Applying Evolutionary Prototyping In Developing LMIS: A Spatial Web-Based System For Land Management

W Agustiono
Department of Informatics Engineering, University of Trunojoyo Madura, Jl. Raya Telang, Kamal, Bangkalan, Madura 69162 Indonesia
Email: wahyudi.agustiono@trunojoyo.ac.id

Abstract. Software development project is a difficult task. Especially for software designed to comply with regulations that are constantly being introduced or changed, it is almost impossible to make just one change during the development process. Even if it is possible, nonetheless, the developers may take bulk of works to fix the design to meet specified needs. This iterative work also means that it takes additional time and potentially leads to failing to meet the original schedule and budget. In such inevitable changes, it is essential for developers to carefully consider and use an appropriate method which will help them carry out software project development. This research aims to examine the implementation of a software development method called evolutionary prototyping for developing software for complying regulation. It investigates the development of Land Management Information System (pseudonym), initiated by the Australian government, for use by farmers to meet regulatory demand requested by Soil and Land Conservation Act. By doing so, it sought to provide understanding the efficacy of evolutionary prototyping in helping developers address frequent changing requirements and iterative works but still within schedule. The findings also offer useful practical insights for other developers who seek to build similar regulatory compliance software.

1. Introduction

There is no exact definition found in the literature of what is considered as a Regulatory Compliance Software (RCS). Instead, some of different terms and words are used by practitioners to represent this concept of RCS such as "Compliance software," "Legal Compliance Software" and "Regulatory Reporting Software." In particular, this paper defines an RCS as a computer application solution designed to enable individual or organizations perform their tasks required for regulatory compliance. Examples of RCSs may include e-tax filing systems [1], e-rulemaking [2] and e-participation [3].

Based on the above general definition, developing an RCS inherently presents unique challenges which could lead to the risk of failure if they are not carefully nurtured for some reasons:

a. First, the software should adhere to and comply with the (stringent) provisions imposed by the relevant laws and regulations which may change during the development process. As a consequence the developers may take frequent iterative works and could face a risk of failure if they are unable to freeze the requirements within the project time.

b. Second, if the software uses sensitive and personal information, then the developers need to ensure that the development process and the resulting systems must comply with various standards such as security, cryptography and privacy.
c. Third, in the case of developing RCS to support the enforcement of a newly passed law, the developers may need to build the software from scratch since there are no comparable software systems that match with such recent legal requirements. Therefore, the developers need to take iterative review to ensure the software meets with specified requirements. Despite its unique nature, there is limited knowledge or technical guidance on how can developers build an RCS successfully. With regards to this gap, this study aimed to examine the implementation of one of software development methods called evolutionary prototyping in developing an RCS.

2. Related Works
There has been growing number of works found in the literature researching the development of RCS due to the increasing demand on the software systems that adhere to legal compliance and accountability imposing by regulations or laws. A great deal of previous research in this area has focused on providing various frameworks and techniques that enable software developers to identify and analyze legal obligation and the refine into software requirements [4-6]. All these studies, while, provide us insights into the techniques used to gather the inevitable changing requirements of RCS development. Nevertheless, some interesting issues remain unclear from previous studies such as what method can help developers effectively incorporate the unknown requirements into the design and how to manage the unexpected change requirements during the development.

For several years, a great effort has been devoted studying the implementation of various software development methodologies. Especially since the introduction of Water fall model early 1970s to late 1980s, software development methodology has attracted much attention from software engineering researchers [7]. Despite its potential use, using waterfall solely may not work efficiently, especially in the software development with high complexity and frequent change in requirements. To overcome this, researchers and practitioners have proposed more agile development methods and some become popular for its applicability such as evolutionary prototyping, scrum and XP [8]. However, there is no method has been a silver bullet which can address all the problems in the software development. Instead, each of which may have strong and weakness and can complement each other.

As introduced above, this study aimed to investigate the implementation of Evolutionary Prototyping (EP) in a software development project. Although there has been a large volume of published studies in this area since several decades ago, current empirical researches describing the use of EP continue to progress [e.g. 9, 10]. However, due to the dynamic and constant change especially on today's software development landscape, there is a need for more study on the implementation of EP. Furthermore, as each software development project has different characteristics, it suggests that research on the implementation of EP within specific case could make a unique contribution to the current body of knowledge. For this reason, to advance our understanding and make an important contribution to the current knowledge, this research particularly looks at the implementation EP in a unique context of Regulatory Compliance Software (RCS) development project.

3. Proposed framework
Given the unique challenges and circumstance surrounding the development of RCS, this research considered EP informed in the previous work [11] as an appropriate method. This is because, in the EP, developers start by designing and implementing the most prominent prototype and then continue adding to and refining the prototype based on user feedback and eventually evolves until agreed for release [11]. Figure 1 illustrates how this research adopted EP to build RCS comply with the existing provisions imposed by the relevant laws and regulations. It also shows how EP enables us to incorporate new requirements in the iterative prototype development.
4. Applying the framework

This section follows on from the previous section, which applied the overarching framework to investigate an RCS development. Before employing the framework, it is necessary to briefly describe the case study to give an overview of the RCS investigated in this study. The description also provides an account of aspects of the project, which had relevance to the unique nature of RCS development.

4.1. The case study

This research looks at software designed for complying regulation. In particular, it investigates the development of Land Management Information System (LMIS, pseudonym) initiated by a state Government in Australia. LMIS was chosen as an excellent case study for some reasons. First, the system was mainly designed as spatial web-based to enable farmers visualizing and measuring practically their current soil condition resulting farm practices (e.g. pesticide application, fertilizing and crop rotation) and then assist them in prioritizing necessary treatment plan as requested by Soil And Land Conservation Act. As such, there is a need for aligning this regulatory provision into the system for attaining compliance issues which matches with the nature of RCS discussed above.

Second, the government as the owner made LMIS as mandatory tool for various stakeholders in the agricultural sector (e.g., industry, agronomists, and farmers) who wish to access government-owned such as natural resource and soil map. The diverse range of stakeholders indicates that design process could present the complexity especially on how to define and incorporate those different and conflicting needs into acceptable requirements as it is usually found in the RCS.

Finally, at the time when the LMIS design project was initiated, there were no comparable or similar tools existed that gave users ability to record and share soil condition with other parties. Developing such new system also means that the requirements can be hidden, unknown and imperfect at initial phase but then become obvious, growing or even changing when it comes to the implementation. Therefore LMIS development is a suitable case given the uncertainty and changing of requirements which are similar to RCS. To address the challenges identified above, the LMIS design was undertaken using evolutionary prototyping which involves three processes as described below.

4.2. Phase I: Requirements elicitation and analysis

Requirements elicitation is one of the critical parts in any software development, without which the developers are unable to produce expected system and comply with given regulation. Likewise in the LMIS design project, this phase was where developers attempted to define what the system should do. Using the framework illustrated in Figure 1 above, it can be thus explained that the process of requirements elicitation and analysis in the LMIS design project was started by reviewing related regulations and laws relating to the soil and land protection. Using legal text analysis, the developer team then identified and discussed law's elements that were transferable into technical or system specification. All the agreed requirements were then documented into Technical Requirement Documents such as System specification, Uses cases, DFD, CDM, and PDM. In some cases, the developer might receive feedback or additional information from the users that were useful for revising the requirements. Once the revisions were approved, the updated requirements documents were then used in the next phase for developing a working prototype as illustrated in Figure 2.
4.3. Phase II: Prototype development

Once the requirements were approved, the developer team started the second phase which aimed to develop a prototype. This phase mainly involved software building activities such as designing and developing architectural, database, module and user interface for LMIS based on the requirements specified in the technical documents. All the activities might be repeated until the software process design produce a working prototype which would be ready for internal verification by developer team. The prototype was modified if the result from the verification process was unsatisfactory. But, if the result was satisfactory and the prototype worked as expected in the requirements, the revised prototype then went through evaluation involving potential users (note: the evaluation process including how it was conducted and who are the participants will be discussed more thoroughly in section 4.4).

At the end of the evaluation, if the developer team and the user participants felt that the prototype was not satisfactory, then the software building process entered new iteration for aligning additional or new requirements as well as responding to feedbacks provided by users into the previous prototype. Having completed each development cycle, a newly evolved prototype was produced and sent to users for another round of evaluation. This cycle was still performed until the evolved prototype met the requirements and satisfied the users' needs or expectation. At this stage, if there were no more feedback or addition requirements to consider, the LMIS development was completed and ready to be delivered or released as shown in Figure 3.

4.4. Phase III: User evaluation

As described in Section 4.3, when the initial prototype development was done, an intensive evaluation was run involving government staffs and farmers in trial feedback sessions. It aimed to seek how easy the prototype was to use; how this could be improved; and which parts were useful especially for measuring, monitoring reporting current soil condition requested by the government. In addition, the evaluation was also aimed at determining its suitability for farmers to comply with soil management act and how its usability might be enhanced.
number of issues were identified which had reduced its suitability for farmers. The government staffs also suggested the need to incorporate nutrient calculator into the prototype. This feature could enable farmers to measure the optimum fertilizer quantities they should apply for paddocks to minimise excess amounts which resulted in wasted money and soil nutrient degradation. Further, it could also help them without expert assistance to reduce the paperwork or administrative burden of collecting, storing and reporting on soil nutrient index at their farm to potential regulators to reduce the likelihood of being audited but still demonstrate managing environmental risk factors.

Therefore in response to the evaluation results, the developer team planned to incorporate the additional requirements and recommendation in the next cycle round of development process (see section 4.3 above on how this iterative prototype development cycle was carried out) and aimed to produce an enhanced version of the prototype. Having completed a few cycles of prototype development-evaluation and when there were no more new requirements to add, the developer team concluded that the LMIS was complete, and was ready to be delivered to the government as the owner for further fully function release and roll out as shown in Figure 4.

5. Conclusion and further work

Overall, this study finds that the development of LMIS using EP has enabled the government to provide a tool for collecting, measuring and sharing soil condition practically. In particular, this shows that EP is potentially suitable for developing RCS under the changing and ambiguity environment (see Figure 5). EP is also found can reduce defect, facilitate end user participation and produce an acceptable system. The analysis also identified challenges encountered during the implementation of EP such as delayed user feedback, difficult to freeze requirements and required the skilled team. To conclude, there is no method has been a silver bullet which can address all the problems in the software development, but EP can be very useful for developing software with highly changing requirements. Future study can address this by combining design methods for improving the development process and producing the quality software.

Figure 5. Applying EP throughout LMIS development project

References

[1] Stafford TF, Turan AH. Online tax payment systems as an emergent aspect of governmental transformation. European Journal of Information Systems. 2011;20(3):343-57.
[2] Deligiaouri A. Open governance and e-rulemaking: online deliberation and policy-making in contemporary greek politics. Journal of Information Technology & Politics. 2013;10(1).
[3] Kim S, Lee J. E participation, transparency, and trust in local government. Public Administration Review. 2012;72(6):819-28.
[4] Islam S, Mouratidis H, Jürjens J. A framework to support alignment of secure software engineering with legal regulations. Software & Systems Modeling. 2011;10(3):369-94.
[5] Breaux TD, Antón AI, Spafford EH. A distributed requirements management framework for legal compliance and accountability. computers & security. 2009;28(1):8-17.
[6] Ingolfo S, Siena A, Mylopoulos J, Susi A, Perini A. Arguing regulatory compliance of software requirements. Data & Knowledge Engineering. 2013;87:279-96.
[7] Raccoon L. Fifty years of progress in software engineering. ACM SIGSOFT Software
Engineering Notes. 1997;22(1):88-104.

[8] Dingsøyr T, Lassenius C. Emerging themes in agile software development: Introduction to the special section on continuous value delivery. Information and Software Technology. 2016;77:56-60.

[9] Chen H-M, Kazman R, Haziyev S. Strategic prototyping for developing big data systems. IEEE Software. 2016;33(2):36-43.

[10] Guerrero-García J. Evolutionary design of user interfaces for workflow information systems. Science of Computer Programming. 2014;86:89-102.

[11] McConnell S. Rapid development: taming wild software schedules. Pearson Education; 1996.