IEICE TRANSACTIONS

on Information and Systems

DOI:10.1587/transinf.2021EDL8073

Publicized:2022/02/10

This advance publication article will be replaced by the finalized version after proofreading.

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

LETTER

Unfolding Hidden Structures in Cyber-Physical Systems for Thorough STPA Analysis

Sejin JUNG[†], **Eui-Sub KIM**[†], ***Nonmembers***, ***and*** **Junbeom YOO**[†a)], ***Member***

SUMMARY Traditional safety analysis techniques have shown difficulties in incorporating dynamically changing structures of CPSs (Cyber-Physical Systems). STPA (System-Theoretic Process Analysis), one of the widely used, needs to unfold and arrange all hidden structures before beginning a full-fledged analysis. This paper proposes an intermediate model "*Information Unfolding Model (IUM)*" and a process "*Information Unfolding Process (IUP)*" to unfold dynamic structures which are hidden in CPSs and so help analysts construct control structures in STPA thoroughly.

key words: *Safety Analysis, Cyber-Physical System, System-Theoretic Process Analysis, Control Structure*

1. Introduction

CPSs (Cyber-Physical Systems) [1] are highly interconnected and collaborated with various heterogenous components/subsystems as well as physical environments. CPSs in safety-related domains should demonstrate that the whole system is acceptably safe from identified hazards and risk [2] with the help of hazard analysis techniques [3]. Hazard analysis must consider a unique feature to CPSs - "*dynamically changing structures of CPSs*" which typical techniques cannot cope with.

STPA (System-Theoretic Process Analysis) [4] is one of the widely used hazard analysis techniques, which can analyze hazards between interconnected and collaborated components/systems. For CPSs whose structures are hidden, *i.e.,* dynamically changing, STPA needs to unfold and arrange all possible hidden structures before starting the analysis in earnest. In some cases, multiple instances of the system can exist and collaborate at runtime in dynamically [5]. However, it is not simple to identify all situations that can bring about the changes in CPS structures [6].

This paper proposes an "*Information Unfolding Process (IUP)*" and an intermediate model "*Information Unfolding Model (IUM)*" which can encompass all possible structural changes in a CPS. The process provides a systematic step to extract an IUM model from various artifacts in software development life-cycles. We then can construct all possible combinations of control structures used in STPA roughly but thoroughly by using the proposed intermediate model. We also performed a case study with the vehicle platooning system to show the proposed model and process can show various possible control structures.

The remainder of the paper is organized as follows: Section 2 provides background information and related work on hazard analysis of CPSs. Section 3 proposes the IUP and the IUM in detail, while a short case study is shown in Section 4. Section 5 concludes the paper and provides remarks on our future research extension and direction.

[†]The authors are with the Division of Computer Science and Engineering, Konkuk University, Republic of Korea.

a) E-mail: jbyoo@konkuk.ac.kr (corresponding author)

2. Background

2.1 STPA

STPA is a hazard analysis technique based on STAMP (System-Theoretic Accident Model and Process) [7]. The system is viewed as a hierarchical structure that higher-level components *control* lower-level components and lower-level components *feedback* to higher-level components [4] according to the STAMP "*control structure*" model. STPA focuses on identifying hazardous controls, called UCAs (Unsafe Control Actions), between the controlling (controller) and controlled components (controlled process) as described in ⟨Fig.1⟩.

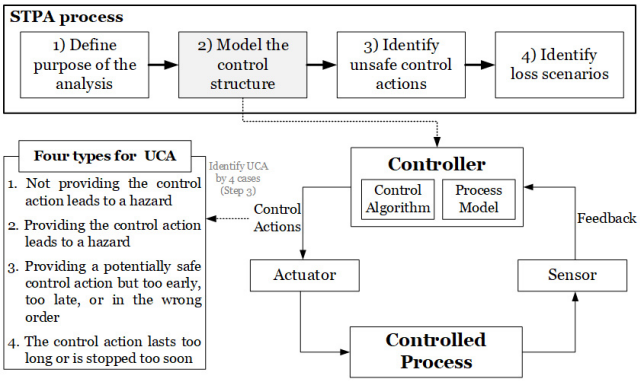

Fig. 1 A typical process and a control structure of STPA

The STPA analysis needs to pin a (set of) control structure(s) down in Step 2 before proceeding to the next. A CPS consists of various dynamically changing elements, and we have to identify and spread out all combinations of structural elements [6] in order to analyze thoroughly without missing components, as motivated by this paper.

2.2 Related Work

Several studies have tried to support the STPA analysis on complex and cooperative systems, such as the system of systems (SoS) and CPS. [8] integrates an FSM (Finite State

Machine) into the STPA process to analyze dynamic behaviors. [9] uses a control diagram to model a three-layer SoS. [10] also proposes an additional STPA process based on operation modes of autonomous ocean systems. [11] tries to identify unsafe control actions of SoS with a Petri-net formal model. [5] revealed there need to validate the behavior originating from the collaboration of multiple instances of the same type systems. The such collaboration/behaivor can show various and different context structures in dynamically. However, they are all attempting to cope with specific dynamic situations, not to analyze and unfold all hidden structures systematically, as this paper proposes.

3. The Information Unfolding Process

The IUP intends to discover all hidden structures through analysis of various system specifications, as illustrated in ⟨Fig.2⟩. From the IUM produced at step 3, we can generate all possible combinations of control structures, which will be used at step 2 of STPA.

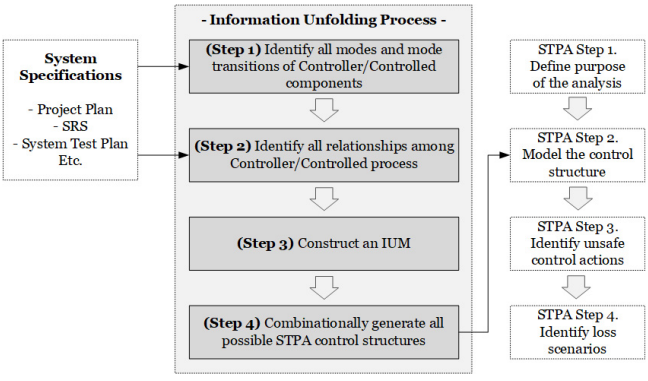

Fig. 2 The information unfolding process

(Step 1) Controllers and controlled components are modeled with FSMs through collected information about dynamically changing elements from available system specifications such as SRS (Software Reuiqrements Specification). The controller/controlled process relations can also be revealed between different modes in multiple instances of systems. The FSMs behave differently depending on the current state or mode, even with the same input, and they play an important role to find combinations of hidden structures. We are now working on extracting FSMs mechanically from artifacts in software development life-cycles.

(Step 2) The relationship between the elements making an FSM up is then identified. Multiplicities (1..1 1..5 0..3), as the UML class diagram, is an easy way to describe their relationships. It would be useful to spread every possible combination to show the possibility of multiple and dynamic existences of controllers. Sometimes a controller may become a controlled process in different modes and situations.

(Step 3) From the information on dynamically changing elements and their relationships, we can construct an IUM as illustrated in ⟨Fig.3⟩. It is an FSM consisting of nodes and transitions, where the nodes `E` are FSMs of modes/states and transitions `C` are controlling relations between nodes with multiplicities. For example, `(Controller 1)` has 3 modes and controls 1..5 `(Controlled 1)` and 0..5 `(Controlled 3)`. An IUM helps STPA analysts list-up all possible combinations of control structures through the generation algorithm introduced in (Step 4). A simplified definition of IUM is as follows:

$IUM = \langle\, \mathtt{E}\,,\, \mathtt{C}\,\rangle$, where

- $\mathtt{E} = \langle S, L, T \rangle$
  - S : a finite set of states (Modes)
  - L : a set of transition labels
  - T : a set of transitions, $S \times L \times S$
- $\mathtt{C} = (T, M)$
  - T : a set of transitions, $\mathtt{E} \times M \times \mathtt{E}$
  - M : a set of pairs of multiplicities

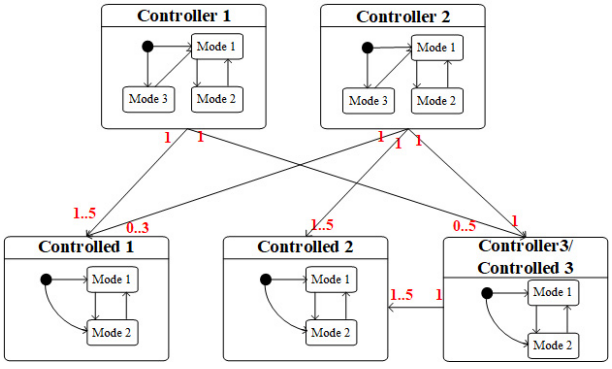

Fig. 3 An example IUM

Algorithm 1 Generate Control Structure from IUM

Require: IUM : The information unfolding model, $Entity$: The selected entity for controller

function GENCSFROMIUM($IUM, entity$)
  Create a controller Con list $ControlList$
  $ControlList \leftarrow findControlList(IUM, Entity)$
  for each Con $\in ControlList$ **do**
    Create a set_n = {x | x = $M.min..M.max$ && $M \in C$ of IUM}
    $CartesianProduct(set_1\ ..\ set_n)$
    for each item $\in CarteisianProduct$ result **do**
      Create a $Controller$ by selected $Entity$
      Create a $ControlledProcess$ × item.$Controlled_{1..n}$.$value$
      Generate a control structure with $Controller$ and $\forall Controlled$
    end for
  end for
  Identify combination of internal states of each control structure
end function

(Step 4) The last step is to generate all possible variations/combinations of rough control structures from the IUM

through the generation algorithm ⟨Alg.1⟩. For example, from the IUM in ⟨Fig.3⟩, the algorithm enumerates 55 different structures as illustrated in ⟨Fig.4⟩. ⟨Fig.4(a)⟩ is the case that the (`Controller 1`) is the controller† and it can have 1∼5 (`Controlled 1`) ⊗ 0∼5 (`Controlled 3`) = 30 different control structures.

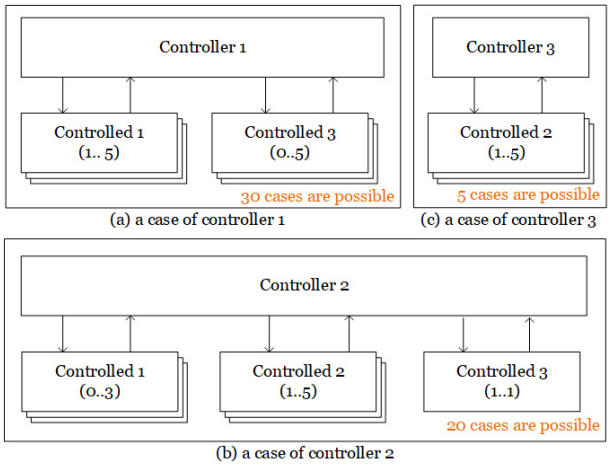

Fig. 4 A set of unfolded control structures

4. Case Study

We performed a case study to show feasibility and effectiveness of the information unfolding process and model with the widely used *vehicle platooning system* example [12], [13]. It is a cooperative automotive system for improving traffic capacity and energy efficiency. A leader vehicle leads and controls following vehicles as described in ⟨Fig.5⟩. It also has many functions such as *create/join/leave platoon*, *merge*, *split*, *acceleration*, *deceleration* and *leader change*.

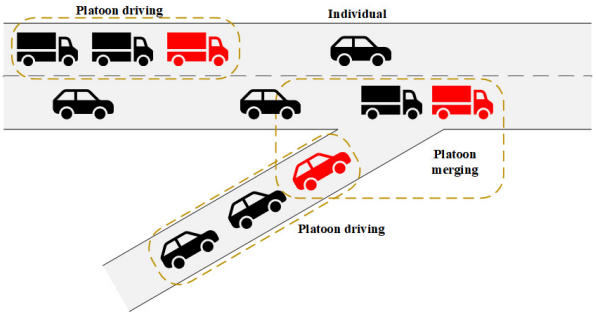

Fig. 5 A conceptual overview of the platooning system

The case study uses as system specifications a project plan and a software requirements specification used in [14]. In accordance with the IUP in ⟨Fig.2⟩, and from the system specifications, we first identified 3 elements such as *leader*, *follower* and *non-platoon (external) vehicle*. We also defined an individual FSM for each element and their control relationships to construct an IUM as shown in ⟨Fig. 6⟩. We can notice that the *leader* can have multiple *followers* while interacting with 0..* *external vehicles*. The *leader* may also have relationships with other *leaders* as the system specification specifies that "*Two different platoons can merge into a form of single platoon with a one leader* [12]."

From the IUM, the algorithm ⟨Alg.1⟩ generates all possible combinations of control structures as roughly shown in ⟨Fig.7⟩, while assuming * is up to 5. Each mode of elements is shown in parentheses as (`Leading`) or (`Merging`). They are roughly grouped into at least 4 categories such as

- (a) *Leader - Followers*
- (b) *Leader - Followers - External*
- (c) *Leader - Followers - External - Followers*
- (d) *Leader - Followers - External - Other Leader*

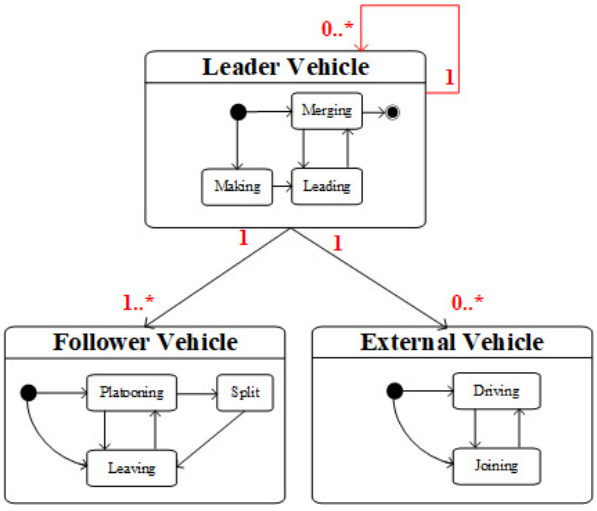

Fig. 6 The IUM constructed in the case study (simplified)

It is worth mentioning that the case (d) is not straightforward to identify from system specifications naively. There are also subtle difficulties in distinguishing (d) from (c) and (c) from (b). It, however, can be identified mechanically from the IUM of ⟨Fig.6⟩. The self-transition of `Leader Vehicle`, made in red, gives a hint about the case (d).

⟨Fig.8⟩ shows a refined version of the control structure for the case (d), which a `Leader` is followed by an `External` and an `Another Leader`. The STPA analysis (Step 3) then tries to identify various UCAs, which may result in system accidents, from the control structure, such as

- "*A leader provides a merge command to another platoon leader when an external vehicle is at the end of the line.*"
- "*A leader provides an acceleration command to followers while a non-platooning (external) vehicle is at the end of the line and another (single) platooning leader tries to merge.*"

† A control structure in STPA has only one controller.

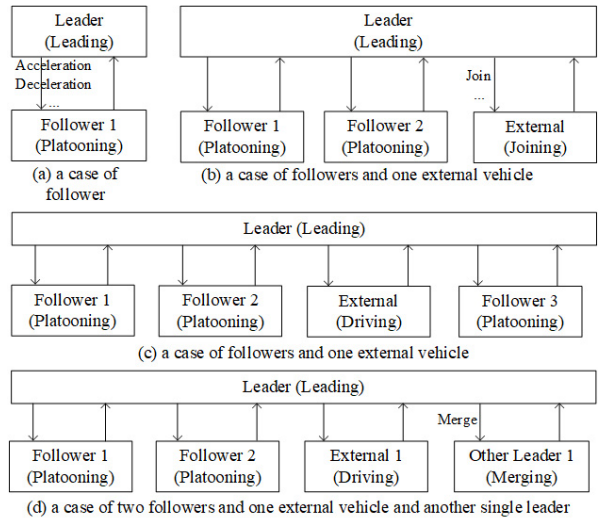

Fig. 7 Samples of control structures generated from ⟨Fig.6⟩

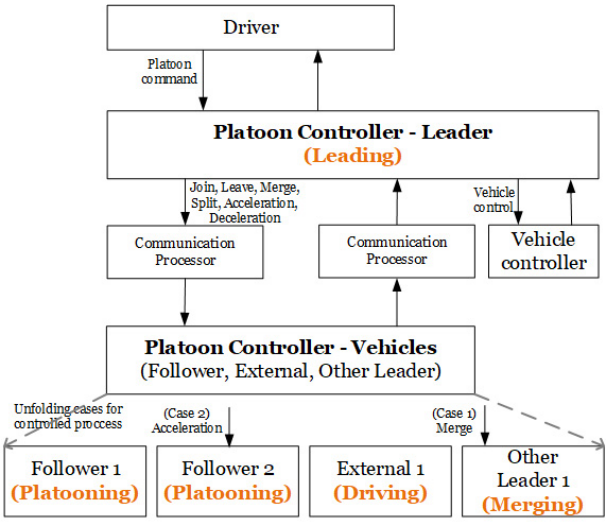

Fig. 8 A control structure with 4 controlled processes from (d)

These UCAs are not easy to extract from the typical control structures such as ⟨Fig.7 (a)⟩ and ⟨Fig.7 (b)⟩. Folded information about external vehicles and another leader who tries to join needs to be unfolded before constructing control structures and identifying UCAs. The information unfolding process, model, and the algorithm for constructing rough control structures will help analysts construct all possible combinations of control structures mechanically and thoroughly.

5. Conclusions and Future Work

This paper proposes an information unfolding process for CPSs which have various hidden dynamic structures. It uses an information unfolding model to capture hidden structural information from systems specifications, and also provides an algorithm constructing all possible combinations of control structures of STPA roughly, mechanically, and thoroughly. The case study shows that it will help safety analysts identify unusual and hard-to-find unsafe control actions thoroughly. We are now working on the systematic derivation and transformation of information unfolding models from informal as well as formal system specifications.

Acknowledgements

This paper was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2021R1F1A1047246).

References

[1] Cyber-Physical Systems Public Working Group, Framework for Cyber-Physical Systems: Volume 1, Overview, Version 1.0, NIST Special Publication, 2017.
[2] N. Ali, M. Hussain, and J.E. Hong, “Analyzing safety of collaborative cyber-physical systems considering variability,” IEEE Access, vol.8, pp.162701–162713, 2020.
[3] C.A. Ericson, Hazard analysis techniques for system safety, 2 ed., John Wiley & Sons, Hoboken, 2015.
[4] N.G. Leveson and J.P. Thomas, “Stpa handbook,” Cambridge, MA, USA, 2018.
[5] M. Daun, J. Brings, T. Bandyszak, P. Bohn, and T. Weyer, “Collaborating multiple system instances of smart cyber-physical systems: a problem situation, solution idea, and remaining research challenges,” 2015 IEEE/ACM 1st international workshop on software engineering for smart cyber-physical systems, pp.48–51, IEEE, 2015.
[6] E.S. Kim and J. Yoo, “A study on application of stpa in safety analysis of platoon system,” 2020 Korea Conference on Software Engineering (KCSE 2020), Pyeongchang, Korea, pp.193–196, February 2020. in Korean.
[7] N. Leveson, Engineering a safer world: Systems thinking applied to safety, MIT press Cambridge, 2011.
[8] A. Abdulkhaleq and S. Wagner, “Integrating state machine analysis with system-theoretic process analysis,” Software Engineering Workshop, San Francisco, pp.501–514, 2013.
[9] J. Axelsson and A. Kobetski, “Towards a risk analysis method for systems-of-systems based on systems thinking,” 2018 Annual IEEE International Systems Conference (SysCon), pp.1–8, IEEE, 2018.
[10] X. Yang, I.B. Utne, S.S. Sandøy, M.A. Ramos, and B. Rokseth, “A systems-theoretic approach to hazard identification of marine systems with dynamic autonomy,” Ocean Engineering, vol.217, p.107930, 2020.
[11] S. Baumgart, J. Fröberg, and S. Punnekkat, “A state-based extension to stpa for safety-critical system-of-systems,” 2019 4th International Conference on System Reliability and Safety (ICSRS), Rome, pp.246–254, IEEE, November 2019.
[12] F. Fakhfakh, M. Tounsi, and M. Mosbah, “Vehicle platooning systems: Review, classification and validation strategies,” International Journal of Networked and Distributed Computing, vol.8, no.4, pp.203–213, 2020.
[13] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen, “A survey on platoon-based vehicular cyber-physical systems,” IEEE communications surveys & tutorials, vol.18, no.1, pp.263–284, 2015.
[14] S. Jung, E.S. Kim, and J. Yoo, “A traceability analysis for integrated relationship analysis of development/safety artifacts of cyber physical systems,” Journal of KIISE, vol.48, no.1, pp.107–118, 2021. in Korean.