UNDERSTANDING PRODUCT LINES THROUGH DESIGN PATTERNS

Daniel Cabrero
Dirección General de Tráfico, Spanish Ministry of Internal Affairs, Madrid, Spain

Javier Garzás
Kybele Consulting S.L. Madrid, Spain

Mario Piattini
Alarcos Research Group. University of Castilla-La Mancha. Ciudad Real, Spain

Keywords: Design Patterns, Software Product Lines, Systematic Literature Review, Variability Points.

Abstract: Many proposals concerning design and implementation of Software Product Lines have been studied in the last few years. This work points out how and why different Design Patterns are used in the context of Product Lines. This will be achieved by reviewing how often those patterns appear in different proposed solutions and research papers for Product Lines for a given set of sources. This information will help us identify which specific problems need to be solved in the context of Product Lines. In addition, we will discuss how this information can be useful to identify gaps in new research.

1 INTRODUCTION

Software Product Lines engineering gathers the analysis, design and implementation of a family of systems in order to improve the reuse of the commonality among them. Thus, a Product Line is a group of "similar" systems (Clements and Northrop, 2001). Each system can be defined as the variability within the rest of the family.

The main challenge of this tendency in engineering is to establish the appropriate mechanisms for modelling and implementing this variability (Myllymäki, 2002) and to save cost and time by reusing components whenever possible.

Design Patterns describe common problems and their solutions (Gamma et al., 1995) in such a way that analysts and software designers can easily retrieve them. Thanks to the fact that experienced software engineers and domain specialists develop patterns, the software community can take advantage of this reliable knowledge, available from pattern libraries.

The experience at hand in the field of reusability of common features and components is well known and a lot of work on this has been successfully applied in a wide variety of systems, adopting solutions defined in patterns. Fortunately, all this experience has been gathered, and is obtainable through pattern libraries.

In terms of modelling variability in design, Software Product Lines are not much different from other systems. Therefore, one of the major differences between a classic development and a Product Line-oriented development is how the requirement analysis is done. In Product Lines requirements are collected in terms of "Variability Points" (Keepence and Mannion., 1999), which are differences among systems within a Product line.

The remainder of the paper is organized as follows. Section 2 describes how we carried out a review on how different frameworks use this existing knowledge. An ordered list of patterns and refactorings will be summarized, based on the frequency of their appearance. Section 3 focuses in more in detail on the most common problems faced by Software Product Lines. Finally, section 4 draws some conclusions and identifies future research work.

Cabrero D., Garzás J. and Piattini M. (2007).
UNDERSTANDING PRODUCT LINES THROUGH DESIGN PATTERNS.
In *Proceedings of the Second International Conference on Software and Data Technologies - SE*, pages 405-408
DOI: 10.5220/0001323304050408

2 DATA RETRIEVAL: A SYSTEMATIC REVIEW IN PRODUCT LINES

A Systematic Literature Review is a means of identifying, evaluating and interpreting all available research that is relevant to a particular research question. Individual studies contributing to a Systematic Review are gathered and new conclusions are obtained from its summarization and analysis (Biolchini et al., 2005). This methodological literature review is common in other science disciplines such as medicine, but was recently introduced in Software Engineering by (Kitchenham, 2004).

2.1 Research Question

The difference between a Systematic Literature Review and a traditional Literature Review is that "the research conduction process of a Systematic Review follows a well defined and strict sequence of methodological steps" (Biolchini et al., 2005). These "strict" steps include the definition of the followed procedure in the research, which focuses different aspects such as the research question, sources, query strings or selection criteria.

In the context of this research, we performed a Systematic Review focusing on Design Knowledge as defined in (Garzás and Piattini, 2005) (Design Patterns, Refactorings, Design Principles, Rules, Bad Smells and Heuristics) applied to Software Product Lines. The research question was defined as shown in the Figure 1.

Which kind of Design Knowledge (e.g. Patterns, Refactorings, Principles, Rules, Bad Smells and Heuristics) is commonly used in Software Product Lines?

Figure 1: Research Question

2.2 Execution of the Systematic Review

In this case, a specific set of Query Strings was used to identify research articles in three sources: the IEEE, the ACM and the SCIENCE DIRECT portals. The Table 1 shows the strings used, as well as the number of selected studies from them. For example, the cell corresponding to the first column and the first row establishes that 30 documents were retrieved for the "Design Pattern" + "Product Line" string queries.

Table 1: First Search. Retrieved Studies.

	Product Line	Product Family
Design Pattern	30	11
Heuristic	19	3
Design Principle	10	9
Bad Smell	0	0
Refactoring	25	6
Design Rule	6	2

Once the search was performed, 121 relevant studies were selected. After that, we filtered the really important papers using the selection criteria. The selection criteria was to read the article abstract in order to ensure that the they talked about Product Lines and Design Knowledge Concepts defined in (Garzás and Piattini, 2005). After filtering each of them, we found that they mostly focused on architectural issues, requirements management, and many other aspects, but very few of them referred to lower level design aspects.

The Table 2 shows the number of results for each string query after the selection criteria was applied.

Table 2: First Search. Filtered Studies.

	Product Line	Product Family
Design Pattern	11	2
Heuristic	2	0
Design Principle	2	1
Bad Smell	0	0
Refactoring	4	0
Design Rule	0	0

Eventually, we discovered that some of the retrieved documents did propose new patterns for managing variability. Those "complex" patterns could be broken down into "classic" Patterns and Refactorings, such as those defined by the Gang of Four (Garzás and Piattini, 2005), (Buschmann et al., 1996) or (Fowler, 1999). Among those "complex" patterns we can highlight the Single Adapter Pattern, Multiple Adapter Pattern and Option Pattern (Goedicke et al., 2004, Keepence and Mannion., 1999)., the SCV Analysis (Coplien et al., 1998) or the Command Language Pattern (Goedicke et al., 2004).

In the end, after reading carefully each selected document, we had found 4 articles published in

Journals related with different patterns, as depicted in the Table 3.

Table 3: First Search. Final Results.

	(Keepence and Mannion., 1999)	(Coplien et al., 1998)	(Goedicke et al., 2004)	(Ziadi et al., 2003)
Abstract Factory	X	X	X	X
Singleton	X			X
Null Object	X			X
Replace If with Inheritance	X			
Adapter		X		
Message Redirector			X	
Service Abstraction Layer			X	
Command Processor			X	
Command			X	
Interpreter			X	

The summarized data given in the Table 3 establishes that only Patterns and Refactorings were found in the retrieved papers.

We noticed that articles focusing 'low level' design aspects used class diagrams. Because of that, we performed a second search, this time in Internet, using 'class diagram' and 'pattern-based' Strings, as shown in the Table 4.

Table 4: Second Search. Retrieved Studies.

	Software Product Line	Software Product Family
Class Diagram + Pattern-Based	90	20

After reading the abstract of the 110 related studies returned by the search queries (90 from the first query string and 20 from the second one), we found that 4 research works used any of the above-mentioned Design Knowledge. The Table 5 depicts the different patterns mentioned in each study.

All of them used patterns, but no reference was found in this second search related to refactorings, bad smells, design principles, design rules or heuristics.

Table 5: Second Search. Final Results.

	(Myllymäki, 2002)	(Bachmann and Bass, 2001)	(Harsu, 2001)	(Muthig et al., 2004)
Abstract Factory	X	X	X	
Strategy	X			
Mediator	X			X
Proxy	X			
Singleton			X	

2.3 Data Synthesis

The next step in our work was to check how often those patterns appear in the Product Line-based solutions and to build an ordered list based on their occurrence in the literature.

The list shows the most-used patterns in Software Product Lines and their occurrence per document in parentheses:

1. Abstract Factory (7)
2. Singleton (3)
3. Mediator (2)
4. Null Object (2)
5. Proxy, Command, Adapter, Interpreter, Message Redirector, Strategy, Service Abstraction Layer, Command Processor, Replace If with Inheritance. (1)

It is interesting to highlight that, by observing the list of patterns used in Product Lines, we can take advantage of the common pattern language provided by pattern libraries. Thus, we can associate those patterns with the problems that they try to solve. In other words, the pattern frequency defines many important aspects of the system.

A quick glance at the list shows a clear preference for the Abstract Factory Design Pattern. A long way off this as regards frequency, we can find the rest of Patterns and Refactorings.

The next sections will explain the basics of the patterns found, and how they are used within Software Product Lines.

3 CONCLUSIONS

Very often, a system or technology can be defined by means of the problems that it tries to solve. Identifying those problems and having an overview of the state of the art in this respect is a necessary step in the process of producing a new proposal.

This research work reaches several objectives. First of all, it highlights what the main problems in SPL are, currently, as well as how they are being solved using patterns. This has been achieved empirically, studying the appearance frequency of patterns, instead of basing conclusions on personal opinions.

Secondly, this article shows a new line of research that aims to cover gaps in research on the use of refactorings, bad smells, design principles, design heuristics and design rules in Product Lines.

In addition, we propose that future work can be focused on the lack of a detailed library that analyses and evaluates each relevant pattern-based solution and then gives guidelines as to which proposal should be used in different cases.

ACKNOWLEDGEMENTS

This research is partially supported by the ESFINGE project of the General Research Council (Dirección General de Investigación) of the Spanish Ministry of Education and Science (TIC 2003-02737-C02-02) and ENIGMAS (Entorno Inteligente para la Gestión del Mantenimiento Avanzado del Software), supported by the Department of Education and Science of the Junta de Comunidades de Castilla-La Mancha (Regional Government of Castilla-La Mancha) (PBI-05-058).

REFERENCES

Bachmann, F. & Bass, L. (2001) *Managing variability in software architectures*, Symposium on Software Reusability ACM Press

Biolchini, J., Mian, P. G., Natali, A. C. C. & Travassos, G. H. (2005) *Systematic Review in Software Engineering*. Rio de Janeiro, COPPE / UFRJ.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. & Stal, M. (1996) *Pattern-Oriented Software Architecture – A System of Patterns*, John Wiley and Sons Ltd.

Clements, P. & Northrop, L. (2001) *Software Product Lines: Practices and Patterns*, Addison-Wesley.

Coplien, J., Hoffman, D. & Weiss, D. (1998) *Commonality and Variability in Software Engineering*. IEEE Software 15**,** 37 - 45.

Fowler, M. (1999) *Refactoring*, Addison Wesley.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995) *Design Patterns*, Addison-Wesley.

Garzás, J. & Piattini, M. (2005) *An ontology for micro-architectural design knowledge*. IEEE Software Magazine, 22**,** 28-33.

Goedicke, M., Köllmann, C. & Zdun, U. (2004) *Designing runtime variation points in product line architectures: three cases*. Science of Computer Programming, 53**,** 353 - 380

Harsu, M. (2001) *A Survey of Product-Line Architectures.* Tampere, Tampere University of Technology.

Keepence, B. & Mannion., M. (1999) *Using patterns to model variability in product families.* IEEE Software, 16**,** 102-108.

Kitchenham, B. (2004) *Procedures for Performing Systematic Reviews. Keele University Technical Report.* Keele, Software Engineering Group.Department of Computer Science. Keele University.

Muthig, D., John, I., Anastasopoulos, M., Forster, T., Dörr, J. & Schmid, K. (2004) *GoPhone - A Software Product Line in the Mobile Phone Domain. IESE-Report.* Fraunhofer IESE.

Myllymäki, T. (2002) *Variability Management in Software Product Lines*. Tampere, Institute of Software Systems. Tampere University of Technology.

Parnas, D., Clements, P. C. & Weiss, D. (1984) *The Modular Structure Of Complex Systems*, International Conference on Software Engineering Orlando, Florida, IEEE Press.

Ziadi, T., Jézéquel, J.-M. & Fondement., F. (2003) *Product Line Derivation with UML*, Groningen Workshop on Software Variability Management,