Semantic Web As An Enabling Technology For Better Design Pattern Adoption

Luka Pavlič, Marjan Heričko
Faculty of Electrical Engineering and Computer Science
University of Maribor
Smetanova 17, SI-2000 Maribor, Slovenia
{luka.pavlic, marjan.hericko}@uni-mb.si

Abstract. *Design patterns are a high level approach for reuse in software engineering area. The number of design patterns is rising rapidly, which makes selecting appropriate one harder task. This issue is especially clear for less experienced developers.*

In this paper we present our approach to address the presented issue of managing and selecting design patterns - an experiment prototype of a new design pattern repository, based on semantic web technologies. New Ontology-Based Design Pattern Repository is currently a work in progress. This is why we point out only its potentials for improving design pattern adoption.

Keywords. Design Patterns, Semantic Web, Ontologies, Design Pattern Repository.

1 Introduction

It is important to develop new systems from existing, already proven reusable elements in all engineering disciplines. This enables engineers to use well known approaches and best practices. Reuse has become an essential and important strategy in the software development area also. The reuse of concrete software elements such as functions, classes and components is already well established and practiced on a daily basis. However, if we observe reuse at higher levels of abstraction i.e. software patterns, the reuse is still not well established.

A pattern is a form of knowledge for capturing a recurring successful practice [10]. Design patterns capture the best practices for solving recurring software design problems and are proven way to build high quality software [5]. They capture knowledge that experienced developers understand implicitly and facilitate training and knowledge transfer to new developers [17]. A survey [13] indicated a low adoption of design patterns among practitioners – respondents estimated that no more than half of the developers and architects in their organization use software patterns. Therefore bridging the gap between the pattern expert communities and the typical pattern user is critical for achieving the full benefits of software patterns [4].

Authors have observed that it might be difficult to find a suitable design pattern even in a catalogue such as the GoF (Gang of Four) Design Pattern Catalogue [5] – with no more than 24 patterns. Several hundred software patterns have already been published. The Pattern Almanac [11] published in 2000, provided a list of over 700 previously published patterns organized into 70 categories. Software developers experience more and more problems finding patterns that match their design problems. Useful patterns might therefore easily be overlooked. Current methods of reuse on design pattern level obviously tend to be intuitive and based on the experience of the individual software engineer. We want to fill a gap between unmanageable number of design patterns and developers searching for a suitable pattern in their current design problems. Since we have many positive experiences in initiating developers to use design patterns we decided to develop an integral web-based tool for selecting design patterns. The tool (Ontology-Based Design Pattern Repository – OBDPR) presented in this paper is a basis for automatic and intelligent services, used to help in selecting appropriate design patterns. It introduces capabilities known from artificial intelligence area to improve the efficiency of design pattern selection used in the software development. The OBDPR prototype includes all lessons learned in previous attempts to improve design patterns adoption. When OBDPR is fully functional we are planning to perform several experiments to indicate if and how much our effort helps software engineers, especially inexperienced ones.

2 Semantic web technologies and design pattern representation

In general there are three main categories of representing design patterns:

- informal representations,
- semiformal representations based on graphical notations such as UML and
- various formal representations which also include notations using semantic web technologies.

Informal representations mainly include printed catalogs, which are traditionally used to describe design patterns using natural language [5]. Because of its loose structure this kind of representation is less suitable for knowledge management and sharing [7]. Those representations also do not enable desired and needed level of design pattern identification and application. This is why there are several forms of formal presentation, mainly by extending UML specification [15][6][8][3][12]. Those representations are efficient for a basic understanding of patterns since they cover their structural elements. But they do not provide information and knowledge on high level aspects such as intent, usability and consequences.

Some authors [7][8] talking about design pattern representations take a standpoint similar to the semantic web (to keep data semantically understandable both to human and machine). Their representation is based on ontologies. It is used primarily to describe structure of source code, which is done according to particular design pattern. One of those in used in “Web of Patterns” (WoP) project [1] has addressed the area of describing knowledge on design patterns. The main focus of WoP project is on analyzing Java source code for used design patterns identification. One of the project results is also ontology for describing design patterns on source code level, which is not sufficient in our case.

After reviewing related work and benefits of using ontologies as explained below we also decided to use them in our work. Although there are some ontologies in the community, we did not use any existing one. We rather develop our own with interoperability in mind.

The idea of semantic web allows automatic, intelligent inferring on knowledge, represented using ontologies. The basic idea of semantic web is a different organization and storage of data and consequentially new possibilities to use this data [19]. The barrier that prevents more advanced usage of available data is believed to be the semantic poorness of today’s solutions. The vast majority of data is presented as a very simple, non-structured human readable and human understandable material. The result is the inability to make a real use of the enormous amount of “knowledge”. In order to overcome these difficulties the concept of meta-data is introduced in the core of the semantic web. Using meta-data, so called smart agents can be used to search for information by content and to infer on the gathered concepts. As a foundation, there has been a lot of work done about common formats for interchange of data and common understanding of common concepts. That allows a person to browse, understand and use data in a more straightforward way, and a machine to perform some intelligent tasks on the data automatically. Furthermore, the semantic web ideas can be used in an internal enterprise information system for knowledge management in a different way to introduce new intelligent services. In semantic web, knowledge is represented as graphs, written down in XML-based language called RDF (Resource Description Framework) [16]. RDF is dealing with URIs (another W3C standard for naming resources globally unique). Advanced use of semantically annotated data can only be accomplished by using ontologies represented as RDFS or OWL [14] documents. There is also a language for efficient querying the RDF-represented knowledge, SPARQL [18]. The whole stack of semantic web technologies is available and described in [20].

3 Role of ontology in OBDPR

As mentioned before we decided to develop our own ontology. One could argue having a separate ontology is problematic. It is actually not a problem, since there is a possibility to connect ontologies in a quite straightforward way. Ontology from WoP project [1] is oriented towards supporting design pattern scanner, which does not help us a lot while being oriented towards design pattern selection.

One of the enabling approaches used in semantic web is ontology, as described before. Ontology describes the subject domain using notions of concepts, instances, attributes, relations and axioms. Concepts can be organized in taxonomies through which inheritance mechanisms can be used in ontology. Ontology adds semantics to the model representation. Their formal, explicit and shared nature makes them an ideal choice for design pattern repository.

With the presented facts we also justify our decision to use ontologies as well as other semantic web technologies to provide a basis, not only for design pattern description, but also for future intelligent services:

- Ontology-based design pattern descriptions are computer readable and therefore suitable for automated (computer) processing.
- Transforming OWL and RDF based design pattern representation into other kind of

representations (textual or graphical form) is not an issue.

- Ontology and related technologies are well established, recognized, extendable and based on standards.
- They enable exchangeability of design pattern descriptions in straightforward way.
- Ontology supported knowledge base is distributed by default.
- Third-party ontology (OWL)–enabled tools are available and will be developed in future which can extend the use of ontology enabled knowledge base.

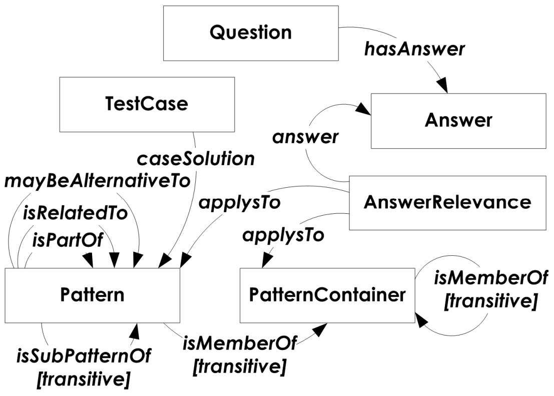


Figure 1. Core of OBDPR ontology

OBDPR underlying ontology is implemented using OWL. A core ontology fragment is shown in figure 1. We use hierarchical organization of pattern containers. Every pattern container may contain several pattern containers and patterns. This enables us to capture several divisions of design patterns, not only those found in fundamental literature. Every pattern can be included in several containers; the same is true for containers. Patterns themselves are connected in a more logical way in means of related, similar, composed patterns and pattern hierarchies. Not only patterns and pattern containers themselves are included in ontology, but there are also real-world examples using patterns to give more meaning to OBDPR user (“TestCase” class).

Design pattern experts can provide experiences in question-answer pairs, which enables them to capture their implicit knowledge on design patterns. Not only experts can give experiences to tell which design pattern is used in particular real-life situation (“Question” class), but they can also specify more possible solutions to real-life situation (“Answer”) with specified probability (“AnswerRelevance”). This is value in range from 0% to 100% and tells user how likely is that particular candidate (“Pattern” or “PatternContainer”) is used when answer to given question is confirmed positive. Answers and possible candidates can easily be updated or added to questions at any time with rich user-friendly web interface.

4 Current state of OBDPR

As name implies, OBDPR is primarily design pattern repository with ontological foundation. It is also more than this. It is a platform for building intelligent services to improve design pattern adoption. As such besides integrating and managing knowledge itself it includes several functionalities, besides holding repository of design patterns, containers etc.:

- Enables design patters experts to annotate patterns with additional knowledge.
- Integrate knowledge on particular design pattern from the web and additional data sources.
- Transform current RDF data to provide user-friendly view on design patterns.
- Index all the integrated data for supporting full text search capabilities of services build on the platform.
- Full access to RDF data to services built on platform including questions and answers, which will enable intelligent services to use expert system-like proposing or validating services.
- Set of real world examples and appropriate design patterns solutions in order to enable services to be used to train users or to show appropriate use of design patterns in real world examples.

Current OBDPR prototype (figure 2) implements all functionalities above. Because of that fact first experiments are in preparation. At the moment of writing this paper there are also three services running on top of the platform:

- simple full-text search service,
- proposing service to support design pattern selection and
- training service using real-world examples from underlying ontology.

OBDPR prototype includes all design patterns found in GoF and J2EE design pattern catalogues. It is also open for other sets of design patterns. GoF and J2EE catalogues give us enough opportunities to perform relevant experiments with design patterns on real-world examples and with real developers, since as mentioned before even repositories of that size showed to be problematic [5].

The implementation technology for OBDPR is Java EE with Jena [9] framework for accessing and performing core semantic operations on ontology. A simple user interface framework with basic functionalities like raw and user friendly view on repository is prepared. Framework is able to host additional services, developed in future.

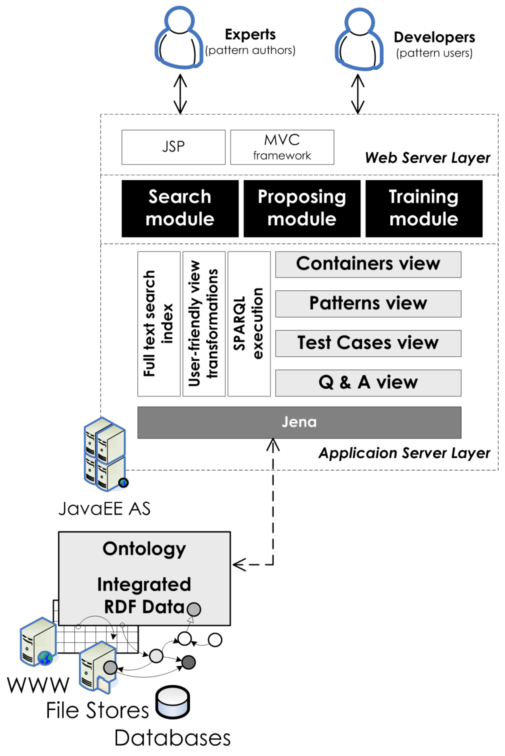
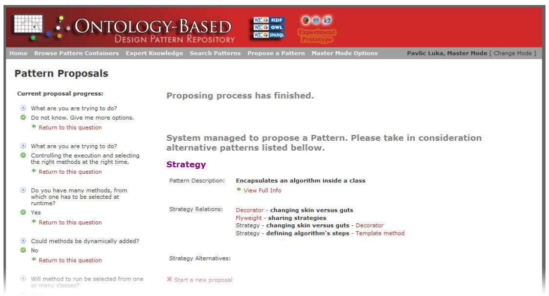


Figure 2. OBDPR architecture

At the moment there are three components on top of foundation platform (figure 2). They enable additional services, such as using full text search capabilities as well as training and using proposing services (figure 3). Not only data in OBDPR is indexed for full text search but also data from web, such as design pattern related content from Wikipedia and other design patterns related pages.



Pattern Proposals

Current proposal progress:

What are you are trying to do?
Do not know. Give me more options.
Return to this question

What are you are trying to do?
Controlling the execution and selecting the right methods at the right time.
Return to this question

Do you have many methods, from which one has to be selected at runtime?
Yes
Return to this question

Could methods be dynamically added?
No
Return to this question

Will method to run be selected from one or many classes?

Proposing process has finished.

System managed to propose a Pattern. Please take in consideration alternative patterns listed bellow.

Strategy

Pattern Description: Encapsulates an algorithm inside a class
View Full Info

Strategy Relations: Decorator - changing skin versus guts
Flyweight - sharing strategies
Strategy - changing skin versus guts - Decorator
Strategy - defining algorithm's steps - Template method

Strategy Alternatives:

Start a new proposal

Figure 3. Web based OBDPR user interface

5 Further work

One of the main goals of the project is to use it in order to perform several real-world experiments. After performing experiments which might show if and how much does OBDPR help adopting design patterns there are also other plans for future work. Some preliminary experiments have been already performed. Since the results were promising (figure 4) we are quite confident we are on the right track. The results showed us that less experienced users (test was made on 15 users) have significantly improved their design pattern adoption through solving 16 real world examples. The difference between the least and the most successful participant has also reduced. Since real experiment using OBDPR is a work to be done we are not going to discuss performed preliminary experiment in details here.

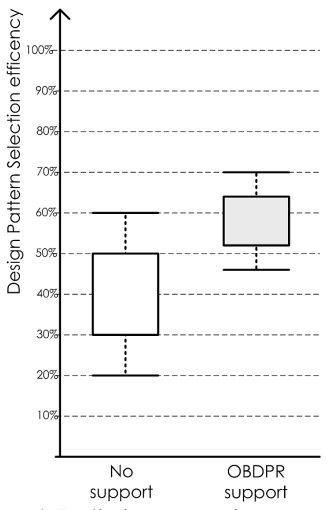


Figure 4. Preliminary experiment results

Before performing final tests it would also be reasonable to develop additional component that would enable user to verify if particular candidate was well selected. Service of that kind might help developers capable of selecting design patterns on their own to verify their selections. It could be even used to verify proposing or searching services correctness.

Another idea that is not jet realized is to expose OBDPR with simple interfaces, implemented as web services for instance. This would not only enable further integration but can also enable developing plug-ins for the most popular development tools such as Eclipse, NetBeans or Visual Studio. Having OBDPR always at hand during development sound like good idea to us.

There will also be several improvements in existing components. For example, we are trying to personalize the proposing component. The proposing component could learn about user from past proposals and ask personalized questions.

After performing research activities in means of experimenting with tool on industry developers we plan to develop a holistic methodology for design pattern selection. It will include both design pattern expert and user activities. OBDPR will be given a role of enabling tool for developed methodology.

6 Conclusions

The potential of using appropriate design patterns is not yet fully realized. On this level of reuse there are many challenges and issues remaining to be solved. Our past experiences tell us that finding a suitable design pattern for a given situation represents a great challenge for a typical developer. OBDPR was therefore developed to capture design pattern explicit and implicit expert knowledge, to enable further development of intelligent services and to test our belief that we can improve design pattern adoption.

Introducing concepts and technologies of the semantic web into the field of design patterns research creates new possibilities for making design patterns more approachable to software engineers. Preliminary experiments were performed, which showed us we are on the right track.

Semantic web technologies showed to be the right solution during the development. Especially when dealing with advanced requirements such as proposing design pattern for a given problem, and there are many more to come.

References

[1] A. H. Eden et al, Precise Specification and Automatic Application of Design Patterns, International Conference on Automated Software Engineering, IEEE Press, 1997.

[2] Core J2EE Patterns, http://java.sun.com/blueprints/ corej2eepatterns.

[3] D. K. Kim at al, A UML-based Metamodeling Language to Specify Design Patterns, Proceedings of the Workshop Software Model Eng. (WiSME) with Unified Modeling Language Conf. 2003, October 2003.

[4] D.Manolescu, W. Kozaczynski, A. Miller, J. Hogg, "The Growing Divide in the Patterns World", IEEE Software, Vol. 24, No. 4., July/August 2007, pp. 61-67.

[5] E. Gamma et al, Design patterns: Elements of reusable object orientated software, Addison Wesley Longman, 1998.

[6] Gerson Sunyé et al, Design Pattern Application in UML, ECOOP'00, http://www.ifs.uni-linz.ac.at/~ecoop/cd/papers/1850/18500044.pdf.

[7] J. M. Rosengard, M. F. Ursu, Ontological Representations of Software Patterns, KES'04, Lecture Notes in Computer Science, Springer-Verlag, 2004, http://w2.syronex.com/jmr/pubs/2004/ontology-pattern.pdf.

[8] J. M. Rosengard, M. F. Ursu, Ontological Representations of Software Patterns, KES'04, Lecture Notes in Computer Science, Springer-Verlag, 2004, http://w2.syronex.com/jmr/pubs/2004/ontology-pattern.pdf.

[9] Jena Semantic Web Framework, http://jena.sourceforge.net.

[10] L. Rising, "Understanding the Power of Abstraction in Patterns", IEEE Software, July/August 2007, Vol. 24, No. 4., pp. 46-51.

[11] L. Rising, The Pattern Almanac 2000: Addison Wesley, 2000.

[12] Marcus Fontoura and Carlos Lucena , Extending UML to Improve the Representation of Design Patterns, Computer Science Department, Pontifical Catholic University of Rio de Janeiro.

[13] Microsoft, Microsoft Patterns & Practices, http://msdn.microsoft.com/practices.

[14] OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features.

[15] R. Singh, Drive: An RDF Parser for .NET, http://www.driverdf.org/.

[16] RDF/XML Syntax Specification, http://www.w3.org/TR/rdf-syntax-grammar.

[17] Schmidt, D.C., "Using Design Patterns to Develop Reusable Object-Oriented Communication Software", Communications of the ACM, October 1995, Vol. 38, No. 10.

[18] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query.

[19] T. Berners-Lee, "Business Model for the Semantic Web", http://www.w3.org/ DesignIssues/Overview.html.

[20] W3C, "Semantic Web", http://www.w3.org/2001/sw/.