Korea-Japan Joint Workshop on ICT (Pohang, Korea, 20-22 September 2012), Paper #03

SW-STPA: A Software Hazard Analysis Technique based on STPA

Sun Hwi Lee[†1,a)] Sanghyun Yoon[†1,b)]
Junbeom Yoo[†1,c)]

Abstract: As the uses of software are various, software is germane to human's life and property. Thus, the importance of software safety increases rapidly and many hazard analysis techniques are used for safety of system/software. STAMP/STPA is an efficient hazard analysis technique for large and complex system. But subject of STAMP/STPA is system, not software. This difference of subjects makes difficulty apply STPA to small software. We propose SW-STPA to overcome the difficulties and conduct a case study. We expect that it will help safety experts when analyze causal factors of software hazards with STAMP/STPA.

Keywords: STAMP, STPA, Hazard analysis

1. Introduction

As the uses of software are various, software is germane to human's life and property. Thus, the importance of software safety increases rapidly and hazard analysis techniques are used for safety of system/software. Hazard analysis is the process of identifying hazards and their potential causal factors [1]. The goal of hazard analysis is accumulating information about how the behavioral safety constraints, which are derived from the system hazards, can be violated.
STAMP (Systems-Theoretic Accident Model and Process)/STPA (System-Theoretic Process Analysis) is the one of accident causal model and hazard analysis technique for large and complex system. In STAMP, systems are viewed as interrelated components kept in a state of dynamic equilibrium by feedback control loops [2]. And STPA is a hazard analysis technique which is based on STAMP causality model. The goal of STPA is to identify accident scenarios that encompass the entire accident process. To satisfy goal, STPA includes all of accidents scenarios about components of system and their interactions like design errors, software flaws, component interaction accidents, cognitively complex human decision making errors, etc. So, STPA is suitable technique for large and complex system. Case studies using STPA show that STPA could identify more hazards than older techniques [3]. But subject of STPA is system, not small software. This difference of subjects to analyze makes difficulty apply STPA to small software. Therefore, if we could develop software hazard analysis technique based on STPA, it helps software developer have more various sights about software hazard analysis.
This paper presents SW-STPA which is transformed STPA to analyze software hazards. We propose new general form of safety control structure for software with keeping the advantages of STPA. And we applied this proposed technique to *FBDtoC*, the translator developed for RPS.
FBDtoC is the translator which is used in RPS (Reactor Protection System) development life cycle. RPS makes decisions for emergent reactor shutdown. Therefore, RPS software should be developed in safety. RPS software is typically modeled with IEC-61131 FBD (Function Block Diagram) [4] in design phase. In implementation phase, the FBD programs are translated into C programs, which are compiled into executable code for RPS software. In this phase, FBDtoC is used for translation with guaranteeing there behavioral equivalence fundamentally.
The organization of the paper is as follows: Section 2 gives introductions to STAMP/STPA. In section 3, we propose SW-STPA and new general form of safety control structure in SW-STPA. Section 4 shows the results of case study which is applying SW-STPA to FBDtoC we developed. Section 5 concludes the paper and gives remarks on future research extension and direction.

2. Backgrounds

2.1 STAMP

In the traditional causality models, accidents are considered to be caused by chains of failure events. But, fast pace of technological change, the occurrence of new types of hazards, increasing complexity of system, and other causes make traditional accident models be adequate for large and complex system no longer. STAMP is developed to solve these problems.
STAMP is an accident model based on three principles: safety constraints, hierarchical safety control structures, and process models [5]. First basic concept is safety constraints. Safety constraints are the enforcements that must not be violated for safety of system. In STAMP, the accidents can occur when safety constraints were not successfully enforced. Second is hierarchical safety control structure. In STAMP, systems are viewed as hierarchical structures. Each level imposes constraints on the activity of the level beneath it. The last concept is process model. In STAMP, every control level has process model. Process model contains everything to decide control action: variables, control logic, current state, state transition, etc. Figure 1 describes the process model and the operations between levels of control. In this figure, accidents can occur when the controller's process model does not match the system being

†1 Konkuk University, 120 Neungdong-ro, Gwangjin-gu, Seoul 143-701, Korea
a) bigsaram@konkuk.ac.kr
b) pctkdgus@konkuk.ac.kr
c) jbyoo@konkuk.ac.kr

controlled and the controller orders unsafe command.

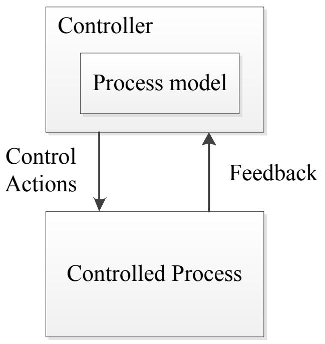

Figure 1 Process model and operations between levels of control

2.2 STPA

STPA is a hazard analysis technique based on STAMP. The goal of STPA is identifying accident scenarios that encompass the entire accident process. Additional goals are providing guidance to users and information necessary to guide the design process and making it can be used before a design has been created. STPA can be applicable to existing designs or systems.

STPA uses a control diagram and the functional requirements, system hazards, and the safety constraints and safety requirements for the component. Thus, before the STPA process, developer must establish these fundamentals.

Fundamentals are:

1) Defining accidents and unacceptable losses for system
2) System hazards
3) System safety requirements and constraints
4) Safety control structure

Developer can use STPA technique after the fundamentals are established. STPA has two main steps as the following:

1) Identify the potential for inadequate control of the system that could lead to a hazardous state. hazardous states result from inadequate control or enforcement of the safety constraints, which can occur because:
   a) A required control action is not provided or not followed;
   b) An incorrect or unsafe control action is provided;
   c) A potentially safe control action is provided too early or too late, that is, at the wrong time or in the wrong sequence;
   d) A correct control action is stopped too soon.
2) Determine how potentially hazardous control action identified in step 1 could occur.
   a) Augment the control structure with process models for each control component.
   b) For each unsafe control action, examine the parts of the control loop to see if they could cause it. Design controls and mitigation measures if they do not already exist or evaluate existing measures if the analysis is being performed on an existing design. For multiple controllers of the same component or safety constraint, identify conflicts and potential coordination problems.
   c) Consider how the designed controls could degrade over time and build in protection.

3. SW-STPA

Subject of analysis is system in STPA. So, general form of safety control structure in STPA is composed of controller, actuator, controlled process, and sensor. And a control loop is made by them. Because of this control loop, system is kept in a state of dynamic equilibrium. When controller gives control action to actuator, actuator operates physical controlled process. And then sensor senses controlled process and gives feedback to controller.

Subject of analysis is software in SW-STPA. Thus, general form of safety control structure in SW-STPA could not be composed of controller, actuator, sensor, controlled process. In figure 2, different from STPA, general form of safety control structure in SW-STPA is composed of SW controller, functional controllers, and information. Omitted in figure 2, each controller has a process model, same as controller in STPA.

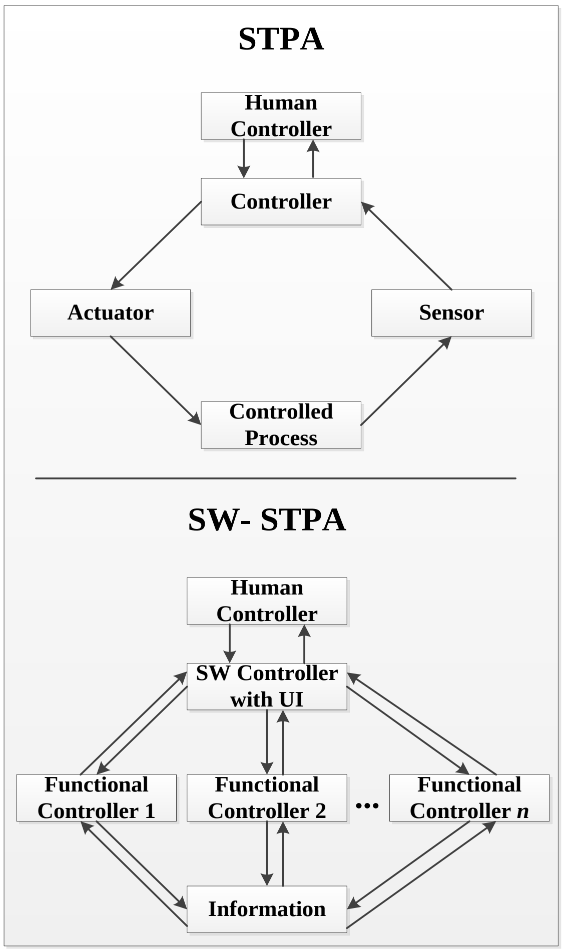

Figure 2 Differences between SW-STPA from STPA

There are two reasons of new general form of safety control structure in SW-STPA. First reason is bringing about advantages from the general form of safety control structure of STPA. In STPA, each controller has control actions. And general four types of inadequate control actions help identifying potential hazardous control actions. Therefore, if each function of the software could be the functional controller, it has control actions and helps to identify more potentially hazardous states.

Second, there is no actuator or sensor which operates physical controlled process or senses in software. Instead of actuator and sensor, functional controller exists. *Functional controller* in figure 2 is the separated function of software. For example, if the subject software is the translator, parser or open file will be *functional controller*. In order to maintain feedback control loop in software without partition like actuator and sensor, each *functional controller* needs to give feedback to *SW controller with UI* in figure 2. It means that if *SW controller with UI* gives control action to *functional controller*, then every *functional controller* actuates that and has to check the current state. In additional, each *functional controller* could be separated by small functional controllers such as safety control structure in STPA. For example, if digital watch is software and alarm controller is the one of functional controllers, and then alarm will be able to be separated into small functional controllers like ringing controller, and setting alarm time controller.

In STPA, controlled process means that actuator operates the physical process that controller ordered. But in SW-STPA, it is not possible that the functional controller operates the physical controlled process. Every functional controller can only handle information in software. Therefore, new concept is needed in software. *Information* in figure 2 is the all of information which functional controller created, changed, and deleted. For example, if the alarm controller is the functional controller of digital watch, then the time when alarm rings and the variable which indicates alarm rings or not will be the elements of Information.

SW controller with UI is a controller which controls software as a whole and communicates with human controller. Details of *SW controller with UI* are described in figure 3. *SW controller with UI* is separated by *input interface*, *output interface*, and *SW controller*. *Input interface* gives what human controller ordered to *SW controller*. *Output interface* gives feedback of *SW controller* to *human controller*. This separation of SW controller with UI has advantage when the software needs complex UI because of large scale.

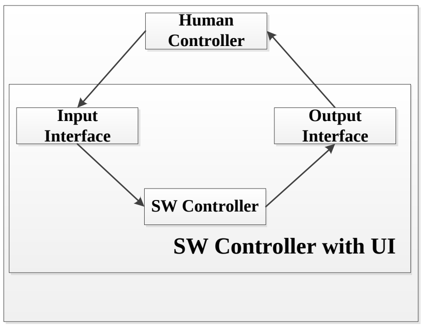

Figure 3 Details of SW Controller with UI

4. Case study

We applied SW-STPA to FBDtoC the translator we developed. Because that FBDtoC has been developed, we established safety control structure refer to source codes. One of the features of STPA is that developer can apply STPA to their project in any development phases. Thus, SW-STPA is same with STPA, we could apply general form of safety control structure in SW-STPA to FBDtoC.

FBDtoC is composed of two levels in safety control structure. It does not need separation of SW Controller with UI to SW controller, input interface, output interface because it has simple translator. High level safety control structure of FBDtoC is described in figure 4. FBDtoC consists of one *FBDtoC controller*, and four functional controllers. *File open* does operation open file, as similar to *file save*. *XML parser* is provided by PLCOpen. *XML parser* parses XML file to parsed data. *Translator* builds parsed data into data structure for translation and translates built data into C language. Therefore, *Translator* can be separated to builder and translator. Figure 5 describes details of translator.

In figure 5, we described details of translator. *Builder* builds four types of elements: *variable*, *block*, *component*, and *system*. After build elements into four types, translate controller orders translation to three translators: *block translator*, *component translator*, *system translator*.

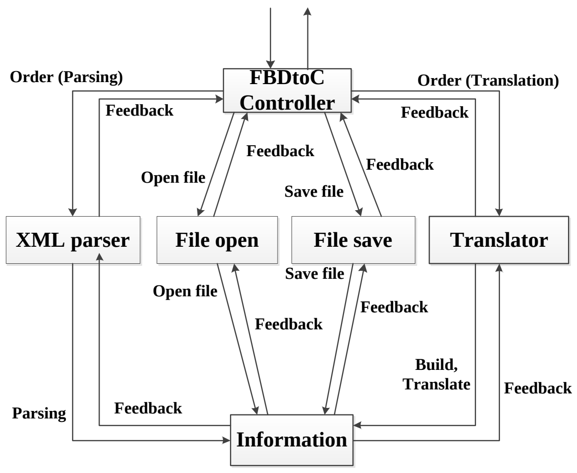

Figure 4 Safety control structure level 0 of FBDtoC

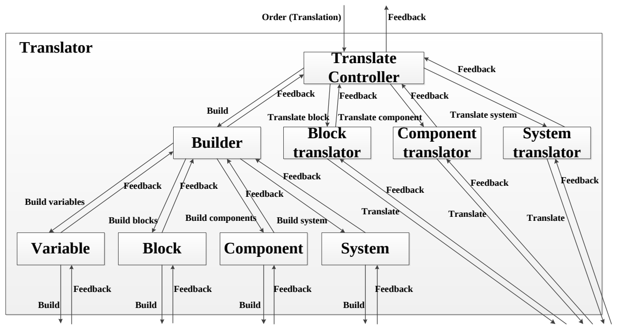

Figure 5 Safety control structure level 1 of Translator

5. Conclusions & Future work

STAMP/STPA is powerful hazard analysis technique for system. Applying STPA to software, we have to solve the problem that subject of analysis is different from subject of STPA. To solve this problem, we proposed SW-STPA which is hazard analysis technique for software. And we proposed new general form of safety control structure in SW-STPA. We applied SW-STPA to FBDtoC, the translator we developed. And the new general form described FBDtoC successfully.

SW-STPA is developed step 1, step 2 is not developed yet. We are now planning to develop SW-STPA step 2. We also plan to apply all steps of SW-STPA to FBDtoC as a whole, and compare with other hazard analysis technique.

Reference

1) Leveson, N.G.: "Safeware: system safety and computers", ACM, 1995.
2) Nancy G. Leveson: "Engineering a Safer World", MIT Press (MA), 2010.
3) http://csrl.scripts.mit.edu/home/stampstpa-workshop.
4) International Electrotechnical Commission: "International standard for programmable controllers: Programming languages," 1993, part 3.
5) Leveson, N.G.: "A new accident model for engineering safer systems", Safety Science, Elsevier, 42(4), 237-270, 2004.

Acknowledgments This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2012-0003619) and by the MKE (The Ministry of Knowledge Economy), Korea, under the Development of Performance Improvement Technology for Engineering Tool of Safety PLC (Programmable Logic Controller) program supervised by the KETEP (Korea Institute of Energy Technology Evaluation And Planning)" (KETEP-2010-T1001-01038). It was also supported, in part, by a grant from the Korea Ministry of Strategy, under the development of the integrated framework of I₩&C conformity assessment, sustainable monitoring, and emergency response for nuclear facilities.

Sun Hwi Lee was born in 1985. He is a M.E. candidate in Konkuk University, Seoul, Republic of Korea.

Sanghyun Yoon was born in 1987. He is a Ph.D candidate in Konkuk University, Seoul, Republic of Korea.

Junbeom Yoo was born in 1975. He is an assistant professor in Konkuk University's Department of Computer Science and Engineering. His research interests include requirements engineering and formal methods. He has a PhD in computer science from the Korea Advanced Institute of Science and Technology.