XSTAMPP 2.0: New Improvements to XSTAMPP Including CAST Accident Analysis and an Extended Approach to STPA

Asim Abdulkhaleq, Stefan Wagner
*Institute of Software Technology, University of Stuttgart, Germany
{Asim.Abdulkhlaleq, Stefan.Wagner}@informatik.uni-stuttgart.de

***Abstract*—XSTAMPP (eXtensible STAMP Platform) is a software tool developed to serve the widespread adoption and use of STAMP methodologies in different domains. The first version of XSTAMPP supported only the STPA application. In this paper, we present a new version of XSTAMPP, including CAST accident analysis and extended approach to STPA. We developed two new plug-in tools called (1) A-CAST (Automated CAST) which implements the CAST activities and (2) XSTPA (Extended Approach to STPA) which supports automatically generating the context tables which will be used to refine the safety requirements and automatically transform the refined safety requirements into a formal specification in Linear Temporal Logic (LTL) to support verification activities. XSTAMPP 2.0 is available as an open source platform at our repository http://sourceforge.net/projects/stampp/files/2.0.0/**

***Keywords*-STAMP; STPA; CAST; extended approach to STPA; formal specification, verification**

I. Introduction

STAMP (Systems-Theoretic Accident Model and Processes) [7] is an accident model based on system and control theory to safety engineering, which treats safety as a control problem rather than a single component problem. STAMP considers the system as an interaction of control loops and accidents arise from the inadequate control of system components or external disturbances within the interactions with other components in the system environment such as humans, organizations or other physical system components. STAMP has two methodologies: STPA (Systems-Theoretic Process Analysis) for safety analysis and CAST (Causal Accident Analysis based on STAMP) for accident analysis. STAMP methodologies have been successfully applied in different areas in industry (e.g. STPA for security [11], Interval Management in NextGen [5], Nuclear Power Plants [10], and STPA for software safety [4].

To support the widespread adoption of STAMP, we have developed a tool support called A-STPA [1] which was our first prototype to implement the STPA activities and provides tool support for the STPA application in different domains. This prototype shows a number of shortcomings in terms of drawing a complex control structure diagram and documenting data in the tables. Moreover, A-STPA is a stand alone version only for STPA application, and it does not have the potential to support the other STAMP applications. To overcome these shortcomings of A-STPA [1], we have developed an extensible platform called XSTAMPP [3] based on our A-STPA tool to support the STPA application and has potential to include new plug-ins for other approaches.

A. Problem Statement

The increase in the usage of STAMP methodologies has fostered the need for developing a support tool to assist safety engineers in performing the safety analysis as well as the accident analysis. XSTAMPP has been already being used by safety analysts in different industry domains. However, the first version supported only STPA and had a limitation in editing a large number of data.

B. Research Objectives

The overall objective of our research is to extend XSTAMPP to support other STAMP applications and new improvements to STAMP. Moreover, we also aim to support safety verification activities by automatically transforming the STPA safety requirements into a formal specification in LTL. Providing a formal specification of STPA safety requirements will help safety analysts to verify design models of the system and software against the STPA safety requirements.

C. Contribution

The contribution of this research constitutes implementing two plug-in tools supporting the application of CAST accident analysis (A-CAST) and the application of an extended approach to STPA (XSTPA). We also used a combinatorial testing algorithm [12] to automatically generate the context tables. Moreover, we implemented an algorithm to transform the refined safety requirements automatically into LTL formulae.

II. XSTAMPP Architecture

XSTAMPP is an open-source platform written in Java based on the Eclipse Plug-in-Development Environment (PDE)[1] and Rich Client Platform (RCP)[2]. The RCP architecture provides easy expandability and flexibility. XSTAMPP

[1]://www.eclipse.org/pde/
[2]https://wiki.eclipse.org/Rich_Client_Platform

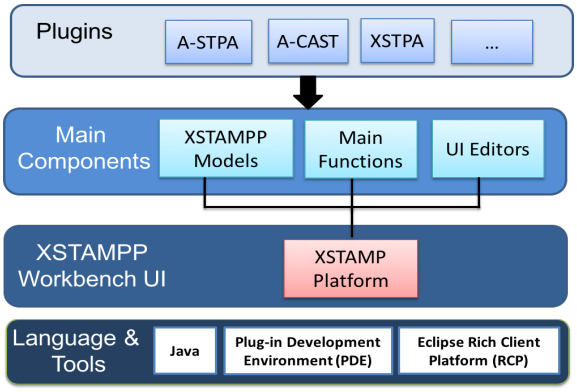

Figure 1. The XSTAMPP Architecture

provides core functionality and components that make it easy to extend with new plug-ins. XSTAMPP was developed with the background to create a base platform support for safety engineering, which can be easily extended with new functions and approaches based on the STAMP model.

Figure 1 shows the main architecture of XSTAMPP. XSTAMPP provides three main components which are shared by different plug-ins: 1) XSTAMPP models which represent all STAMP components such as the STAMP data tables, STAMP data lists, and STAMP text areas; 2) Main functions of the platform such as the save, load and export functions; and 3) the UI editors which are the user interface editors to view each STAMP component in the main workbench. XSTAMPP also provides an internal representation of each STAMP model which is associated with an XML element that documents the data of each component.

Figure 2 shows the main workbench of XSTAMPP which is divided into three main parts: a project explorer, the user interface viewer and the toolbox. The safety engineers can use XSTAMPP to perform CAST accident analysis and STPA safety analysis. They can create and open two different types of XSTAMPP projects: STPA safety analysis and CAST accident analysis in the project explorer. Any XSTAMPP project includes all necessary components to document the results of applying STPA and CAST in different domains such as fundamentals of analysis, drawing the control structure diagram of the system, editing STAMP data (i.e. the unsafe control actions, causal factors, safety roles and responsibilities).

III. XSTAMPP Plug-ins

The current version of XSTAMPP has three parts implemented as Eclipse plug-ins: 1) The A-STPA plug-in which supports the application of STPA safety analysis, 2) A-CAST (Automated CAST) which is a tool that implements the functionality of CAST and 3) XSTPA (extended STPA) which is a tool that implements an extended approach to STPA. XSTPA also implements an algorithm to automatically transform safety requirements into LTL formal specifications (see Algorithm 1).

A. A-STPA

A-STPA is an eclipse plug-ins based on the A-STPA stand-alone version to support the application of STPA. In XSTAMPP 2.0, we enhanced the usability and functionality of A-STPA by removing the shortcomings of A-STPA. We built a new editor for documenting a large number of unsafe control actions and causal factors. Moreover, we improved the drawing features of the control structure diagram by implementing a new component as a dashed box component to add comments or to group some components in one box and a component of the control actions list in which the safety analyst can draw and link multi-control actions to one arrow.

B. A-CAST

A-CAST (Automated CAST) is our proposed tool which we developed based on the XSTAMPP architecture to support performing the CAST accident analysis steps and to help the analyst during investigating an accident based on the STAMP model. CAST has the following main steps [7]:

- Identify the basis of analysis e.g. system-hazards which are violated, the system safety design constraints and proximal events.
- Construct the safety control structure as it was designed to work.
- Accident analysis in which the analyst should evaluate each component in the safety control structure diagram and determine whether it fulfilled its responsibilities or provided inadequate control. The analyst should also examine the coordination and communication between the main players of the accident under analysis.
- Generate recommendations.

A-CAST implements the aforementioned steps of CAST. It allows the safety analyst to draw the control structure diagram and document the safety responsibilities and roles of each component in the control structure diagram. Moreover, A-CAST allows the safety analyst to edit and document recommendations for each component directly on the control structure diagram by double-clicking on the component. The results of the CAST accident analysis can be exported completely in one PDF file or individually as PDFs, images, and CVS files for each CAST step.

C. XSTPA

Thomas [9] proposed a new extended approach to STPA for refining the unsafe control actions which are identified in STPA Step 2 based on the combinations of process model variables of each controller in the control structure diagram. Suo and Thomas developed a prototype [8] called STPA Tool to automate the extended approach to generate the combinations of process model variables and refine the

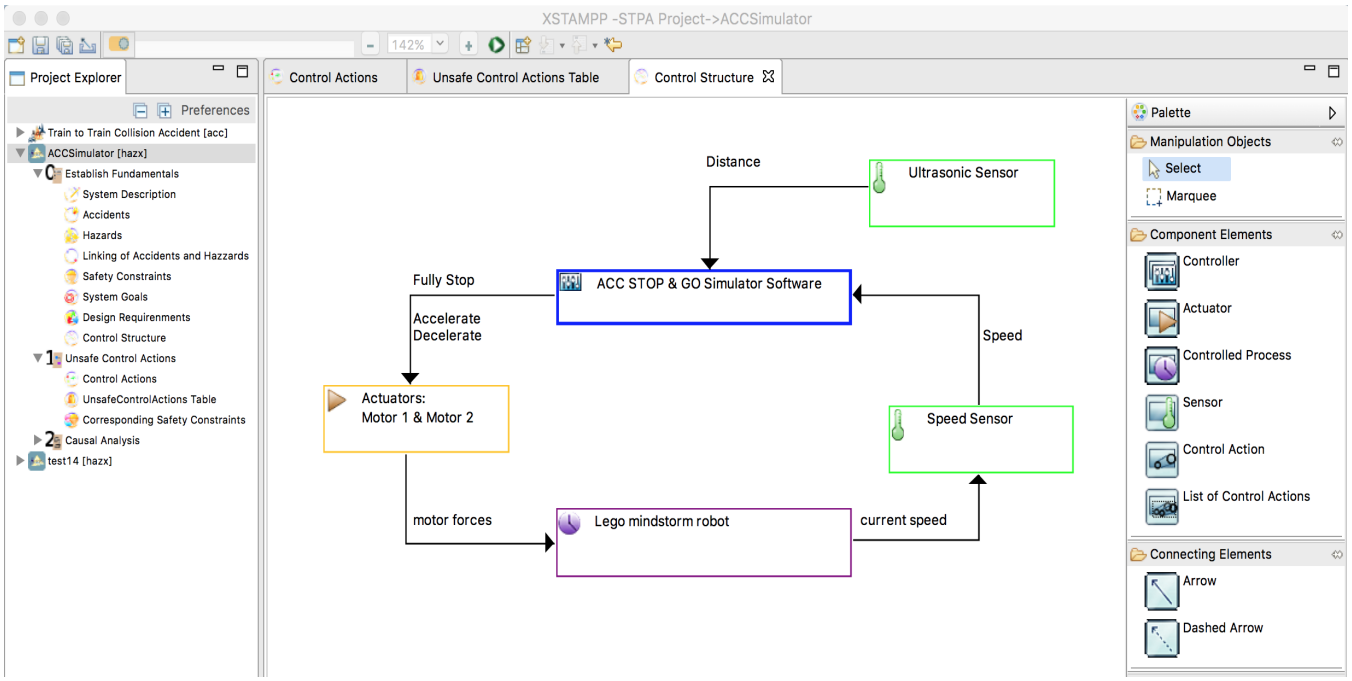

Figure 2. The main workbensch of XSTAMPP

unsafe control actions. STPA Tool was presented in the 2014 STAMP workshop at MIT. However, at present there is no version available of this prototype. Therefore, we developed a tool support called XSTPA (Extended plug-in to STPA) to automatically generate the set of combinations of process model variables.

Abdulkhaleq and Wagner [2] developed an algorithm to transform hazardous combinations in context tables into the LTL specifications. We implemented this algorithm in XSTPA to formalize the STPA safety requirements into LTL specifications to be used for verification purposes such as model checking or test case generation.

XSTPA is an Eclipse plug-in containing both algorithms. It uses a combinatorial testing algorithm [12] to automatically generate the context table and identify a minimal combination of process model variables for large and complex systems. XSTPA uses a Java library for the combinatorial testing algorithm called ACTS[3] which was developed by the American National Institute of Standards and Technology to generate combination sets of t parameters with n values.

When the safety analyst augments the process model into a controller component in the control structure diagram view, the XSTPA editor will automatically appear under the control structure diagram UI view with necessary information about the controllers, their process model variables with their values and the control actions on the control structure diagram. Based on the transform algorithm (see Algorithm 1), the safety analyst has to choose which control action is safety-critical to generate its context tables. Then, the safety analyst has to determine the dependences D between the control action and its relevant process model variables in two contexts (Co_1= providing and Co_2 =not providing). For each dependency relation, XSTPA will automatically generate an input file N as input to the ACTS algorithm

[3]http://csrc.nist.gov/groups/SNS/acts/index.html

Algorithm 1 Transform Algorithm of STPA safety Requirements into LTL specification

Input: CAs : A list of control actions, $PMV_{i,j}$: A table of process model variables $\bigcup \mathcal{P}$ and values
Data: Cos : Two contexts {Provided, Not Provided}
Output: LTL= A list of LTL specifications
Description:

for each control action $CA_i \in CAs$ **do**
  for each context $Co_i \in Cos$ **do**
    Define a dependence matrix $D_i = CA_i \times PMV$
  Construct an input text file N_i with ACTS format
  $ACTS \leftarrow N_i$
  Generate combination sets $CS_i = \bigcup(\mathcal{P}_{i,1} \wedge \ldots \mathcal{P}_{i,n})$
  Evaluate each combination set $f(\overrightarrow{CS_i})$
  if $f(\overrightarrow{CS_i})$ is **hazardous** in the context Co_i **then**
    Check for any conflicted combination set in CS
    if $f(\overrightarrow{CS_i})$ is not **conflicted then**
      Refined Safety Requirements $\overrightarrow{SR_i}$
      $SR_i = (\Omega CS_i \rightarrow (CA_i \vee \neg CA_i))$
      $LTL_i \leftarrow G(SR_i)$

to automatically generate the context tables of each safety-critical control action. The input file of ACTS is a text file with the following format [12]:

```
[System]
 Name: ACC Adaptive Criuse Control
[Parameter]
 ACC_Mode (enum): Standby, Cruise, Follow
 Brake (enum) : NotApplied,  Applied
 Distance (enum): D>= safeDistance,
  D<safeDistance
[Relation]
[Constraint]
  ACC_Mode !='Off'
```

The [**System**] section contains the information about the STPA project and the [**Parameter**] section contains the definition of process model variables and their values. All process model variables are defined as *enum* data type to accept different kinds of values of the process model variables. The [**Relation**] section is an optional section which defines the strengths between the process model variables. For example, the ACC system has 3 process model variables, then the first relation can be created that consists of all the process model variables with strength 1 or 2. The safety analyst can define a custom relation when some process model variables have a higher degree of interaction or they are closely related to each other. The [**Constraint**] section is an optional section which includes the boolean conditions that combinations must satisfy to be valid. For instance, the constraint (ACC_Mode ! ='Off') will exclude all rows in the generated context table of process model variable *ACC_Mode* which contain the value *off*.

When the safety analyst clicks on the generate button in the context table, XSTPA will automatically generate the combinations of the process model variables $CS_i = \bigcup(\mathcal{P}_{i,1} \wedge \ldots \mathcal{P}_{i,n})$. From setting UI view, the safety analyst can apply different combination coverages [6] on the generated results such as pairwise coverage or t-way coverage to minimize the number of the combination sets. Pairwise coverage means that for each 2 process model variables, every possible combination of values of these 2 variables must be covered in at least one combination. T-way coverage means that for t process model variables, every possible combination of values of these t variables must be covered in at least one combination.

XSTPA can automatically show the conflicted combination sets. The conflicted combination sets mean that two or more combination sets have the same values of the process model variables in the both two contexts (providing and not providing) of one control action. If the safety analyst selects any combination as hazardous in any context, XSTPA will automatically express them as boolean expressions with an AND operator. Finally, the XSTPA will generate automatically the LTL formulae of the hazardous combination sets of a safety critical system. The results of XSTPA can also be exported as PDFs, images and CVS sheets.

IV. New Features of XSTAMPP

XSTAMPP is already being used by safety analysts in different industrial domains. Based on the feedback from XSTAMPP's users, we implemented new features to facilitate the use of XSTAMPP and improve the quality of XSTAMPP. The new features that we included in the new version 2.0 of XSTAMPP are listed below:

- XSTAMPP 2.0 runs on the Linux operating system besides Windows and Mac OS X.
- XSTAMPP 2.0 has a new editor to enable the user to document a large number of unsafe control actions and causal factors.
- XSTAMPP 2.0 accepts any Eclipse plug-ins to run under the XSTAMPP architecture to allow other researchers to integrate their own tools in XSTAMPP.
- XSTAMPP 2.0 allows the users to create a different kind of STAMP projects (e.g. STPA project and CAST project).
- XSTAMPP 2.0 provides a button to generate the final report and all the results of STAMP project and export them into different formats.
- XSTAMPP 2.0 improves the quality of drawing the control structure diagram by providing new additional components and allowing the user to draw a link between components from any point in the border of a component.
- XSTAMPP 2.0 allows the users to copy and paste the components in the control structure diagram.
- XSTAMPP 2.0 has a filter search option in each UI editor to enable the user to search for data by using different search criteria.
- With the new editor of the control structure diagram, the user can link one or more control actions to an arrow in the diagram. The information about the source and destination of the control action will appear automatically in the control actions table.

V. Conclusion and Future Work

We have presented a new version of our XSTAMP platform, including CAST accident analysis and the extended approach to STPA to support the application of STAMP methodologies in different industrial environments. The new version aims at implementing the steps of STAMP/CAST to support accident analysts in performing the STAMP/CAST method. Moreover, the new version aims to provide a tool support to generate the context tables and to generate the corresponding formal specifications of the refined safety requirements to support the safety verification activities.

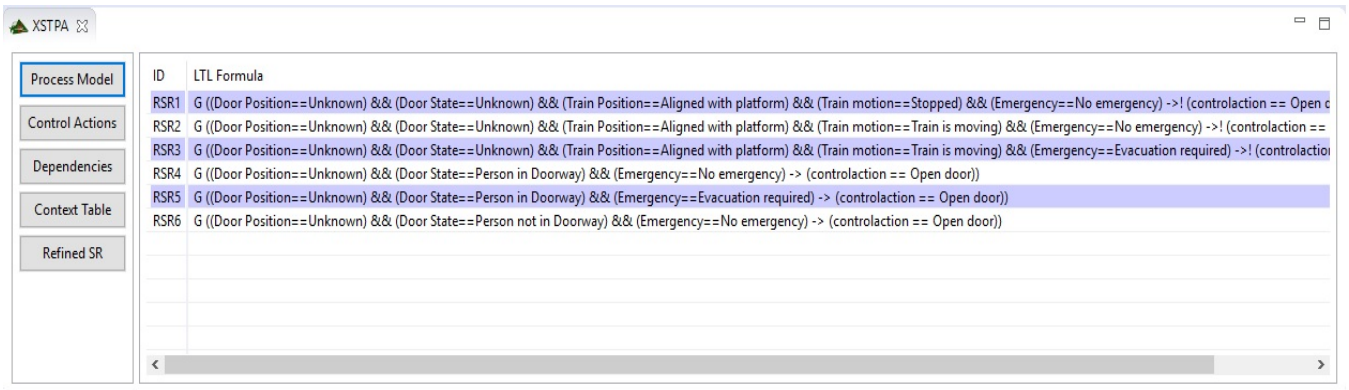

Figure 3. The generated LTL specifications by XSTPA

We developed the STAMP platform based on the Eclipse Rich Client Platform (RCP) as an extensible software platform that can be extended in the future with new requirements and improvements to STAMP. The platform is implemented in Java as open source.

As future work, we aim to benefit from the new architecture to implement a new plug-in called STPASec to implement the steps of the STPA security approach [11] and provide them in the upcoming version of the platform. We also aim to support our safety engineering approach for software-intensive systems [4] by developing a plug-in called STPA verifier to enable the safety analyst to verify their software design and implementation against the STPA safety requirements which are transformed into LTL formulae with XSTPA by using model checking and generate a safety verification report. Providing a plug-in which connects A-STPA with model checking in XSTAMPP will enable the safety analysts to verify the system design and software code against the STPA safety requirements.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to Lukas Balzer, University of Stuttgart, who worked with us to improve and build the XSTAMPP platform, Martin Root who developed A-CAST and Yannic Sowoidnich, University of Stuttgart, who developed XSTPA, and Mr. Rick Kuhn, National Institute of Standards and Technology, USA who provides us the automated combinatorial testing tool (ACTS) . We are grateful for their effort and time.

REFERENCES

[1] A. Abdulkhaleq and S. Wagner. A-STPA: Open tool support for system-theoretic process analysis. *2014 STAMP Conference at MIT*, 2014.

[2] A. Abdulkhaleq and S. Wagner. *Integrated Safety Analysis Using Systems-Theoretic Process Analysis and Software Model Checking*, volume 9337 of *Lecture Notes in Computer Science*. Springer International Publishing, 2015.

[3] A. Abdulkhaleq and S. Wagner. XSTAMPP: An extensible STAMP platform support tool safety engineering. *2015 STAMP Conference at MIT*, 2015.

[4] A. Abdulkhaleq, S. Wagner, and N. Leveson. A comprehensive safety engineering approach for software-intensive systems based on STPA. *3rd European STAMP Workshop, Amsterdam*, 2015.

[5] C. Fleming, M. Placke, and N. Leveson. Technical report: Stpa analysis of nextgen interval management components: Ground interval management (gim) and flight decn interval management (fim). *MIT*, 2013.

[6] D. Kuhn, R. Kacker, and Y. Lei. A combinatorial coverage measurement. *National Institute of Standards and Technology*, September 2012.

[7] N. Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety*. Engineering Systems. MIT Press, 2011.

[8] D. Suo and J. Thomas. An STPA tool. *STAMP 2014 Conference at MIT*, 2014.

[9] J. Thomas. Extending and automating a systems-theoretic hazard analysis for requirements generation and analysis. *SANDIA National Laboratories report 2012-4080*, 2012.

[10] J. Thomas, F. Lemos, and N. Leveson. Evaluating the safety of digital instrumentation and control systems in nuclear power plants. *MIT Technical Report*, November, 2012.

[11] W. Young and N. Leveson. Inside risks-an integrated approach to safety and security based on system theory: Applying a more powerful new safety methodology to security risks. *Communications of the ACM*, 57(2):232–242, 2014.

[12] L. Yu, Y. Lei, R. Kacker, and D. Kuhn. ACTS: A combinatorial test generation tool. *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*, pages 370–375, March 2013.