Trace4PF: A tool for Automated Decomposition of Problem Diagrams with Traceability

Yajun Deng[†], Zhi Li[†*], Hongbin Xiao[†]
[†]School of Computer Science and Engineering, Guangxi Normal University, Guilin, China
Email: dengyajun@stu.gxun.edu.cn, zhili@gxnu.edu.cn, hongbinx1997@foxmail.com

***Abstract*—This paper provides a support tool for Jackson's Problem Frames approach – named *Trace4PF* for decomposing a global problem diagram into sub-problem diagrams. The tool provides a web browser interface with features such as drawing, editing and performing syntactical checking, and highlighting the trace of causal chain. A video demonstration of the tool is available at $https://youtu.be/XSUVGGqEKkw$.**

***Index Terms*—Cyber-Physical Systems, Problem Frames, Requirements Engineering, Decomposition and Traceability**

I. Introduction

Cyber-Physical Systems (CPS) play a crucial role in various fields and contain various elements (networks, sensors, actuators, displays, etc.) compared to the previous common systems [1]. Since CPS operate in an open, dynamic and diverse environment, their close interactions with the environment and users lead to many challenging problems. We argue, in this paper, that the most important challenge among these problems is a lack of an automated method or technique to reduce model complexity of such systems, and we provide a tool support for automatically decomposing a global problem diagram into sub-problem diagrams, once a traceability analysis is completed based on causal relationships elicited from stakeholders.

It is observed that Jackson's Problem Frames (PF) approach [2] is better suited for modeling, verifying and validating the complex contextual environments in relation to requirements of such systems. However, when faced with very complex CPS, the PF approach and existing tool support for modeling and analyzing the requirements in contexts usually encounter an overly large and global problem diagram, and fail to provide a traceability analysis based on causal reasoning, when decomposing the global problem into sub-problems.

This pape provides a support tool named *Trace4PF* for decomposing a global problem diagram into sub-problem diagrams, after syntactical checking, highlighting the trace of causal chains. The *Trace4PF* is a problem diagram tool that provides various editing and checking features for problem descriptions, which are prerequisites for problem decomposition with traceability analysis.

II. Tool Features

A. Problem Diagram Modeling

The GUI of *Trace4PF* is shown in Fig. 1. It is an online modeling tool for Problem Frames, in which stakeholders can draw their copies of problem diagrams using most mainstream web browsers. *Trace4PF* consists of three parts: (a) a canvas for the user to draw and edit; (b) predefined model elements for the user to draw and drop onto the canvas; (c) a toolbar that allows users to zoom the diagram to fit the current window, modify edge shape, upload or export files.

B. Syntactical Checking of Problem Diagrams

Once the diagram is completed by the user, a verification module is provided to help the user check the diagram for two types of syntactical errors.

Firstly, Syntax errors in labeling the phenomena and domain property. According to the grammars in PF [2], the format of the phenomena should be *DomainName!*{*phe1,phe2*}, so both *DomainName!*{*phe1phe2*} and *!*{*phe1,phe2*} are wrong, which must be corrected. The correct syntax of phenomena should follow the regular expression *regDP* below:

$$regDP = /\texttt{\^{}DomainName!\{(}phe'\texttt{*|}phe\texttt{)*\}\$}/ \tag{1}$$

$$phe = /\texttt{\^{}[a-zA-Z](\textbackslash w)*\$}/ \tag{2}$$

$$phe' = /\texttt{\^{}}phe\texttt{,\$}/ \tag{3}$$

In *regDP*, *DomainName* should be a name of a domain which is connected to the edge, and both *phe* and *phe'* are regular expressions as well. The domain property is defined using a regular expression named *regProp* as well as *regDP*:

$$regProp = /\texttt{\^{}(}phePhe'\texttt{*|}phePhe\texttt{)*\$}/ \tag{4}$$

$$phePhe = /\texttt{\^{}}phe\texttt{->}phe\texttt{\$}/ \tag{5}$$

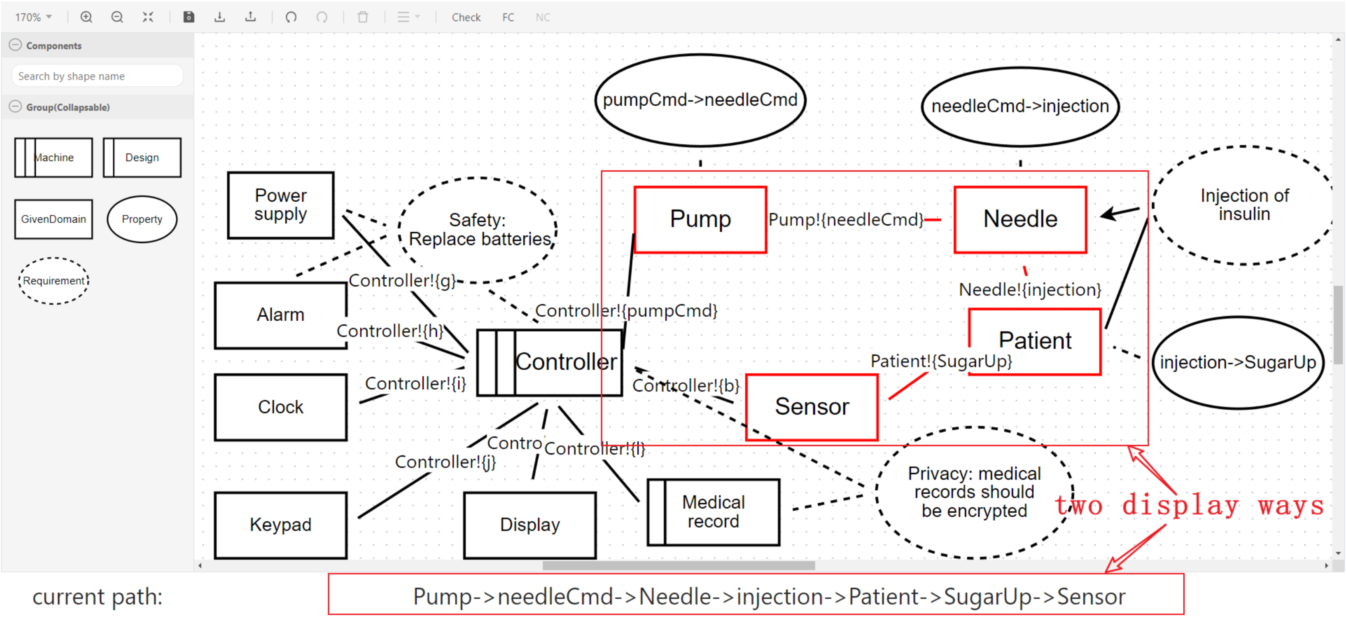
$$phePhe' = /\texttt{\^{}}phe\texttt{;\$}/ \tag{6}$$

If a domain or an edge has a syntax error, then the tool will show the domain or the edge and tell the user by highlighting the error in red.

Secondly, syntactical errors in circular causal relationships. In order to prevent circular causal relationships from reducing search efficiency. For example, in Fig. 1, phenomenon *pumpCmd* can indirectly evoke phenomenon *sugarUp*, and if *sugarUp* can directly or indirectly evoke *pumpCmd* at the same time (Suppose domain *Sensor* controls *pumpCmd* and *Sensor* has the property *sugarUp→pumpCmd*), then the tool would warn user that a loop exist in the diagram and show the loop.

*corresponding author: zhili@gxnu.edu.cn

DOI reference number: 10.18293/SEKE2022-181



current path: Pump->needleCmd->Needle->injection->Patient->SugarUp->Sensor

Fig. 1. The insulin control problem diagram modelled using our Trace4PF tool [3]

C. Decomposing Problem Diagrams with traceability

After the above syntactical checking is passed, the user can use and experience the core function of the tool – problem diagram decomposing with traceability. The current version of the tool provides the following three search options based on elicited causal relationships.

- Heuristic search. This method allows the tool user to explore all possible causal chains with traceable permissible paths (for reasons of space, this feature can be shown in the tool demo video).
- End-to-end search. This method shows all the permissible paths from a starting domain *Pump* to an ending domain *Sensor*, as shown in Fig. 1.
- Closed loop search. This method shows all the permissible paths starting from and ending at the Machine domain[1].

The tool can display all the permissible paths in both graphically and textually, as shown in Fig. 1.

III. Related Work

In recent years, a number of scholars have researched and developed problem framing tools. Some examples are as follows: Chen et. al. proposed DPtool [4], a tool to guide problem decomposition through scenario projection. Unlike our work, their tool has not yet implemented automated decomposition of problem diagrams with traceability. In this paper, we have extended the applicability of the current tool sets available to deal with more complex requirements inherent in CPS, with a web-based interface for ease of use.

IV. Conclusion

This paper presents the *Trace4PF* tool for automated decomposition of problem diagrams with traceability. The tool can be used through a web browser. The implementation of this tool uses AntV's open diagram editing engine X6 [2] – JavaScript Diagramming Library. In the future, we will continue developing this tool to provide more features, such as measuring the complexity of problem diagrams and automated test case generation.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (61862009), Guangxi “Bagui Scholar” Teams for Innovation and Research.

References

[1] Z. Jin, X. Chen, Z. Li, and Y. Yu, “RE4CPS: requirements engineering for cyber-physical systems,” in *27th IEEE International Requirements Engineering Conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019*, D. E. Damian, A. Perini, and S. Lee, Eds. IEEE, 2019, pp. 496–497. [Online]. Available: https://doi.org/10.1109/RE.2019.00072

[2] M. Jackson, *Problem frames: analysing and structuring software development problems*. Addison-Wesley, 2001.

[3] G. Liu, Z. Li, and Z. Ouyang, “CARE: A computer-aided requirements engineering tool for problem-oriented software development,” in *The 27th International Conference on Software Engineering and Knowledge Engineering, SEKE 2015, Wyndham Pittsburgh University Center, Pittsburgh, PA, USA, July 6-8, 2015*, H. Xu, Ed. KSI Research Inc. and Knowledge Systems Institute Graduate School, 2015, pp. 727–729.

[4] X. Chen, B. Yin, and Z. Jin, “Dptool: A tool for supporting the problem description and projection,” in *RE 2010, 18th IEEE International Requirements Engineering Conference, Sydney, New South Wales, Australia, September 27 - October 1, 2010*. IEEE Computer Society, 2010, pp. 401–402. [Online]. Available: https://doi.org/10.1109/RE.2010.58

[1] this is the only case where circular causal relationship is allowed starting from and ending at the *Machine* domain

[2] https://x6.antv.vision/zh