A Proposal for Software Accountability of Law Enforcing Information Systems

Ryo HAYASAKA and Koichiro OCHIMIZU
Japan Advanced Institute of Science and Technology
School of Information Science
1-1 Asahidai, Nomi, Ishikawa 923-1292 Japan
{ryoh,ochimizu}@jaist.ac.jp

Abstract

Everyone in our society must obey law. Similarly, in e-society, information systems which play the role of enforcing law must obey law. A law usually defines its purpose, conditions, and activities/procedures. In this paper, we propose a concept of Software Accountability of Law Enforcing Information Systems (LEISs) and show its realization method.

Software accountability is that a LEIS itself can explain the reason why it makes decisions or calculated results to the user of the system. The explanation is generated by using the laws relating the decisions or results and the empirical data, namely execution history of the LEIS.

We expect that the concept of software accountability can be applied to global software development.

1. Introduction

In recent years, the range of computerization of social systems has been rapidly expanding including e-governments/local-governments to form e-society. Our daily life heavily depends on such e-society systems. Therefore the system should be designed, implemented, operated, and maintained to assure us that e-society systems are dependable and trustworthy in addition to the traditional services.

Katayama [3] proposed the five requirements for trustworthy e-society in the 21st century COE program "Verifiable and Evolvable e-Society." They includes accountability and evolvability. Our research target is to define and realize software accountability and ease-of-evolution (evolution with low cost).

In our society, there are a lot of laws, regulations, and rules of some organizations we must obey. We call them social rules. The e-society system should support the application of some specific social rules, and such a system should be constructed to satisfy the social rules fully. Moreover, we need to certify and confirm that the e-society system is made satisfying the corresponding social rules correctly. Our society is always evolving, and social rules should be changed to catch up the evolution. The e-society system should evolve immediately after the change of social rules. So we should be able to evolve the e-society system with low cost. We call e-society systems that have the features mentioned above Law Enforcing Information Systems (LEISs).

2. Reconsideration of accountability

A concept of accountability has been applied to wide variety of research areas, such as software development process and traceability [2, 6], security and privacy for accessing data [7], distributed systems and middleware, and multi-agent and responsibility.

Eriksén [1] explores the concept itself by comparing three papers in HCI and CSCW literature which are associated with using ethnomethodology as a research approach. In the analysis, Eriksén gives two definitions of accountability: The first, *answerable*, that is responsible for giving an account as one's acts. The second, *explainable*, that is capable of being account for. The former is stronger then the latter because an acting subject, in the former case, has to answer by itself when it is asked and the latter case only requires to record a subject's state or quality.

Most of the research related to accountability mentioned above uses the latter. Johnson [4] uses the former and proposes a method to answer decisions made by a policy framework. To generate answers, policy rules are described in RDF and a reasoner inferences the rules.

We target the former and do not take a reasoning approach to generate answers. Instead, we take an approach which uses execution history of a LEIS and tree-structured law.

Eriksén uses the agenda "Accountability – for whom" to consider what is it. This is an important view to define software accountability so that we discuss a stakeholder analysis in the next section.

3. Software accountability

Accountability for each stakeholders differs, hence we define software accountability intuitively as follows:

> **Software Accountability** is that a LEIS itself can explain the reason why it makes decisions or calculated results to the user of the system.
>
> In other words, a LEIS should answer the question from the stakeholders who have some doubts about the decisions or value of calculation made by the LEIS. The LEIS needs to make the answers using its execution history to satisfy the stakeholders of the LEIS.

Here we used the word stakeholder to represent: people who made the social rule; people who developed the LEIS; people who operated the LEIS to show output of the LEIS; people who received the results from the LEIS; and so on. We show two examples of LEISs and their stakeholders below.

Local government has a lot of regulations. There are several types of stakeholders such as lawmakers, civil service, system developer, citizens. A company has a lot of in-house regulations based on its management policy. There are several types of stakeholders such as managers and employees.

In this paper, we focus on software accountability for people who received the results from the LEIS.

4. Realization approach

We organize the semantics of the world understood by stakeholders as a goal-oriented tree in goal-oriented requirement engineering approach [5]. It is possible to define and describe the world intended, understood, and represented by lawmakers. We call this tree an Accountability Tree.

Figure 1 shows the reference architecture, assuming a LEIS adopts web-based three tier model. The logging agent always monitors how the LEIS is working and records its execution history. The accountability module has the functionality of generating an answer for a question from the stakeholder who doubts about the decisions made by the LEIS. The answer consists of the origin of the law, the law itself, and data expressing the stakeholder's situation, relating to the question.

5. Conclusions

We proposed software accountability which features stakeholders of LEISs. Software accountability is realized by using execution history of a LEIS and tree-structured law.

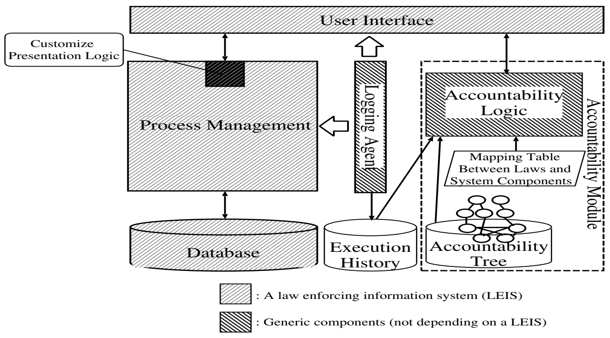
User Interface
Customize Presentation Logic
Process Management
Logging Agent
Accountability Logic
Accountability Module
Mapping Table Between Laws and System Components
Database
Execution History
Accountability Tree
: A law enforcing information system (LEIS)
: Generic components (not depending on a LEIS)

Figure 1. Reference architecture.

The concept of software accountability can be apply to global software development. In such development, projects on each site employ each process model, which defines development phases, procedures, workflows, etc. To improve software quality, actual development process of a project should follow the process model the project employs. This process model can be considered as "law" of the development process. Software Tag [2] is proposed to realize traceability, and can include the history of the actual development process. Thus, a system with software accountability can answer why an actual development phase is correct, by using law and history of development process.

Acknowledgements

Thanks to Professor Takuya KATAYAMA for his useful suggestion to promote this research. Thanks also to Professor Mitsuru IKEDA and Associate Professor Masato SUZUKI for their useful discussion.

References

[1] S. Eriksén. Designing for accountability. In O. W. Bertelsen, editor, *NordiCHI*, pages 177–186. ACM, 2002.
[2] K. Inoue and M. Barker. Accountability and traceability in global software engineering (ATGSE2007). In *APSEC*, pages 553–554. IEEE Computer Society, 2007.
[3] JAIST. The 21st century COE program "Verifiable and Evolvable e-Society". http://www.jaist.ac.jp/jaist-coe/.
[4] D. G. Johnson and J. M. Mulvey. Accountability and computer decision systems. *Commun. ACM*, 38(12):58–64, 1995.
[5] J. Mylopoulos, L. Chung, and E. Yu. From object-oriented to goal-oriented requirements analysis. *Communications of the ACM*, 42(1):31–37, 1999.
[6] Y. Wang. Approaches to accountability for offshore software development. In *Proceedings Workshop on Accountability and Traceability in Global Software Engineering (ATGSE2007)*, pages 19–20. Information Processing Sciety of Japan, 2007.
[7] D. J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. A. Hendler, and G. J. Sussman. Information accountability. *Commun. ACM*, 51(6):82–87, 2008.