Proceeding Temu Ilmiah XI, 2002 ISSN 0918-7685
pg. 1-7

On the Requirements Pattern of Software Engineering

Romi Satria Wahono
Department of Information and Computer Sciences, Saitama University
Indonesian Institute of Science (LIPI)

E-mail: romi@aise.ics.saitama-u.ac.jp

Abstract

Nowadays, patterns are one of the latest hot topics in software development. The goal of patterns within the software community is to create a body of literature to help software developers resolve recurring problems encountered throughout all of software development. This paper illustrates the efforts for applying the patterns paradigm in requirements engineering. The current research focusing on requirements pattern is also described.

Key words: requirements engineering, requirements pattern.

1. Introduction

Nowadays, patterns are one of the latest hot topics in software development. The goal of patterns within the software community is to create a body of literature to help software developers resolve recurring problems encountered throughout all of software development. Patterns help create a shared language for communicating insight and experience about these problems and their solutions.

Patterns have been used for many different domains ranging from organizations and processes to teaching and architecture. Today, software patterns are readily available in the field of analysis pattern, data pattern, design pattern, fundamental pattern, message pattern, and transaction pattern. This paper illustrates the efforts for applying the patterns paradigm in requirements engineering. The current research focusing on requirements pattern is also described.

2. Terminology and Concepts of Patterns

The current use of the term pattern is derived from the writings of the architect Christopher Alexander who has written several books on the topics as it related to urban planning and building architecture [Alexander-77] [Alexander-79]. Although Christopher Alexander was talking about architecture and urban planning, the patterns that he talked about are applicable to many other disciplines, including software development.

Software patterns became popular with the wide acceptance of the book *Design Patterns: Elements of Reusable Object-Oriented Software* by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides [Gamma-94]. Patterns have been used for many different domains ranging from organizations and processes to teaching and architecture. At present, the software community is using patterns largely for software architecture and design, and software development processes and organizations. Other recent books that have helped

popularize patterns are: *Pattern-Oriented Software Architecture: A System of Patterns* by Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal [Buschmann-96], and the books *Pattern Languages of Program Design* [Coplien-95], which are selected papers from the conferences on Patterns Languages of Program Design.

So what is a pattern anyway? Some researchers have created the definitions of a pattern and explained well in their books and papers. Dierk Riehle gives a nice definition of the term pattern [Riehle-96]:

> *A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts.*

In [Ferdinandi-02] a nice definition of the term pattern is given:

> *A pattern is any reusable template based on experience that can be used to guide the creation of a solution to the problem or need in a specific context.*

Christopher Alexander says [Alexander-77]:

> *Each pattern describes a problem which occurs over and over again in our environment and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it in the same way twice.*

And a definition, which more closely reflects its use within the pattern community, is:

> *A pattern is a named nugget of instructive information that captures the essential structure and insight of a successful family of proven solutions to a recurring problem that arises within a certain context and system of forces.*

However, a pattern is a solution to a problem in a context. It is based on the following observations [Fredj-99]:

- Some problems are recurrent in different applications.
- The good solutions should be shared.
- A pattern capitalizes an experience and supports its communication to other designers.

According to James O. Coplien [Coplien], a good pattern will do the following:

- *It solves a problem*: Patterns capture solutions, not just abstract principles or strategies.
- *It is a proven concept*: Patterns capture solutions with a track record, not theories or speculation.
- *The solution is not obvious*: Many problem-solving techniques (such as software design paradigms or methods) try to derive solutions from first principles. The best patterns generate a solution to a problem indirectly--a necessary approach for the most difficult problems of design.
- *It describes a relationship*: Patterns don't just describe modules, but describe deeper system structures and mechanisms.
- *The pattern has a significant human component (minimize human intervention).* All software serves human comfort or quality of life; the best patterns explicitly appeal to aesthetics and utility.

3. Requirements Pattern

As shown in Figure 1, requirements engineering is an iteration process, which contains requirements elicitation, requirements specification, and requirements validation processes. The core idea of using pattern in requirements engineering is based on this process. That is to identify a set of patterns that can serve as nucleus to elaborate the system analysis. The analyst can select components that he judges relevant for its application and adapt them to its context. Each pattern highlights the problem, its context and its solution expressed in a semiformal language.

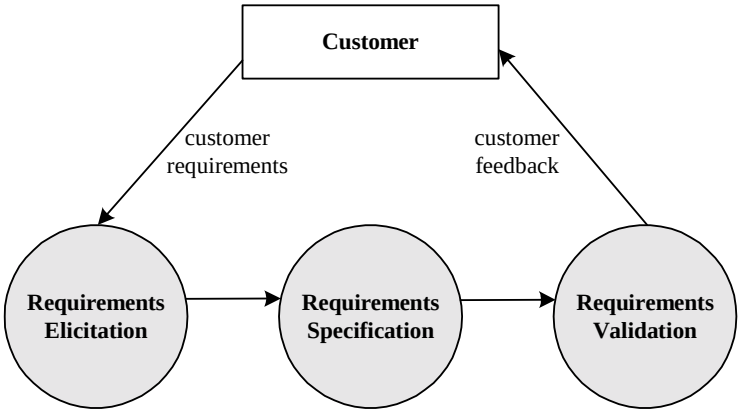

Figure 1. Requirements Engineering Process

A requirements pattern is a framework for requirements that supports the product needs, minimizing gaps in knowledge that may cause project failure. A requirements pattern supports multiple design and implementations. A requirements pattern also supports any technique for capturing and specifying requirements. It is reusable framework based on experience that can be used to identify pieces of the needs that a solution must satisfy. Patterns propose generic and reusable solutions to the recurring problems of requirements engineering. However, to develop the design for reuse it is necessary to have a library of reusable components. The management of this library is an independent process including tasks such as: identification of useful and good quality components and rewriting of these components to improve their genericity and legibility, their maintenance and diffusion.

So, we can conclude that the advantages of the requirements engineering based on patterns approach are essentially the facts of:

- Capitalizing the now-how in requirements engineering
- Enhancing solution mining due to the patterns genericity
- Simplicity and formalism
- Ensuring flexible use because one has not to apply a method in its globality
- Speed the specification process
- Ease the consumers understanding of the specifications
- Subsequent design documents generated from the specifications capture the spirit of the user’s needs without any intervention

- Patterns propose generic and reusable solutions to the recurring problems of requirements engineering

4. The Current Researches Focusing on Requirements Pattern

Recently, there are few efforts and research topics focusing on the idea for applying pattern paradigm in requirements engineering. We summarize these researches as follows.

4.1. Christopher Creel

Christopher Creel [Creel-99] presented 3 patterns for documenting requirements:

- *Specify*: to specify how an actor can identify an object, reducing the coupling between the method to find an object and the operations that are performed on that object.
- *Presentation*: to describe the data that an application must present, enabling you to focus on the information to display, not how to display it.
- *Prioritize*: to communicate the urgency of one application aspect over another without specifying the exact implementation for communicating that priority.

He successfully used each of these patterns on at least four separate requirement specifications. Each specification addressed different problem domains, yet the requirement patterns helped to speed the specification process in all four instances. The consumers were the first to comment that the requirement patterns eased their consumption of the specifications. The subsequent design documents generated from the specifications captured the spirit of the user's need without the requirements engineering's intervention.

4.2. Sascha Konrad and B. H. C. Cheng

Sascha Konrad and B. H. C. Cheng [Konrad-02] describe research into investigating how reuse can be applied to requirements specifications, which they term requirements patterns. They explore how object-oriented modeling notations; such as the Unified Modeling Language (UML) can be used to represent common requirements patterns. Structural and behavioral information are captured as part of a requirements pattern. They focus on requirements patterns for embedded systems.

Sascha Konrad and B. H. C. Cheng claimed that developed requirements patterns have three major impacts in the development of embedded systems.

- Facilitated the quick construction of requirements models for different automotive embedded systems. Having both structural and behavioral information further facilitated the modeling process.
- Prompted the developer to think about aspects of a system that might have been otherwise overlooked until much later in the development process, such as fault tolerant and safety considerations.
- The use of these patterns made the systems more uniformly structured, thus facilitating the understanding and the evolution of the systems.

4.3. Mounia Fredj and Ounsa Roudies

Mounia Fredj and Ounsa Roudies [Fredj-99] [Fredj-01] developed a pattern language to assist requirements elicitation and to guide the analysis of an organization. Its objectives are:

- to represent these three systems main elements which are useful for an organization evaluation,
- to justify requirements choices,
- to guide the analysis,
- to offer a support to the communication with the final user.

Their selection and adaptation depends on the analyst way of working.

4.4. Romi Satria Wahono

Romi Satria Wahono proposed the extensible requirements patterns, especially for web applications [Wahono-02]. He argued that web application development process and time to market must be highly accelerated, regarding to the growth of web application to be an e-business application. He also investigated that there are a lot of redundant works actually occurred in web application development. A lot of web applications actually have similar basic requirements, although have very different in design. By reusing and reconstructing these extensible requirements patterns, the time to market can be reduced and the redundancy problems can be solved (Figure 2).

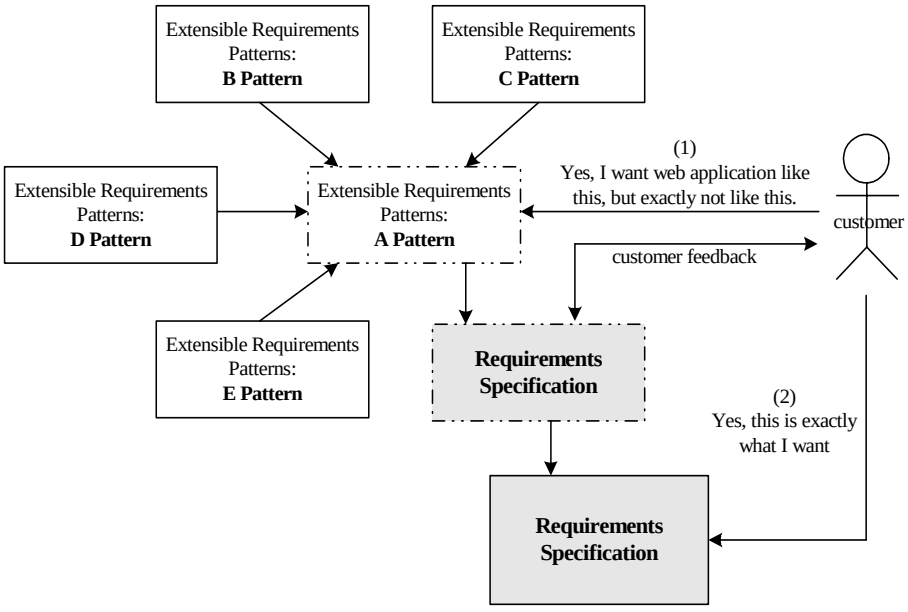

Figure 2. The Core Concepts of Extensible Requirements Patterns

An extensible requirement pattern is a reusable and an extensible framework for requirements that supports a system analyst to validate the customer feedback (and needs), by reusing and reconstructing requirements patterns. An extensible requirements pattern has

some features related to the extensibility characteristic. Not like other existing patterns, an extensible requirements pattern has two kinds of pattern elements: constraint elements and extensible elements.

5. Conclusion

Recently, there are few efforts for applying pattern paradigm in requirements engineering. The core idea is to identify a set of patterns that can serve as nucleus to elaborate the system analysis. The analyst can select components that he judges relevant for its application and adapt them to its context. Each pattern highlights the problem, its context and its solution expressed in a semiformal language. In this paper we illustrated the efforts for applying the patterns paradigm in requirements engineering. The concepts and current research focusing on requirements pattern are also described.

Acknowledgements

The author would like to thank to the Ministry of Education, Culture, Sports, Science and Technology of Japan for its financial and scholarship support.

References

[Alexander-77] Christopher Alexander, Shara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel, “A Pattern Language: Towns, Bulding, Construction”, *Oxford University Press*, 1977.

[Alexander-79] Christopher Alexander, “The Timeless Way of Building”, *Oxford University Press*, 1979.

[Buschmann-96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal, “Pattern-Oriented Software Architecture: A System of Patterns”, *John Wiley & Son*, 1996.

[Coplien] James O. Coplien, “Software Pattern”, *Patterns Home Page*, http://hillside.net/patterns/definition.html

[Coplien-95] James O. Coplien, Douglas C. Schmidt, Jim Coplien (Editors), “Pattern Languages of Program Design”, *Addison Wesley*, 1995.

[Creel-99] Chris Creel, “Requirements by Pattern,” *Software Development Magazine*, December 1999.

[Ferdinandi-02] Patricia L. Ferdinandi, “A Requirements Pattern: Succeeding in the Internet Economy”, *Addison Wesley*, 2002.

[Fredj-99] M. Fredj, and O. Roudies, “A Pattern Based Approach for Requirements Engineering”, *Proceeding of the 10th International Conference and Workshop on Database and Expert Systems Applications*, IEEE, Italy, 1999.

[Fredj-01] M. Fredj, and O. Roudies, “A Reuse Based Approach for Requirements Engineering”, *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications 2001*, IEEE, 2001.

[Gamma-94] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software”, *Addison Wesley*, 1994.

[Konrad-02] Sascha Konrad, and B. H. C. Cheng, “Requirements Patterns for Embedded Systems”, *Technical report number: MSU-CSE-02-4*, February 2002.

[Riehle-96] Dirk Riehle and Heinz Zullighoven, "Understanding and Using Patterns in Software Development", *Theory and Practice of Object Systems*, Vol. 2, No. 1, pp. 3-13, 1996.

[Wahono-02] Romi S. Wahono and Jingde Cheng, "Extensible Requirement Patterns of Web Application for Efficient Web Application Development", *Proceedings of the 1st International Symposium on Cyber Worlds: Theories and Practices (CW2002)*, Tokyo, Japan, November 2002.

Biography of Author

Romi Satria Wahono, Received B.E. and M.E degrees in Information and Computer Sciences in 1999 and 2001, respectively, from Saitama University, Japan. He is currently a Ph.D. candidate at the Department of Information and Computer Sciences, Saitama University, Japan and also a researcher at the Indonesian Institute of Science (LIPI). His current research interests include Software (Requirements) Engineering, Web Engineering, and Object-Orientation. He is a member of the ACM, IEEE Computer Society, The Institute of Electronics, Information and Communication Engineers (IEICE), Information Processing Society of Japan (IPSJ), Japanese Society for Artificial Intelligence (JSAI), and Indonesian Society on Electrical, Electronics, Communication and Information (IECI).