The Adoption of Software Patterns in the Dutch Software Industry

J. KABBEDIJK, G. NIJBOER and S. JANSEN, Utrecht University

There is no clear insight into the adoption of software patterns within the Dutch software industry. This leads to the fact software pattern researchers experience difficulties communicating pattern research with industry and struggle to gather pattern usages. This study explores the use of patterns within the Dutch software industry. Six different software companies were interviewed on how they used software patterns in their daily practise. The research questions are aimed at identifying what type of patterns are used and whether patterns are used by developers in their own or more as a company wide policy. The research concludes that software patterns are often implicitly or even unconsciously used by software companies, instead of explicitly in communication. Patterns identified within academic research are indeed known and used within the software industry, but still more on a personal level. The companies see a lot of value in the use of patterns, but the actual use currently lacks behind because of missing support from within the company.

Categories and Subject Descriptors: D.2.11 [**Software Engineering**]: Software Architectures—*Patterns*

General Terms: Software Patterns

Additional Key Words and Phrases: Case Study, Interview, Software Pattern, Pattern Adoption

ACM Reference Format:

Kabbedijk, J., Nijboer, G. and Jansen S. 2013.The Adoption of Software Patterns in the Dutch Software Industry. jn 2, 3, Article 1 (May 2010), 8 pages.

1. INTRODUCTION

In software development, developers often face the same problems over and over again, gaining experience in designing and implementing solutions, resulting in faster delivery of the same solution over time. But why would you re-invent the wheel if others have already done so, and documented it for reuse? Patterns are developed as solutions for common problems in software development [1]. One of the characteristics of all software patterns (descriptions of sets of successful solutions for recurring problems within a certain context [2]) is that they have several levels of abstraction, spreading from detailed code-implementations to abstract architectural concepts. The concept of patterns is that they are generic, and can therefore be used in many processes of software design, and even in designing physical objects, such as buildings.

Since new software development methodologies have been introduced in the past 20 years, one might expect the concept of related concepts, including software patterns, to change along as well. The concept of patterns could even disappear, due to the lack of support for such topics in modern, agile software development methodologies. The generic characteristic of patterns, however, leads to the fact that they are currently still used and researched, also causing the rise of new patterns, for instance in research by Kabbedijk [3].

This research is part of the 'Product as a Service' project (www.productasaservice.org).
Corresponding author: J. Kabbedijk (j.kabbedijk@uu.nl)

Through this research, we have created an indication of how software patterns are currently used within software developing companies in the Netherlands. This research, performed at six software developing organisations, is conducted through interviews with key actors in the organisations, playing a central role in the development process. The result of this research is aimed at providing insight in the current use of software patterns in the software industry. This creates a better understanding of the way patterns should be communicated with the software industry. This paper will first report on the research approach, containing our interview guidelines and structure in section 2. In section 3 the characteristics of the case companies are presented, followed by the results of the conducted interviews in section 4. These results are analysed in section 5. A conclusion summarises the analysis in section 6 and answers each of the sub-questions introduced in the next section of this paper.

2. RESEARCH APPROACH

We have approached this research by stating one primary research question, and then dividing this main topic into four sub-questions, specifying the scope of the research. The research question and its four sub-questions are formulated as follows:

How are software patterns currently used within software developing companies in the Netherlands?

—Are software patterns intentionally used in software development?

—How are software patterns consulted by developers and architects?

—Are software patterns mostly used at architectural, design or idiom level?

—Are software patterns primarily applied in implicit problem-solving, or are they also explicitly referred to in software development documentation and communication in-house?

Six of the invited companies were willing to participate in our research. We attempted to cover the full spectrum of software companies in the Netherlands as good as possible by selecting case companies from different domains. This prevents that our results are based on only companies from, for example, the ERP-domain. Though we have only spoken with one interviewee per company, a variety of responsibilities amongst the interviewees was gathered. Each of the participating organisations has been given an alias, in order for them to remain anonymous. The names of the interviewees can be found in the Acknowledgements in section 7.

The interviews conducted at the participating companies have focused at gathering qualitative results, rather than quantitative results. Each case has been conducted on a one-to-one basis, in a quiet space, to prevent interference from others during the interview. These interviews have primarily taken place at the location of the companies itself, though there are exceptions for the interviews with GenerateComp1 and AllroundComp. The case of GenerateComp1 was performed at the Utrecht University. The case of AllroundComp was performed through an online video call with Skype, because the interviewee was located in Belgium.

The interview protocol set up and respected as a guideline for conducting the interviews was semi-structured, flexible and dynamic. Due to the openness of the questions in the document, we have implemented some freedom in the thread of the interviews, leading to usable, qualitative data. The interview protocol did not focus only on the adoption of software patterns, as we also left an adequate amount of space for information about other aspects. Included was information on the participating company, covering its size, locations, focused industries and products. Data about the interviewees themselves was also gathered, to state more about their background and education. We also gathered data about the approach to starting new software projects, since we believe the way the software is developed plays a strong role in the adoption of software patterns by the same team.

The section of the interview focusing on software patterns covered both basic question - such as: *"Could you give us a definition of what you believe software patterns are?"* - and more in-depth questions about the way software patterns are used and referred to.

Audio was recorded during every interview, so that it could be used as a reference in later stages of the research. Apart from the recordings, notes on paper were also made during the interview, to capture the main topics and to produce an outline for the written reports. These reports consisted of approximately two pages of text about the topics discussed in the interview, which could be used as results for the analysis. The outcomes of the interviews were summarised carefully and with detail within 24 hours.

3. CASE COMPANIES

As mentioned in the previous chapter, we have gathered the data from the interview cases into separate reports, which will be discussed and analysed in this research. We will go into more detail about each of the companies, accompanied with data from table I, which lists the participating organisations and interviewees and their characteristics.

Table I. : Case Company Characteristics

Company	Location	FTEs	Product	Type	Industry	Interviewee
ResearchComp	Enschede	60-70	Process modeller	On premises	Generic	Researcher
CRMComp	Utrecht	25	CRM-suite	On premises / hosted	IT / Telecom	Developer
GenerateComp1	Leiden	5	Software generator	SaaS to on premises	Generic	Lead Developer
GenerateComp2	Rotterdam	95	Software generator	On premises to SaaS	Generic	CTO
AllroundComp	Gouda	5,300	SCM, ERP, CRM	On premises	Generic	SCRUM master
SCMComp	Rotterdam	20	SCM-suite	SaaS	Generic	Lead Developer

The first company, **ResearchComp**, is a research-company in Enschede with 60 to 70 employees, focusing on research and innovation in ICT. One of their current topics of research is enterprise architecture. The company produced one software product in the past, which became an independent and successful spin-off. The product is a process-modelling software product, which is installed on-premises. The development of this software product has been our main focus during the interview, because our interviewee has participated as a principal researcher in the development process. He has worked in the development of the software product we spoke about in the previous paragraph. He currently researches topics on enterprise architecture for the organisation and its clients.

The second company, **CRMComp**, is located in Utrecht and builds a CRM-suite which focuses on IT and telecom companies. They currently have 25 people employed, including 5 developers. The reason they develop the product especially for these industries is that such organisations are distinguished by project-based planning and less logistic planning. The company currently exists for 15 years, and they have transformed their product into a .NET-product five years ago. The product is currently deployed on-premises, but a SaaS-solution is scheduled for development. A developer responsible for the architecture of the software framework was interviewed for our research. Their team is formed by a group of 5 developers. There is no specific software architect, all software developers are responsible for the software architecture.

GenerateComp1 is located in Leiden and builds a software generator, based on model-based development. It is a small business of 5 FTEs which is still evolving, but the lead developer has given us a good idea of the use of software patterns within their development process. The software generating product can be applied to generally any industry, though the applications often focus at CRM and ERP. The generator is a SaaS-solution, and the generator products are installed on-premises or hosted.

At **GenerateComp2**, also a software generator is developed, but then at a much larger and much more completed stage than at GenerateComp1. Their Chief Technology Officer has participated in our interview, giving not only details about the use of software patterns, but also about how to organise a modern, dynamic business that is purely active in developing software. Their development follows the SCRUM-methodology, the teams participate actively in this concept. Since the organisation is also located in the USA and in South-Africa, other sales-channels are opened for the software to be sold. Added to this is a large network of partners, who use the software generator to deliver software products to their clients. The responsibilities of the interviewee included product development and defining the corporate strategy.

The **AllroundComp** organisation does not focus on one information system, neither does it only develop software. But due to its size, it gave us insight in how software is developed in a multinational company that develops software in teams that could cross national borders. Our interviewee was one of the SCRUM-masters, being responsible for the planning and realisation of the planning of his team, by preventing any external factors of interfering with the daily workflow. He therefore delivers coordination and commitment to his team.

The last company listed in the table is **SCMComp**. The company, settled in Rotterdam, builds a hosted Supply Chain Management suite. It is also located in New York and India, making 24/7 development and support possible. Since the suite can be customised to fit any type of industry, the hosted deployments are often reused by customers with the same characteristics and industrial focus. The interviewee participating in our research has taken a more management-central task, and is also responsible for management in engineering. The SCMComp maintains a strict policy against software-faults, where the fault has to be acknowledged within 15 minutes, and solved within 4 hours, delivering great added value to its customers. Because the software focuses mainly on large organisations with complex logistics, it is also a required and requested service, as one may lose large amounts of money per minute if the entire logistics software suite has crashed.

4. INTERVIEW RESULTS

Table I gives an overview of the characteristics of the participating interviewees, the organisations they are employed for, and the software product that is developed. According to data from the Dutch Chamber of Commerce, the total amount of organisations in the industry of software development in the Netherlands was stated at 47,712 for January 2012 [4]. 495 organisations have entered the industry and 974 have stopped their business, leaving a negative net score of -479 for that month. We believe that the results of the six organisations that have participated in our research apply to the larger part of all software developing companies, because there is diversity in size, domain, product and industry amongst the participating companies.

The size of the organisations differs a lot amongst the participants. We see a small-sized company at GenerateComp1, employing only five FTEs. But nevertheless, the software product that is built is a software generator, which is not a common software product. On the other hand, we have visited AllroundComp, which employs 5,300 people and works on multiple software development processes at the same time, in multiple countries. This diversity gives us a good indication of how software patterns are used within both small and large companies.

Though all interviews were conducted from a location in the Netherlands, we have visited companies that are pure national players, but also some that are multinational software vendors. The GenerateComp2 company uses a broad network of partners to create new sales-channels, and the SCMComp has locations in New York and India, making 24/7 service delivery possible. We have requested each interviewee to give us a definition of the term software patterns. Though using different words, as expected, each interviewee managed to give a definition that meets the required elements that we set before starting the interviews. We have focused on the general definition of *"software patterns are descriptions of sets of successful solutions for recurring problems within a certain context"* [2].

Each of the interviewees, excluding the participant of GenerateComp1, has had an education that included one or multiple courses on software patterns. This implicit knowledge was once taught, but is either consciously or unconsciously applied during implementation of code later, too.

In the case of **ResearchComp**, we were not able to gather results on the approach to new software development, since we could only identify and discuss one event where this actually took place. This did not give us the opportunity to speak of a standardised approach to setting up a new software development project. During the development of their previous software product, design patterns were actively used. Not only were they implicitly used in the minds of the software developers, they were also explicitly used by means of communication and books. They were not only used for problem-solving, but also as a guideline in designing and developing software. The use of software patterns has been at a higher level of abstraction, not at the lower, more detailed level with for instance the Singleton-pattern.

By being told that ResearchComp did not only implicitly use design patterns, the interviewee referred to a book case full of literature on topics such as patterns, architecture and methodologies. Not only printed literature was used, digital references on the internet were also used to gain information from. At ResearchComp, research has been done on multichannel-architectures, and patterns are being created to deliver common solutions on this topic, gathered in a catalogue by a group of thirty patterns.

CRMComp lays focus on implementing features from previous products when starting the development of a new software product. This requires little customer participation, because those existing features are already stabilised and accepted through customer feedback. The functional requirements of the new software product are formed by features from previous systems and new requirements requested by management, sales, and customers. The interviewee confirmed that software patterns are used at CRMComp. An example was raised when several interfaces used the same switch statement over and over again. They tackled the issue by implementing the Abstract Factory pattern, making the switch statement reusable throughout multiple classes. The interviewee indicated that factories are commonly used as a solution to tackle development issues. Our interviewee was educated to use patterns during software development, which created a basic knowledge for him to rely on during the process of development. This leads to the fact that our contact owns books like that of the *"Gang of Four"* [1].

Though the contact used the book during education, he no longer uses the book at his job. He does use resources on the internet when a quick recap on any of the concepts is required. We were told there is no such thing as communication in terms of patterns between colleagues, so patterns are only implicitly used by developers. Interesting to note is that the interviewee mentioned that "design patterns should not be used because one believes that they must be applied, but they should be used to solve problems during development". By this, he means that you should not confuse the means and the ends, since design patterns are those means to get to the ends. There is no management push to use software patterns, the decision to use them has been taken and executed by the developers themselves.

During our interview with **GenerateComp1**, it became clear that there is a strong presence of agile software delivery when using the generator to create a software product. Strong involvement of the customer or end-user is required and this makes it possible for them to deliver software while having completely no knowledge of the domain it is designed for.

Software patterns are "sometimes" used by the interviewee, and they are often applied at a lower level of abstraction, which means the more detailed, creational patterns. The use of patterns is actually implicit, so developers are not explicitly and consciously studying and implementing patterns during development. Added to this, the interviewee admits that he, as a graduated information scientist, did not have the proper education to be an expert in software patterns. Though, his understanding of software patterns was sufficient to contribute to the research results. There is no management-push to implement software patterns, and neither is there any training - internally or externally provided - in software patterns. Whenever necessary, it does happen that more information on patterns is researched through sources on the internet.

At **GenerateComp2**, new software products are explicitly guided by a product roadmap. Input for a new version of the software generator comes from over seven sources, and therefore strong considerations between product marketing and development have to be made. The flexibility in the development at GenerateComp2 comes from their SCRUM and Agile software development methodology.

Our interviewee was very aware of software patterns and the way they can be implemented. He was able to name some patterns and authors on books that describe software patterns. He indicates that design patterns are used as a source of inspiration, and are therefore implicitly used, but there is no top-down push to force the developers to use design patterns during implementation. The abstraction at which software patterns are used within GenerateComp2 is at a lower level, enabling "pattern based engineering". The model-based engineering done through the software generator of GenerateComp2 is therefore purely the use of patterns, as is implementing a template of such a generated software product.

At **AllroundComp**, many software products are developed at the same time, and since each might have a completely different approach to software development, we decided to focus on the business unit that our interviewee is currently active in. There's a great respect to their methodology, which is SCRUM.

The interviewee mentioned that "it is easy to write code, but it is hard to write good code. Code has to be reusable, flexible and not too rigid, so loosely coupled". Those solid design principals assist in applying solutions to often recurring problems. He says: "and when you give such a solution a name, it becomes a pattern". He states that it is a skill to be able to identify and implement patterns, but you should never apply software patterns just because you think it makes your code "cooler". A developer should only implement patterns when it can aid in solving issues during design or development. The term "pattern-fever" is related to the overactive use of patterns and was mentioned to describe the previously statement.

The interviewee believes that one can become an expert in software patterns by studying the solid patterns, and by implicitly implementing these patterns in self-written code. After reviewing the code and learning from what you did, you can get highly experienced in patterns. In the organisation, there are no specific external trainings provided for (new) employees which focus on patterns, but it might be a subject during a more generic training. The most valuable thing in learning patterns and learning to write good code is transferring implicit knowledge amongst colleagues. There is no specific management-push to use software patterns, so whenever this happens, it is often the choice of the developers themselves.

Sometimes patterns may be explicitly communicated, though this does not take the form of code, but more in principles of implementation. Books on the topic are present in the office of AllroundComp, and even better, a complete book-exchange has taken place there. Patterns are often implemented at a lower level of abstraction, because the interviewee believes it is also easier for developers to think about them at a lower, concrete level, rather than at a higher, more abstract level. He also mentions to think of higher level patterns more as being software architecture.

We conducted the final interview at **SCMComp**. The plan of approach for new software products from SCMComp is based on the PRINCE2 five-phases model, existing of divine, design, develop, deliver and deploy. It is mainly the cost-technical driver in their software development. Their software is developed according to the SCRUM and Agile methodology. There is no internal education on software patterns, but all employees have studied at a university, and software patterns were part of their education. New personnel is not sent to an outsourced training on software development and the use of patterns, but at SCMComp they would rather choose to hand the new employee some books on design patterns and make use of internal knowledge exchange. There is no sincere push from management for implementing software patterns, though new colleagues may be informed about changing their code to meet the patterns' aspects during code reviews. This is rarely needed for the older colleagues, since their code is often developed conform the company's patterns.

Software patterns can be found both at the higher and lower level of abstraction at SCMComp, both during the design and implementation phases of the software product. One may find patterns in terms of communication, for instance during code reviews, but they are mostly used implicitly. Currently, the use of patterns is becoming less, because they are already implemented in the product and so they require little attention for further development. Patterns are sometimes researched through books or the internet, but the "less exotic" software patterns are in the developers' minds.

5. ANALYSIS

In order to analyse the results of our research at six software developing organisations, we will cover each of the four sub-questions in a separate section.

5.1 Are software patterns consciously consulted when requested for problem-solving in software development?

We have seen several cases in which it was explicitly mentioned that software patterns were referred to in software development, in order to solve problems or issues in code implementation. We have seen such cases at

ResearchComp, CRMComp, AllroundComp and SCMComp, where they were truly, actively and explicitly used to tackle common software development problems.

Added to this, we saw very little management push. There was mostly bottom-up influence on the use of software patterns in problem-solving. It seems that in our six cases, software developers have decided for themselves that implementing software patterns into their code will help them in creating efficient, reusable, maintainable and sustainable code.

Five of the interviewees participating in our research have also been taught about software patterns during their education. This leads to the fact that the knowledge and skills on using and implementing software patterns have become implicit knowledge. Therefore, they are often implicitly implemented throughout code, not necessarily consciously, too.

5.2 How are software patterns consulted by developers and architects who are using them to solve problems?

When explicitly and consciously using software patterns to solve common issues and problems in software development, we have mostly seen books, literature and the internet as a source to conduct research. Researched topics did not only include recaps on how a pattern should be implemented, but also which software pattern is best to solve a specific problem. This second issue is often much more complex and could take up much more time than just doing a quick recap.

We have even heard about an internal library of related literature at AllroundComp, where the contents of the collection can be used to recap implicit knowledge or to gain new knowledge, including topics such as software patterns.

Searching the internet with Google for the terms *"software patterns"* returns approximately 366,000,000 results. Although the internet contains plenty of information on the topic, we saw that popular books such as that of the Gang of Four [1] still remain popular as a source for reference. This is a good indication that these books - and their contents, the software patterns - have survived the test of time, and are currently still of great value in software development.

5.3 Are software patterns mostly used at architectural, design or idiom level?

We have seen software patterns to be mostly adopted at the idiom level, meaning at code-implementation. This comes down to patterns such as the Singleton and Abstract Factory, patterns that are close to the implementation.

The analysis that the software patterns with least abstraction are most often used comes close with the analysis for the previous sub-question. Since the decision to use software patterns is often made by the developers themselves, they are only adopted during implementation, so when writing the actual code, and not during design or requirements analysis. However, if patterns were adopted by - for instance - software architects, we could expect software patterns at a higher level of abstraction to be used, too. Unfortunately, we did not speak with any software architect for this research.

An occurrence of usage of patterns with a higher level of abstraction was observed at the companies that have built a software generator, which are GenerateComp1 and GenerateComp2. They are using patterns with a higher level of abstraction to go along with their models, which enables the model-based engineering of the software generator.

5.4 Are software patterns primarily applied in implicit problem-solving, or are they also explicitly referred to in software development documentation and communication in-house?

We have seen cases of implicit problem-solving guided by patterns at each of the six participating organisations. During their education, the developers have been taught how to use software patterns, and reuse that implicit knowledge to apply software patterns to their implementation of code. This can be either consciously or not.

We did see cases at which explicit references to software patterns are used in software development, for instance at ResearchComp, GenerateComp1, GenerateComp2, Allround-Comp and SCMComp. At the companies building a software generator, explicit use of architectural patterns is found in the models that drive their software generators. AllroundComp and SCMComp seem to use explicit communication by means of patterns during code reviews and to transfer implicit knowledge.

6. CONCLUSION

We have started this research with the question how software patterns are currently used within software developing companies in the Netherlands, trying to find out whether or not there is still a good synchronisation between software patterns that have been documented in the past, and software products that are developed in the present.

We have visited six software developing organisations, conducting a qualitative, semi-structured interview with one of their employees involved in the software development process. Each organisation, product and interviewee had unique characteristics, in order to reach a broad spectrum of factors.

Software patterns are mostly consulted with focus on solving a common software problem, rather than being consulted to gain new knowledge. They are searched through means of literature or digital media, such as the internet. Software patterns are often used at a lower level of abstraction, because it is often the developer who decides to implement patterns in the code. In most cases, patterns are implicitly applied, instead of being used by means of communication amongst colleagues.

This means that software patterns and their accompanied knowledge are of added value in the process of software development, even though they might not be explicitly and consciously implemented in software code. Since software patterns are best practises, and a good developer aims at writing qualitative code, they can assist in reaching the previously named goal.

We may conclude that software patterns are actively used within the Dutch software industry. Thanks to the attention paid to patterns during education, the knowledge becomes implicit and can even be unconsciously applied later during software development. As a consequence, the description of patterns gathered by academia and the way they are communicated with the industry delivers added value in software development.

7. ACKNOWLEDGEMENTS

We would like to express our special thanks to the participating interviewees and their organisations for their time and knowledge, which made it possible to conduct this research. We would like to thank Marc Lankhorst, Sander Weerderburg, Sander Kouwenhoven, Johan den Haan, Arne Wauters and Pascal Durr for their participation.

We would like to express a special thanks to Dr. Eduardo B. Fernandez for reviewing and commenting on the paper in its several stages.

REFERENCES

E. Gamma, R. Helm, R. Johnson, and J. Vissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

H. Gawand, R. Mundada, and P. Swaminathan. Design patterns to implement safety and fault tolerance. *International Journal of Computer Applications*, pages 6–13, 2011.

J. Kabbedijk and S. Jansen. The role of variability patterns in multi-tenant business software. In *Proceedings of the WICSA/ECSA 2012 Companion Volume*, WICSA/ECSA '12, pages 143–146, New York, NY, USA, 2012. ACM.

Kamer van Koophandel. Kvk: Software ontwikkeling, January 2012.