Searching Design Patterns in Source Code

Detlef Streitferdt, Christian Heller, Ilka Philippow
Ilmenau Technical University
{detlef.streitferdt | christian.heller | ilka.philippow}@tu-ilmenau.de

Abstract

Maintenance is a time consuming activity within software development and it requires a good understanding of the system in question. It is hard or even impossible to understand poorly documented legacy systems. Nevertheless, developers try to understand unknown object oriented systems by analysing the source code to recover the architecture of the system, which is a hard task since the dependencies between the classes cannot be recovered good enough. Here, the knowledge about design patterns can help developers to understand the underlying architecture faster. We analysed existing pattern search approaches and compared them by their recall and precision values, metrics out of the Information Retrieval domain. As a result we developed own pattern search algorithms for the 23 design pattern described by Gamma et al. [1]. This fast abstract briefly explains the basics of our pattern search and describes first results of the search algorithms developed as a Java plug-in for the Together IDE. This work was funded by the BMBF [2] and is part of the InPULSE [3] project.

Keywords: Pattern Search, Search Algorithms, Design Patterns.

1 Introduction

Developers need to understand software systems before they can maintain them, although documentation and/or design models are missing or of a poor quality. In most cases only the source code as the basic form of documentation is available.

Design patterns are pre-designed and tested solutions of fundamental design problems leading to an advantage for developers, since they can use and learn from the wisdom of experienced colleagues. Knowledge about patterns in existing systems together with the relations of classes to corresponding patterns leads towards a better and faster understanding of software systems. Patterns used in an architecture are not explicitly described in most systems and thus, we developed own algorithms for the automated search of patterns in source code.

1.1 Background

We analysed current pattern search methods like [4], [5] or [6], to rate their search quality, quantified by metrics based on the found and existing patterns. Three cases need to be distinguished:

- *True positive*, in case a patterns was identified and is really existing in the source code. This case is desired.
- *False positive*, in case a pattern was found, but is not implemented in the source code. This case has to be avoided.
- *False negative*, in case a pattern is implemented and existing in the source code, but was not found. This case has to be avoided.

We have done the quantification using the recall and precision values out of the Information Retrieval domain [7]. *Recall* is the number of the implemented patterns divided by the number of found patterns. *Precision* is ratio of found patterns divided by the number of existing patterns. Unfortunately none of the existing approaches could be sufficiently used for searching all Gamma patterns. Thus, we developed new and extended existing search algorithms.

1.2 Searching for minimal key structures

The key point of our algorithms are negative search criteria for patterns. With this we define the structure of each pattern by its required, forbidden (negative) and don't-care elements. All required elements have to be present, forbidden elements are not allowed in the structure and uncertain elements can be present, but will be ignored to come to a successful identification of a pattern by the algorithm. We defined search structures for all Gamma patterns together with algorithms which we implemented as Java plug-in for the Together IDE.

Analysed System		Drawlet		AWT		Tomcat		Patterns			
	Pattern	included	found	included	found	included	found	included	found	prec. in %	recall in %
Creational Patterns	Abstract Factory	n	3	y	10	n	4	1	1	100	100
	Builder	n	87	n	127	n	469	1	12	8	100
	Factory Method	y	13	n	27	n	22	4	4	100	100
	Prototype	y	n	n	11	n	2	2	2	100	100
	Singleton	n	n	y	8	n	6	1	1	100	100
Structural Patterns	Adapter	n	40	n	52	y	123	1	3	33	100
	Bridge	n	61	y	61	n	345	1	25	4	100
	Decorator	n	n	n	n	n	n	1	1	100	100
	Facade	n	194	n	322	y	973	1	77	1	100
	Flyweight	n	n	y	n	n	n	1	1	100	100
	Composite	y	n	y	n	n	n	1	1	100	100
	Proxy	n	9	n	14	y	73	1	3	33	100
Behavioural Patterns	Command	n	13	n	20	n	46	1	1	100	100
	Observer	y	n	y	n	y	n	1	1	100	100
	Visitor	n	1	n	6	n	4	1	1	100	100
	Interpreter	n	n	n	n	n	n	2	2	100	100
	Iterator	y	n	n	n	n	n	1	1	100	100
	Memento	y	n	n	n	n	3	1	1	100	100
	Templ. Method	y	1	y	22	n	17	1	1	100	100
	Strategy	y	4	n	4	y	12	1	17	6	100
	Mediator	y	135	y	88	n	220	1	25	4	100
	State	n	n	n	n	n	2	1	1	100	100
	Chain of Respons.	n	n	n	n	y	26	1	3	33	100

The numbers are showing the included or found patterns.
y The pattern was used to develop the system, but we don't know how often.
n It is not known whether or not the pattern is present in the system.

Table 1: Results of Searching Patterns

2 Results so far

Up to now we tested our algorithms with several systems. *Drawlet,* a picture processing system with 195 classes [8], the *Abstract Windowing Toolkit* (AWT) as part of the *Java 2 Standard Edition* with 354 classes and *Tomcat* as part of the *Jakarta Open Source* project with 1035 classes. We have taken the documentation for AWT and Tomcat from [9] and for the *Drawlet* project from [10]. Unfortunately these documents are rather poor when it came to patterns. Thus, we were not able calculate the precision and recall values for these projects.

The lack of a good reference system turned out to be the key problem for the comparison of pattern search algorithms. We found no reference system with well documented patterns, that is freely available for testing pattern search algorithms. Thus, a very important issue in our research group is the development of such a reference system for pattern search algorithms. First, we implemented all patterns according to Gamma, what lead to the *Patterns* project with 88 classes. For better results we already have a version of a more complex pattern reference system, which is still under development. We plan to have the first public version ready this fall.

Given the poor documentation, only in our *Patterns* system we have been able to quantify the search quality, see Table 1. The low quality values are due to structural similarities to non-pattern structures, what we address in one of our current projects.

3 Outlook

We are permanently enhancing our search algorithms, which will soon be adjustable to address many implementation variants for a given pattern. The precision and recall results can be influenced by parameters that will be adjusted by the developers. In addition, we are working on an enhanced reference system for all 23 Gamma patterns, fully documented and implemented in different flavours. With this reference system we will be able to compare different search algorithms better. Finally a pattern search plug-in for Eclipse is an issue for the near future.

4 References

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns - Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[2] BMBF, Bundesministerium für Bildung und Forschung, www.bmbf.de, 2004.

[3] InPULSE, Integrative Pattern- und UML orientierte Lern- und System-Entwicklungsumgebung, www.inpulse-online.de, 2004.

[4] G. Antoniol, R. Fiutem, L. Cristoforetti, "Design pattern recovery in object-oriented software", In Proceeding of the 6th International Workshop on Program Comprehension (Ischia, Italy, June 1998), pp. 153-160 , 1998.

[5] Jagdish Bansiya, Automatic Design-Pattern Identification, www.ddj.com, 1998.

[6] Hyoseob Kim, Cornelia Boldyreff, "A method to recover design patterns using software product metrics", In Proceedings of the Sixth International Conference on Software Reuse (ICSR6), Vienna, Austria , 2000.

[7] Salton G., *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.

[8] Rolemodel Software, Homepage, www.rolemodelsoftware.com/drawlets/index.php, 2004.

[9] Wiki-Server, PatternStories, wiki.cs.uiuc.edu/PatternStories/DesignPatterns, 2004.

[10] Ken Auer, "Fundamental Elements of an Extendible Java Framework", Rolemodel Software, www.rolemodelsoftware.com , 1997.