurance systems; Design requirements; Hazard analysis methods.

1 Introduction

Critical interactive systems are those the human operator interacts with. Failures in such systems have the potential to cause undesired or unplanned events, which Leveson [2011] characterises as accidents, with the potential for causing losses of different nature, such as social (e.g., threat or loss of human life), environmental (e.g., damage to the environment), financial (e.g. property damage), etc. Some examples of critical systems can be found in today's medical, aerospace, nuclear, financial and energy domains. To avoid undesired events, those systems need to provide safe operation. Although in general the design of critical systems demands full attention from system engineers and designers, special attention should be given to the design of their user interfaces, as problems in the way the human operator interacts with the system have the potential to lead the system to accidents with incalculable costs. This is because the human factor creates additional levels of complexity in the system design.

Due to the increasingly high level of automation in such systems (e.g., a high technological aircraft cockpit, sometimes also referred to as "glass cockpit"), the role of system operators (e.g., pilot) has changed. A previous condition in which the operator manually provides actions to change themselves the system operation modes (e.g., pilot manipulating the input controls required to change the aircraft's altitude) is being shifted to an abstract concept, in which the operator monitors several complex systems included in the automation (which acts as user interface for the system) that command the system's operation modes themselves. In the case of an aircraft, systems such as the Flight Management System (FMS) operate in discrete modes, meaning that as an outcome of the interaction, the automation either performs an action or it does not [Thimbleby, 2010]. As part of his/her work of piloting the aircraft, the pilot has the (limited) capacity to control the mode selection of such systems.

Several approaches have been historically used for tackling issues in critical interactive systems, to avoid problems that might arise when the human operator is monitoring the automation (e.g., *automation surprises*, where the system might react differently than the operator expects, causing surprise, confusion and uncertainty). This article provides an overview of such approaches, alongside their most common methods. It covers some historical definitions that intents to create the required context for explaining the traditional approaches and its methods. Finally, the article focuses in one method of a non-traditional approach, which is described in more details, as well as some of its extensions, aiming to improve the hazard analysis in complex systems. The evaluation procedure for the extensions is based on the analysis of works published in previous conferences about the surveyed method, as well as the search for prominent related works indexed in relevant scientific databases.

ISSN 2183-489X

The article's main contribution is providing an updated literature revision on the Hazard Analysis (HA) methods, with focus on the safety of complex, software based systems. For accomplishing that, the article is organised as follows: Section 2 provides a summary of Hazard Analysis approaches. Section 2.1 provides information on the traditional methods used for tackling problems in systems for more than 50 years. Section 2.2 provides information on the system theoretic approach, detailing the accident model that Systems Theoretical Process Analysis (STPA) is based on, as well as the method itself. The same section also provides an overview of the different extensions to the method, created for improving the detection of hazards in different kinds of systems. Section 3 concludes the article.

2 Hazard Analysis

For Rasmussen [1981], the potential for producing some undesired consequence is commonly referred to as "hazard". In his historical context, Hazard Analysis (HA) and Reliability methods were synonymous. Leveson [2011] defines them as techniques for identifying system failure modes, estimating their probability of occurrence so they can be eliminated or controlled during design or operations before damage occurs. Failure modes in this context are defined as the ways in which something might fail. Hazard Analysis methods have been used for many years [Song, 2012] and in general are based on a traditional approach, whose main focus was frequently related with examining hazards caused by malfunctioning/failures in single components (hardware), present in systems.

As the complexity of the systems increase, a new unity of analysis is necessary. Systems-theoretic methods are used for examining and understanding complex socio-technical systems. They consider factors such as critical design errors or requirements flaws, which have the potential for causing an accident even if there is no component failure related with the flaws.

In order to make this difference between the traditional and non-traditional HA approaches clear, we can think about the existing difference between reliability and safety. In terms of system components, generally they can be considered reliable when their particular behaviour is verified, in other words, it complies with the design requirements and performs as designed, with no flaws or inconsistencies (errors) in their behaviour. In the perspective of Systems Theory [Zadeh, 1962], however, although a component's implementation might be correct and reliable, its safety can only be determined by the relationship between the component and the other system components [Leveson, 2011], that is, it can be unsafe even when performing as designed. Interactions problems that may arise during human interactions with the system, specially software systems, might present problems with those characteristics. This is because, although in general the specification of computer programs influences its safety (if some function in the software specification is not correct, the software behaviour is likely to present errors when that particular function is used), software based user interfaces might even have correct specification and yet behave unsafely, due to the complexity of the human operators's interactions with the system. In terms of HA, the old methods are not enough to guarantee the safety of computer systems because they do not consider the complexity of the human operators's interactions with the system (e.g., one could also consider the analysis of user interface's usability and safety properties for ensuring that the behaviour of the user interface is compliant with properties, capturing best practice in human-machine interface design).

We describe hereafter the two approaches and the analysis methods based on their assumptions.

2.1 Traditional Hazard Analysis approach

Traditional methods for hazard analysis base their assumptions on the idea that hazards are the result of a sequence (chain) of events caused by the loss of functions in the system's components. Additionally, the effects of such loss of functions are propagated throughout the system. According to Leveson [Leveson, 2004], these methodologies may work well for losses caused by failures of physical components and for relatively simple systems. Henceforth we briefly describe some methods which implement the traditional Hazard Analysis approach.

FMEA Failure Modes and Effects Analysis [Bowles and Peláez, 1995; NASA, 1966] is a bottom-up method developed for the U.S. military and used for identifying failure modes and their causes, as well as for carrying out corrective actions to address them. The analysis, normalised in the Standard BS EN 60812:2006 [EN, 2006], begins by identifying the system's components and their failure modes.

ISSN 2183-489X

Then, the potential causes and effects of each failure mode are investigated. The outcome of the analysis is a set of scenarios triggered by a failure, namely, a safe and unsafe cause-effect chain of events. Both set of scenarios are analysed in similar levels of detail looking for unsafe scenarios, which in case of analysing non-hazardous failures, consumes time and does not add value to the analysis. The scenarios with highest probability of successful operation are selected for improving the system.

FMECA Failure Modes, Effects and Criticality Analysis [NASA, 1966; Standard, 1980] is a bottom-up FMEA extension which in addition assigns a criticality ranking for each failure mode, which is based on Criticality Analysis (CA), a method for determining the significance of each potential failure for each component in the system's design, based on a failure rate and a severity ranking [Lipol and Haq, 2011].

SFMEA Software Failure Modes and Effects Analysis [Reifer, 1979] is another FMEA extension tailored for analysing software, by analysing how a software component (e.g., a variable, a function) might fail and how the detected failure propagates throughout the system and leads to a hazard. Because software components do not fail in the same manner as hardware components, the analysis is conducted by replacing the latter with the former. The analysis can be conducted at design level or code level and requires the detailed design documentation of the software, or the availability of the source code [Stadler and Seidl, 2013].

SFMECA Software Failure Modes, Effects, and Criticality Analysis [Dehlinger and Lutz, 2004; Lutz and Shaw, 1999] is a bottom-up method based on hardware FMECA, tailored for finding potential software problems. The method can be used both in the design and requirements phase for tracking the propagation of anomalies from the causes (failure modes) effects in the software components to the system effects. SFMECA follows a similar analysis procedure such as SFMEA, which in summary starts by breaking the software into logical components (e.g., functions, variables), to predict the potential failure mode for each component on the overall system.

HAZOP Hazards and Operability Analysis [Lawley, 1974] is an analysis technique originally designed by the Imperial Chemical Industries in the United Kingdom in the early 1960s, to help a multidisciplinary team looking for hazards in chemical processes, later normalised in the standard IEC 61882 [IEC, 2001]. The process of analysis begins by collecting all relevant system information. In consonance with the other approaches which rely on system information for the analysis, it is important that the system representation selected for documenting the system (e.g., a data-flow diagram in computer industry, or a piping and instrumentation diagram in process industry) is updated and leaves no ambiguity in the understanding of the system. "Attributes" are identified in the system representation (e.g., in a data flow diagram, attributes can be related with the rate of flow of data). The method uses a list of guide-words for identifying the intended behaviour and also stimulate the discussions among the team members about how the system attributes can deviate from normal operation. Thomas IV [2013] states that the simplicity of the method contrasts with the highly detailed system model required for identifying the parameters and apply the guideline words, which limits the application of the method to system models in advanced stages of design, as well as the number of potential solutions, since many important decisions have been made and solutions are usually related with minor changes such as patches or protection systems added to an existing design.

FTA Fault Tree Analysis [Haasl et al., 1981] is a top-down method used for investigating the causes of a high-level event, defined as the undesired system state or condition to be investigated. The method performs decomposition of the high-level event for creating lower-level events, which must occur to cause the higher-level event; the decomposition uses Boolean logic for identifying the possible combinations (lower-level) of basic causes for the undesired (higher-level) system state or condition. Since FTA works backwards for exploring and identifying ways in which the high-level event might emerge, a graphical hierarchical tree made of logical operators (e.g., *AND, OR*) is used in the analysis in order to demonstrate the relationship between the first event and its effect on the higher levels of the system; at each level of the tree, possible errors or failures that can trigger errors or failures in the higher-level are identified.

SFTA Software Fault Tree Analysis [Leveson and Harvey, 1983] is a top-down hazard analysis method based on FTA, which applies the same methodology used for analysing hardware components, to the analysis of the software counterparts (e.g., functions). The analysis begins with an identified hazard, which must be defined using a method such as Preliminary Hazard Analysis (PHA). The starting point for the analysis is the software or functions responsible for controlling the system when the hazard occurred. The goal is to find specific failure modes or scenarios which can led to specific safety failures, or even demonstrate that the software is free of hazardous behaviour. As SFTA is based on FTA, the analysis uses a fault tree for documenting the analysis. At the top of the tree is the root event to be analysed. The logical operators *AND* and *OR* are used to describe the necessary preconditions of the next level of the tree.

ETA Event Tree Analysis [Kenarangui, 1991] is a bottom-up method, which uses a visual tree structure, known as Event Tree (ET) for identifying all sequences of occurring events following an initial event. ETA uses similar boolean logic as FTA for identifying whether the derived events from the root event in the tree (main event) are sufficiently controlled by safety procedures implemented in the system design, or can develop hazardous behaviour. Although dependent on an initial event, the method does not provide a way to identify it, therefore other methods are required for complementing the analysis. Other limitations are related with the method's assumption that the initial event occurred, providing no preventive measures to avoid it. Also the method considers the results of human actions as binary (success, failure), according to critics, a dangerous simplification that hides a great number of possible scenarios [Thomas IV, 2013].

PHA Preliminary Hazard Analysis [Ericson, 2005; Ericson et al., 2015] is a System Safety technique commonly used for identifying potential hazards at very early design level, when detailed design information is not available.

2.2 Systems theoretic Hazard Analysis approach

Currently there is only one hazard analysis method which is Systems-theoretic based, referred to as System Theoretic Process Analysis (STPA) [Leveson and Thomas, 2013]. This is a deductive (top-down) method, based on the STAMP accident model, both well funded on Systems and Control theory rather than reliability theory. The method targets failure of more complex systems, such as software intensive systems (systems which intensely make use of software) or socio-technical systems (systems which include human operators) [Leveson, 2004]. Although STPA is based on a accident model, the method can be applied early in the development phase, before an accident happens. It encourages the analyst exploring classes of aspects such as the human interaction related with complex, software intense, socio-technical systems.

Even though the goal of the article is not to meticulously discuss the method (the reader will find satisfactory information in [Leveson, 2004]'s book and practical examples of the method's application in [Leveson and Thomas, 2013]), we present STAMP and STPA briefly.

2.2.1 STAMP accident model. Systems-Theoretic Accident Model and Processes is a accident model that intends to understand the rules that govern the causes of an accident. The model is established on the top of three pillars: safety constraints, hierarchical safety structures and process models [Leveson, 2004].

According to STAMP, events leading to losses (i.e., to an accident) happen due to insufficient enforcement in the system's safety constraints. More traditional accident models, such as Domino Theory [Heinrich et al., 1941] or the Swiss Cheese Model [Reason, 1990], emphasise the prevention of failures in system's components to impede future accidents. STAMP shifts the unity of analysis from prevention of component failures to enforcing the system's behavioural safety constraints [Leveson, 2011].

Figure 1 depicts a basic conceptual high level control structure, which is the representation/model of the system under analysis, containing the following elements:

– **Controller** - uses its information about the current state of the controlled process for determining if control actions are needed, and in that case, sends control signals to the controlled process. According to Leveson and Thomas [2013], an important issue on the design of an effective safety

ISSN 2183-489X

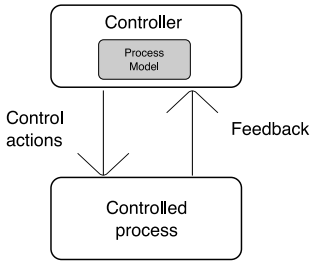

Fig. 1. A basic conceptual form of hierarchical safety control structure. Adapted from [Leveson, 2011].

control structure is providing the feedback and inputs required for keeping the automated controller's model consistent with the actual state of the controlled process;
- **Process model** - is a representation the controller has of the controlled process, which includes information on how the controlled process operates (e.g., about sensors, actuators, displays, system variables, control actions) to affect the current state of the controlled process [Leveson and Thomas, 2013];
- **Controlled process** - represents the automation being controlled by the controller (human operator, automated controller, or both). Responds to the control actions issued by the controller and provides feedback about the result of the operation;
- **Control action(s)** - the action(s) issued by the controller (which can be human or automated) towards the controlled process (which can be the automated controller or the controlled process) to affect the state of the system;
- **Feedback** - notifies the higher level (automated controller or human operator) about the result of the control actions in the lower level's controlled process.

According to Control Theory, the control structure's components are kept in state of dynamic equilibrium by feedback loops of information and control [Leveson, 2011].

2.2.2 The STPA method. STPA, short for System Theoretic Process Analysis [Leveson, 2011], is a Hazard Analysis (HA) methodology based on the STAMP accident model, which considers the dangers that can arise during the operational phase of the system for establishing causal relations with problems.

The method can consider aspects of the environment where the system is inserted in the analysis, along with the human operators and their interactions while using the system. For this reason, working with a multidisciplinary team (e.g., designers, system engineers and human factors specialists, along with other multidisciplinary team members) during all the phases of the analysis is highly recommended.

One of the main contributions of STPA is providing a systematic way and clear guidance for identifying scenarios that could lead to hazardous system states involving unsafe interactions among components, even when they are operating according to their requirements. Leveson and Thomas [2013] states that hazardous components or unsafe interactions can be controlled through design.

In this method, design information from the system whose unsafe behaviour needs to be identified is provided as input. The design information might include:

- The system boundary;
- System and system components responsibilities;
- Knowledge the system components must have about the system;
- The system goals.

A working system, rather than a system design, may be used as source of information for the analysis. In such case, the responsibilities of the components comprising the system's control structure must be defined. Figure 2 presents the overall summary of the STPA analysis process. At the end of every engineering activity throughout the groups, output results are used as inputs for the next activity to be performed in the analysis.

Leveson categorises the engineering activities required for using the STPA method in three main groups: *fundamentals*, *step 1* and *step 2*.

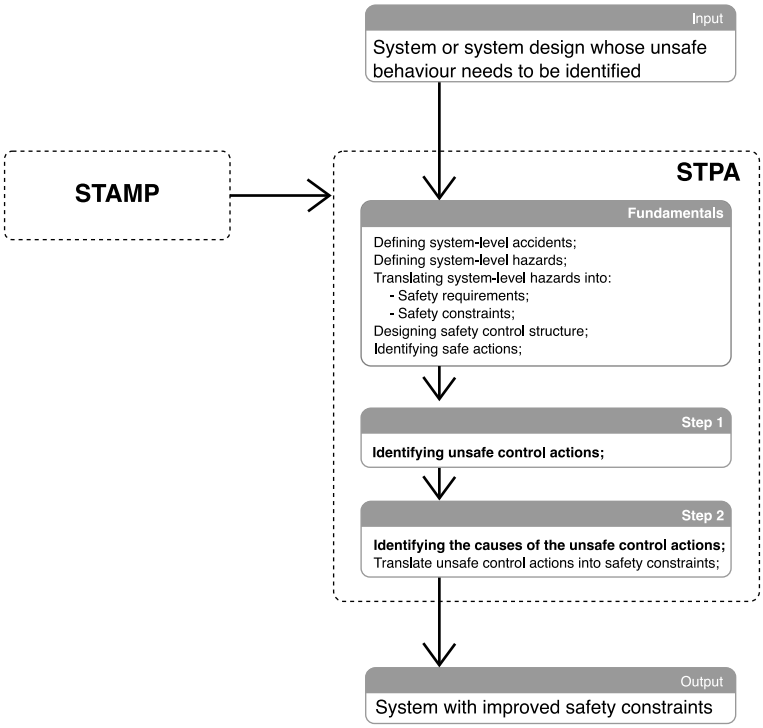

Fig. 2. STPA summary.

The STPA fundamentals

A number of early activities need to be performed in the early phase of the method's application, with the intention of collecting information about the system to be analysed, the safe behaviour to be enforced and the hazardous behaviour to be avoided [Leveson, 2004, 2011]. We summarise them hereafter.

Defining accidents. The first STPA activity necessary for conducting the analysis is related with agreeing/defining what an accident is. For instance, for an aircraft operation, monetary losses might not make sense. Instead, the definition of what an accident is in aviation domain must consider injury of persons. This definition is implicit to different domains and need to be discussed with the stakeholders and custumers, as there are different kinds of losses to consider. Government and specific agencies (e.g., Federal Aviation Administration - FAA, Food and Drug Administration - FDA, European Space Agency - ESA, etc) might also be consulted, as they might have their own definitions of accident. One important aspect here, however, is that the definition of accident incorporates the system boundaries, derived from the system documentation, which must consider the difference between the control structure and the physical structure of the system.

Leveson, for instance, adopts a general accident definition which includes general (material) losses, as well as human injury, as follows:

– *"Accident is an undesired or unplanned event that results in a loss, including loss of human life or human injury, property damage, environmental pollution, mission loss, etc".*

Defining hazards The second STPA activity requires that practitioners agree on/define the system-hazards. For Leveson, a general hazards definition is as follow:

– *"Hazard is a system state or set of conditions that, together with a particular set of worst case environment conditions, will lead to an accident (loss)".*

Leveson reminds us that the hazard should be within the system boundaries over which we have control [Leveson and Thomas, 2013], because the result of the analysis will be able to affect the design. For instance, for an aircraft a hazard can not be weather, because we can not control the weather. Instead, a hazard can be to fly in areas of bad weather.

To assist the STPA practitioners in defining their own set of system-level hazards, the system documentation might contain useful information such as the system's goals. The practitioner can identify a small set of high-level system hazard at first, and refine them as the method evolves [Leveson and Thomas, 2013]. Practitioners can identify several hazards and work with them concurrently, as hazard scenarios identified in the analysis can be related with the same state or set of conditions.

Translating system-level hazards into requirements and constraints. The identified system-level hazards can be translated into design requirements (i.e., a specification of what should be implemented, descriptions of how the system should behave [Wiegers and Beatty, 2013]) and the first safety constraints (namely, the safeguards that prevent the system from leading to accidents). According to [Masci et al., 2017], safety requirements can be formulated in natural language as the logical negation of the corresponding system-level hazards. For instance, if the system-level hazard "Pilot engages autopilot too early" is identified, a safety-requirement can therefore be defined as *Pilot must not engage autopilot too early*. Leveson states that this is a simple yet very important process, as engineers and system designers need that information in their standard engineering processes [Leveson and Thomas, 2013].

Designing the control structure. The system engineering activities conducted so far, identifying the system-level accidents, hazards and safety requirements/constraints are common activities to all safety engineering approaches, independent of which accident model or hazard analysis method is used [Leveson and Thomas, 2013].

The first specific STPA effort that needs to be done is generating the control structure (sometimes also referred to as safety control structure, hierarchical control structure or functional control structure). The control structure is a relevant component in STPA as it allows illustrating not only the main elements that participate in the analysis, but it also illustrates their interrelations, i.e., what is the controller, what is the controlled process, what are the control actions and the feedback necessary for keeping the system working. Leveson states that using a control structure diagram is not standard in STPA, as the control structure might be represented using non graphical notations. However, the graphical diagram is a documentation effort needed for performing STPA [Leveson and Thomas, 2013], as the method provides a graphical diagram with guide-words registered for causal analysis.

After the control structure is designed, safe control actions required for keeping the system out of a hazardous scenery can be identified, as they are issued by controllers towards the controlled process.

Identifying control actions. The control actions required for keeping the system out of a hazardous scenery are potentially safe. The system-level requirements and constraints identified previously are an important source of information about the control actions. Another source is the model of the controlled process (system design), because it presents information about the boundaries of the system, which the STPA practitioner can use to help determine what control actions are needed. The model of the controlled process contains relevant information about the system, such as:

a) the variables and states that are going to be monitored and/or changed;
b) the control actions that must be provided by the controllers (automated, human or both) to operate the controlled process adequately. They are functions which monitor and/or change the system's variables and/or states. The control actions are also relevant for helping to identify the hazards that we want to avoid (unsafe control actions) and the safety control loop problems that may lead to the hazardous problems.

Identifying the unsafe control actions (Step 1)

After performing the *fundamentals*, the next activity is *Step 1*, which consists in identifying unsafe control actions (UCA), namely, control actions that can lead the system to a hazardous behaviour. Usually UCA are result of inadequate enforcement of the system's safety constraints. For each CA, the conditions under which the control action may become inadequate are identified by the four general categories [Leveson, 2011], as follows:

1. A control action required for safety is not provided or is not followed. E.g., a pilot does not engage the vertical mode of the autopilot;

ISSN 2183-489X

2. An unsafe control action is provided that leads to a hazard. E.g., when engaging vertical mode, a pilot set wrong parameters for vertical speed.
3. A potentially safe control action is provided too late, too early, or out of sequence. E.g., when in descend mode, the pilot deploys flaps before airspeed is within safe values.
4. A safe control action required for safety is stopped too soon, or applied too long (for continuous or non discrete control action). E.g., the pilot pushes the aircraft side stick forward too long (which causes the aircraft to descend steeply).

Identifying UCA in STPA starts by analysing the previously identified control actions, using these categories.

Identifying the causes of the unsafe control actions (Step 2)

The objective of *Step 2* is to determine how the UCA identified in *Step 1* were caused. Although Leveson states that this step is not always performed in STPA analysis, due to the need of a more experienced analyst, she also states that it is in this phase that additional safety requirements are identified, as well as information is generated for assisting the designers in eliminating or mitigating potential hazardous behaviours [Leveson and Thomas, 2013].

STPA provides a series of guide-phrases for helping the practitioner to identify the UCA causes not only by looking into failures or inadequate operation of individual components in the control structure, but also by finding scenarios and combinations of problems that could lead to unsafe control and hazards. Figure 3 presents a general representation of a control structure, along with the guide-phrases recommended by the method for determining the causes of the previously identified UCA.

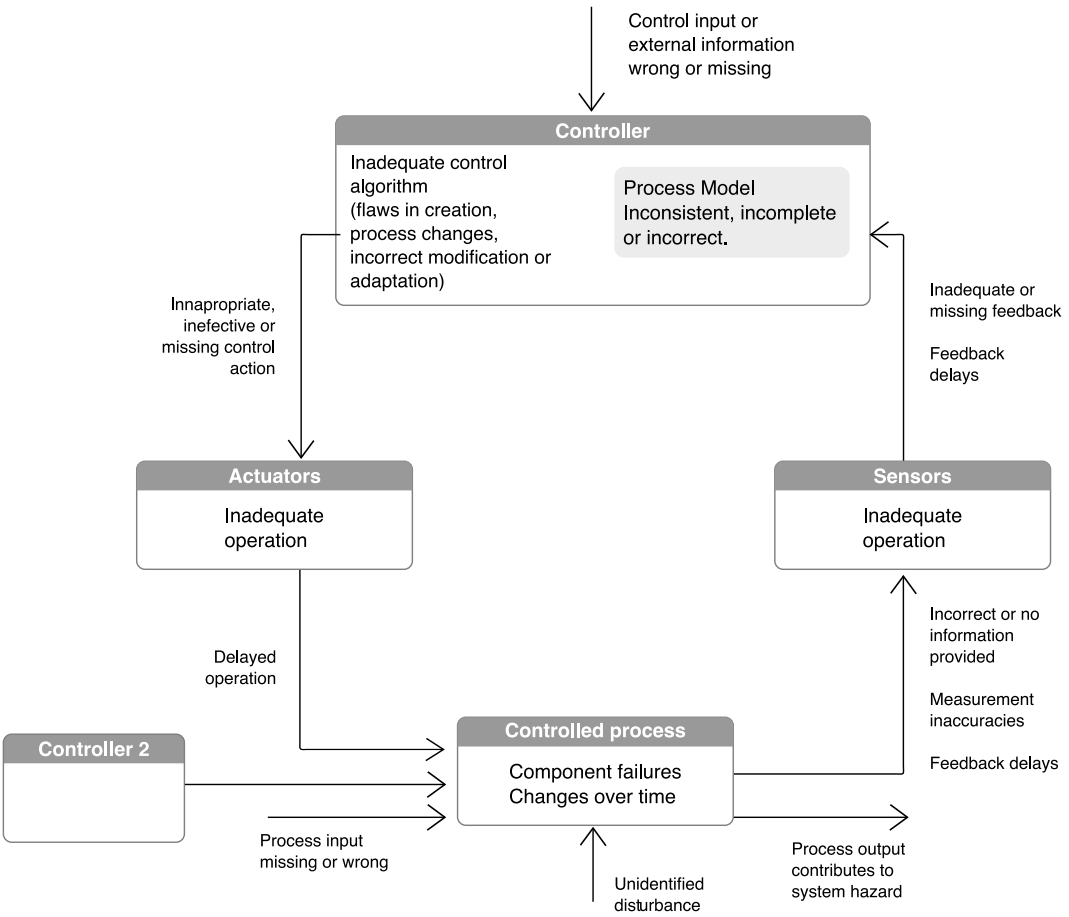

Fig. 3. Problems that can arise from the control structure leading to hazards according to Leveson [2011].

ISSN 2183-489X

The causes for unsafe control actions, according to Figure 3, can be diverse and depends on which part of the control structure is being analysed, namely, the controller, the controlled process, the control action or feedback part. E.g., if a controller is being analysed, one assumption that can be made is that it takes too long to activate (provide a control action) an actuator, or commands its activation with the wrong control action. In general, these problems are divided in the following categories:

a) Unsafe inputs - every controller in the hierarchical control structure is itself controlled by an upper level controller; in this process, input information or control actions coming from the upper controller may be missing or wrong, and compromise operations.
b) Unsafe control algorithms - these are procedures designed for hardware controllers (e.g., engaging a hardware actuator) and procedures that human operators use (e.g., changing the system state). The procedures of the human operator are the ones they must follow to accomplish a goal in the system. They are created/affected by training and by feedback and experimentation over time [Leveson, 2011]. A hazardous control action will be caused, for instance, if the process changes and the algorithm that controls it was not updated, or if the control algorithm was inadequately designed (receives the correct command from the controller's model of controlled process but cannot execute it correctly).
c) Inconsistent, incomplete or incorrect model of controlled process - automated controller(s) must have a model of the controlled processes that is consistent with the actual process or incorrect commands (control actions) might be sent causing a hazard.
d) Incorrect information provided, measurement inaccuracies - the controlled process provides information for sensors keeping the system operating safely; incorrect or inaccurate information might cause hazards in the system.
e) Issues with feedback - sensor(s) might create inadequate feedback, provide feedback with delays or not provide (missing) any feedback to the controller, and consequently cause hazards in the system;

Note that in the particular case of human controllers, the context and the environment where the activity is developed plays a relevant role in the decision making process. STPA recommends observing three categories of hazards issues related with human operators Masci et al. [2017]:

a) Issues with feedback - in this case, feedback created inadequately, provided with delays or not provided (missing) by the automated controller to the human operator might cause hazards in the system, e.g., once the controlled process sends information to the sensors, the former send feedback with delays to the automated controller, which propagates the information with delays to the operator;
b) Issues with mental model - the operator's understanding of how the system works is incorrect, e.g., operator's mental model of controlled process different from controller's model of controlled process;
c) Issues with external information - wrong or missing procedures described in user manuals, or wrong information communicated to the operator.

After the unsafe control actions and their causes are identified, practitioners can enforce the first system safety constraints identified in Step 1 and continue the analysis to identify the scenarios leading to the hazardous control actions that violate the safety constraints. In order to understand the scenarios that may lead to hazards, the practitioner can make use of tools that range from tables or figures, to modelling and analysis tools, which provide relevant information about the system under analysis.

2.2.3 STPA extensions. STPA's system representation is originally designed to fit the representation of different systems in domains ranging from aerospace and automotive [Castilho et al., 2018; Stringfellow et al., 2010], nuclear [Thomas et al., 2012], medical [Antoine, 2013], cyber security [Young, 2014] and energy [Rosewater and Williams, 2015]. Eventually though, STPA is being tailored to improve its capacity of analysis and hazards identification in a specific range of systems. This section introduces extensions to STPA, that improves the detection of hazards in hardware systems, software systems, as well as in the human operator analysis.

Stringfellow [2010] states that systems with flaws are operated in a state of high-risk. She proposed the *STAMP extension for Humans and Organisations using guideWords (SHOW)* for analysing how organisational factors, e.g., policies, might contribute to high-risk operations and affect safety in socio-technical aerospace systems. The method uses STPA's structured approach for identifying design flaws, diagnosing policies that are resistant to pressures and contribute to high-risk operations, as well as allowing system decision-makers to predict how proposed or current policies will affect safety. A new set of guide-words is used in the STPA risk analysis, tailored for the identification and analysis of human and organisational factors and their role in accidents. The guide words are built upon individual error and organisational error taxonomies, identified with Grounded Theory [Robson and McCartan, 2016].

Early Warning Sign Analysis (EWaSAP) [Dokas et al., 2013] is a STPA extension which adds two types of steps into the typical STPA control structure. In STPA the control structure provides, among other things, safety constraints for enforcing the system safety on the levels below and feedback indicating how the system safety was enforced. The first type of step added by the extension aims to establish a form of cooperation between the system being analysed and the system surrounding it, and for exchanging information about safety that might constitute early signs of flaws. The second type of step aims at analysing the early signs indicating the presence of flaws in the control structure of the system in focus, or the violation of its design assumptions that can contribute to losses, during the operational phase of the system. The method also introduces the notion of *awareness actions*, responsible for enforcing the transmission of early warning signs to other controllers within or outside the system, when a controller identifies factors contributing to losses in the system. Typical awareness actions transmits i) *all clear*, ii) *warnings*, iii) *alerts* and iv) *algedonic* signals (a special transmission for informing the controller at highest levels about a serious detected condition).

Thomas IV [2013]'s approach for extending STPA provides a rigorous procedure that systematically helps to identify hazardous control actions based on the combination of process model context variables and context values. The *context* is the condition in the system and environment that make action (or inaction) hazardous. A combinatorial testing algorithm is also introduced, responsible for performing a STPA analysis using the context variables and values to produce the context tables. The context table gathers those informations and allows other types of risk identification, based on the context of the control action. The approach also formalised STPA and STAMP using logical and mathematical structures that can be used by automated tools, and a method for using the results of the hazard analysis for generating formal safety requirements.

As Yang [2014] remarks the importance of software requirements as one of the most important elements during Software Safety Test Process, he proposes a framework of software safety testing based on Systems Theory, for improving the test analysis over the test approaches based on single components failure methods. The *Software Safety Test Requirements Elicitation based on STPA* approach uses the causal scenarios identified in the STPA's causal analysis and the safety requirements identified after the STPA's risk analysis, for generating the safety test requirements. Next, safety test requirements are used for designing specific safety test cases, implementing the safety test that are used for verifying the realisation of the system software requirements.

Leveson et al. [2014] also extends the STPA human controller, integrating two new causal factor categories based on principles from ecological psychology and cognitive models. The extension updates two parts of the STPA control structure. The detection of flawed feedback is included for verifying the operator's ability to understand feedback correctly. In the same work an action component is included, responsible for identifying actions that are appropriate for manipulating controls of the user interface.

STPA for Software-Intensive Systems (STPA SwISs) is an approach by [Abdulkhaleq et al., 2016, 2015]. In short, it comprehends mainly tree parts: i) deriving software safety requirements at the system level using STPA, ii) creating a model of the safe behaviour from the STPA results using a UML state-chart notation and iii) verifying the functional requirements on the model of the software, at the design and implementation levels, using formal specification.

Masci et al. [2017] proposes an extension that refines STPA's standard causal factors categories with 16 new categories, specialised in improving the analysis of user interfaces in medical devices, in terms of safety and usability. The categories are used in different parts of the STPA control structure, namely, the Feedback component, the human operator's Mental model component and in the External information component. The categories are derived from usability heuristics user

ISSN 2183-489X

interface design guidelines, defined in medical device usability standards (ANSI/AAMI HE75 and ISO 62366-1).

France [2017] proposed STPA-Engineering for Humans, an extension for improving how the method analyses human errors. In order to accomplish that, the system model/representation is updated, adopting a new model of the human operator, which relies on three new components. (1) Control Action Selection captures the human controller's goals and how the decisions are made based on the mental models. (2) Mental Model verifies the knowledge the user has of the system and environment for capturing flaws. (3) Mental Model Updates analyses the human experiences and expectations and how they can affect the system use. This provides a high-level characterisation of human information processing, which intends to improve the characterisation of the human operator's mental models. As a result, it helps to create more robust causal scenarios that explore problems related to the human operator, and allows designers and human experts to discuss how to consider the role of the human operator in the system under analysis. The approach is tested in a realistic case study from automotive industries.

3 Conclusion

The field of Human Computer Interaction (HCI) owes a lot to the traditional hazard analysis methods, for the development of extensions for analysing software based systems, as well as user interfaces of those systems, aiming to ensure the safety of interactive software systems. However, we are entering in a new era of computing systems. Emerging complexity of software systems used for operating systems that used to be simpler (e.g. highly technological "self-driving" cars and "glass cockpit's" airplanes) are providing new challenges in terms of assessing the interaction among the human operators and those systems.

By combining human factors considerations along with the rigour of the engineering discipline in the analysis, STPA provides a flexible framework for analysing complex interactive systems. The method is being tailored to accommodate different kinds of systems, this way extending the range and deepth of the analysis, including complex interactive systems. This work is theoretical, therefore future work includes moving to a realistic case study, that will provide us with a test-bench suitable for testing the approaches and present a comparison between them.

Acknowledgement. We would like to acknowledge Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for the support, as well as Dr. José Creissac, Dr. Paolo Masci, Dr. João Fernandes and Dr. Orlando Belo for the valuable insights.

Bibliography

Abdulkhaleq, A., Vöst, S., Wagner, S., and Thomas, J. (2016). An industrial case study on the evaluation of a safety engineering approach for software-intensive systems in the automotive domain.

Abdulkhaleq, A., Wagner, S., and Leveson, N. (2015). A comprehensive safety engineering approach for software-intensive systems based on STPA. *Procedia Engineering*, 128:2 – 11. Proceedings of the 3rd European STAMP Workshop 5-6 October 2015, Amsterdam.

Antoine, B. (2013). *Systems Theoretic Hazard Analysis (STPA) applied to the risk review of complex systems: an example from the medical device industry.* PhD thesis, Massachusetts Institute of Technology.

Bowles, J. B. and Peláez, C. E. (1995). Fuzzy logic prioritization of failures in a system failure mode, effects and criticality analysis. *Reliability Engineering & System Safety*, 50(2):203–213.

Castilho, D. S., Urbina, L. M., and de Andrade, D. (2018). Stpa for continuous controls: A flight testing study of aircraft crosswind takeoffs. *Safety Science*, 108:129–139.

Dehlinger, J. and Lutz, R. R. (2004). Software fault tree analysis for product lines. In *High assurance systems engineering, 2004. proceedings. eighth ieee international symposium on*, pages 12–21. IEEE.

Dokas, I. M., Feehan, J., and Imran, S. (2013). Ewasap: An early warning sign identification approach based on a systemic hazard analysis. *Safety science*, 58:11–26.

EN, B. (2006). 60812: 2006 analysis techniques for system reliability. procedure for failure mode and effects analysis (fmea).

Ericson, C. A. (2005). Event tree analysis. *Hazard Analysis Techniques for System Safety*, pages 223–234.

Ericson, C. A. et al. (2015). *Hazard analysis techniques for system safety.* John Wiley & Sons.

France, M. E. (2017). *Engineering for humans: a new extension to STPA*. PhD thesis, Massachusetts Institute of Technology.

Haasl, D. F., Roberts, N., Vesely, W., and Goldberg, F. (1981). Fault tree handbook. Technical report, Nuclear Regulatory Commission, Washington, DC (USA). Office of Nuclear Regulatory Research.

Heinrich, H. W. et al. (1941). Industrial accident prevention. a scientific approach. *Industrial Accident Prevention. A Scientific Approach.*, (Second Edition).

IEC, B. (2001). 61882: 2001: Hazard and operability studies (hazop studies). application guide. *British Standards Institute*.

Kenarangui, R. (1991). Event-tree analysis by fuzzy probability. *IEEE transactions on reliability*, 40(1):120–124.

Lawley, H. (1974). Operability studies and hazard analysis. *Chemical Engineering Progress*, 70(4):45–56.

Leveson, N. (2004). A new accident model for engineering safer systems. *Safety science*, 42(4):237–270.

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

Leveson, N. and Thomas, J. (2013). An stpa primer. *Cambridge, MA*.

Leveson, N. G. et al. (2014). *Extending the human controller methodology in systems-Theoretic Process Analysis (STPA)*. PhD thesis, Massachusetts Institute of Technology.

Leveson, N. G. and Harvey, P. R. (1983). Software fault tree analysis. *Journal of Systems and Software*, 3(2):173–181.

Lipol, L. S. and Haq, J. (2011). Risk analysis method: Fmea/fmeca in the organizations. *International Journal of Basic & Applied Sciences*, 11(5):74–82.

Lutz, R. R. and Shaw, H.-Y. (1999). Applying adaptive safety analysis techniques [for embedded software]. In *Software Reliability Engineering, 1999. Proceedings. 10th International Symposium on*, pages 42–49. IEEE.

Masci, P., Zhang, Y., Jones, P., and Campos, J. C. (2017). A hazard analysis method for systematic identification of safety requirements for user interface software in medical devices. In *15th International Conference on Software Engineering and Formal Methods (SEFM 2017)*, volume LNCS, volume 10469, Springer. Springer, Springer.

NASA, N. (1966). S. administration. procedure for failure mode, effects and criticality analysis (fmeca), rm 63tmp-22. *NASA, Tech. Rep.*

Rasmussen, N. C. (1981). Methods of hazard analysis and nuclear safety engineering. *Annals of the New York Academy of Sciences*, 365(1):20–36.

Reason, J. (1990). *Human error.* Cambridge university press.

Reifer, D. J. (1979). Software failure modes and effects analysis. *IEEE Transactions on reliability*, 28(3):247–249.

Robson, C. and McCartan, K. (2016). *Real world research.* John Wiley & Sons.

Rosewater, D. and Williams, A. (2015). Analyzing system safety in lithium-ion grid energy storage. *Journal of Power Sources*, 300:460–471.

Song, Y. (2012). *Applying system-theoretic accident model and processes (STAMP) to hazard analysis.* PhD thesis.

Stadler, J. J. and Seidl, N. J. (2013). Software failure modes and effects analysis. In *Reliability and Maintainability Symposium (RAMS), 2013 Proceedings-Annual*, pages 1–5. IEEE.

Standard, U. M. (1980). Mil-std-1629a. *Procedures for Performing a Failure Mode, Effect and Criticality Analysis, Department of Defense, USA*.

Stringfellow, M. V. (2010). *Accident analysis and hazard analysis for human and organizational factors.* PhD thesis, Massachusetts Institute of Technology.

Stringfellow, M. V., Leveson, N. G., and Owens, B. D. (2010). Safety-driven design for software-intensive aerospace and automotive systems. *Proceedings of the IEEE*, 98(4):515–525.

Thimbleby, H. (2010). *Press on: principles of interaction programming.* The MIT Press.

Thomas, J., Lemos, F., and Leveson, N. (2012). Evaluating the safety of digital instrumentation and control systems in nuclear power plants. *NRC Technical Researcy Report2013*.

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

Wiegers, K. and Beatty, J. (2013). *Software requirements.* Pearson Education.

Yang, C. (2014). Software safety testing based on stpa. *Procedia Engineering*, 80:399–406.
Young, W. E. (2014). Stpa-sec for cyber security mission assurance. *Eng Syst. Div. Syst. Eng. Res. Lab.*
Zadeh, L. A. (1962). From circuit theory to system theory. *Proceedings of the IRE*, 50(5):856–865.
[heading=subbibliography]

ISSN 2183-489X