A Requirements Engineering Perspective to Repositories for Interaction Patterns

Christian Graf

Fraunhofer-Institute for Experimental Software-Engineering
Fraunhofer-Platz 1
67663 Kaiserslautern, Germany
Christian.Graf@iese.fraunhofer.de

Abstract. The aim of this position paper is to motivate a requirements engineering approach to pattern repositories for interaction patterns. The paper introduces properties of patterns repository so that they can support all activities and roles involved in a typical development process of software applications.

Requirements engineering looks at stakeholders, what goals they pursue, and what activities they do. When a user interface (UI) is designed, the same is true: activities of the prospective user are in the focus. One technique in designing and implementing UIs is to use interaction patterns. Interaction patterns are proven solutions for design problems, but in contrast to classic software patterns (Gamma et al. 1995) specific for design problems with user interfaces. With the perspective of requirements engineering, interaction patterns must fulfil the requirements of the stakeholders. The same is true for a repository that holds patterns.

Typical stakeholders in a software development process that include UIs are designers and developers. Designers are responsible for creating a pleasant experience with the product usage; while developers assure that this experience can be delivered to the customer. Management is another stakeholder: managers want to decide about key matters of the company, like what development is marketed next. In large companies, a research department plays a significant role by providing knowledge about key technologies for such a development.

From this perspective, it becomes interesting to see what the different people want to achieve when using a pattern repository. Developers and designers typically want to use pattern repositories to search for solutions to specific problems encapsulated in patterns. The designer is focused on design problems, the developer on implementation problems. They both have to be supported very specifically by the same repository. Designers need a description of the context to see if the pattern fits in the design created so far. They probably want to have an illustration or rough sketch of the solution as example. Developers need this support too, in some form of code snippets. Most importantly, they need that support as part of an Integrated Development Environment (IDE). Both, designers and developers may want to comment on patterns they have used, e.g. if they were helpful in a specific project. This information is not useful for anybody outside his or her own context, so it should be kept private.

Management takes high-level decisions that might influence projects: one such decision could be whether patterns developed in the company (e.g. through the researchers) will be restricted to certain departments, distributed in the company, or even made public. The question of the level of distribution becomes more and more important since intellectual property is a key factor of information or knowledge driven businesses. Knowing how to develop software efficiently with proven results and high user acceptance can make the decisive difference between successful and failed businesses. Thus, management needs to be able to consider if a pattern bears such a unique selling point. To do so, there must be a short description of the pattern (synopsis) plus some kind of rating from the in-house experts, if the particular pattern developed in the company holds any uniqueness worth to be kept inside. This requirement is especially valid in the industrial context and motivates private pattern repositories.

The research department could be the source for new patterns or acts as a quality gate for patterns that are already established. It can submit full descriptions of patterns that were found to be useful for the usage in some of the company’s projects. Submitting complete patterns and altering existing ones requires full access to the content of a pattern repository for the research department.

Involvement of other stakeholders might be useful depending on what the company wants to achieve. If the goal is to systematically educate the personnel with regard to the usage of patterns, it might be wise to have the training department overlook how people handle, annotate, and discuss patterns. For this purpose, there must be a) access to the comments to look up what the problems might have been; b) access to the ratings to see if the ratings significantly divert from a mean value (implying a sign of unclear specification in the pattern description or misinterpretations by humans because of lack of knowledge); and c) access to any other information that might represent results from processes around the pattern repository.

A list of requirements for a pattern repository can be constructed from the stakeholders mentioned above. The repository should:

1. Help the different stakeholders by supporting the individual role and its activities to achieve the role specific goals.
2. Minimise the effort to access the repository in the common work environment of the users.
3. Foster the collaboration between the roles involved in the development process by using a common repository which gathers all information in a role dependent way.
4. Represent the experience of the stakeholders handling a pattern, e.g. how easy it was to include it and how successfully it was included in the ongoing project.
5. Represent recommendations on how the pattern is used the best in the specific context of a project or the company.
6. Allow the separation of private patterns from common ones (e.g. on the internet), but on the other hand present patterns from different sources in a concise way.
7. Allow having community (this could be the company's employees only or a broader audience) involvement with suggesting, adding, editing, rating, and commenting patterns

In examining the requirements above, we notice that there are three types of requirements: structural, behavioural, and presentational ones. Structural requirements concern the structure of the repository and what content this structure should hold. Presentational requirements are about how the content is presented to the user. Behavioural requirements define how the user should be able to work with the information provided by the pattern repository.

As a repository holds patterns in a certain format, fulfilling structural reuirements means finding a format that satisfies all structural requirements at once. A large-scale check with the formats of the well-known collections and repositories (Borchers 2001, Graham 2002, Tidwell 2005, van Welie 2006) shows that none of these can satisfy all requirements. The best approach seems to be PLML (Fincher 2003) which offers a wide variety of attributes that cover more aspects than other formats. A repository based on the PLML template can be extended in an object-oriented way with attributes concerning the rating, commenting, recommendations, and experience in the company's context. This holds the advantage that the repository would not store patterns available somewhere else (that might have their own copyright restrictions), but can only reference these patterns and store extended information locally. It permits the control over what extended information is disseminated in addition to the basic pattern in the hand of the specific author. In this way, different companies can augment common patterns with information only valid for their context and development process, without disturbing each other.

Behavioural and presentational requirements can be satisfied on a practical level, via the concrete access structure. Access can be provided through a tool, for example a web front-end, and in general a user interface (UI). The UI allows access to a common repository and provides a means of interaction to support the stakeholders in their activities to let them achieve their goals. A UI presents information from patterns in the repository to the user, and it allows the user to perform desired actions. Such a UI has to serve the user and his requirements, for example let him view, comment, and edit patterns depending on his role in the process and rights to the repository. Integrating a pattern repository with some development process must rely on a framework that can both support stand-alone solutions (for the researcher, designer, and manager to view and edit the pattern) and at the same time an integrated development environment (for the developer to include patterns in his work). One such solution is the Eclipse Platform (Beaton & Rivieres 2006).

Developing pattern repositories along the requirements of the users brings the advantage that the result fits the needs of the ones who should use it. As the repository is the same for all users, issues like different names for the same pattern will become obsolete, and the cooperation among the stakeholders will be streamlined. On the other hand, there is some extra work: one must defined which stakeholder should have access to what attribute in the patterns. If this is done, it will be an advantage too: stakeholders are not confused anymore with information they do not need and cannot interpret.

References

Beaton, W. & Jim des Rivieres (2006). Eclipse Platform Technical Overview. Version 3.1, http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.pdf. International Business Machines Corp. April 19, 2006. Last accessed: 15.06.2007

Borchers, J. (2001). A Pattern Approach to Interaction Design. Chichester, England, John Wiley & Sons, Ltd.

Fincher, S. (2003). "CHI 2003 Workshop Report - Perspective on HCI Patterns: Concepts and tools (introducing PLML)." Interfaces **56**: 27-28.

Gamma, E., R. Helm, et al. (1995). Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.

Graham, I. (2003). A Pattern Language for Web Usability, Addison-Wesley Longman Publishing Co., Inc.

Tidwell, J. (2005). Designing Interfaces, O'Reilly Media.

van Welie, M. (2006). The Amsterdam Collection of Patterns in User Interface Design. http://www.welie.com/index.html. Last accessed: 15.06.2007