# Task Patterns to support task-centric Social Software Engineering

Benedikt Schmidt[1] and Wolfgang Reinhardt[2]

[1] SAP Research CEC Darmstadt, Bleichstrasse 8, 64283 Darmstadt, Germany,
Benedikt.Schmidt@sap.com
[2] University of Paderborn, Institute of Computer Science, Fuerstenallee 11,
33102 Paderborn Germany wolle@upb.de

**Abstract.** Experience sharing to support software engineering is an important, yet difficult task. This paper presents an integration of the Task Pattern concept to support social software engineering. Task Pattern provide information objects and code examples to support software engineering tasks in a community of developers. They are generated based on information resulting from task-centric software development tools, e.g. the Tasktop system. As centrally organized and automatically extended information source they give a valuable insight into the process and product of software development tasks.

**Key words:** task pattern, social software engineering, social information retrieval

## 1 Introduction

Software development can be considered as a specific type of knowledge work [11]. Characteristics like weakly-structured processes, high degree of personal decision involvement and only partial knowledge of the outcome are valid. These aspects complicate adequate support of development processes. In this paper we introduce a concept to support task-centric social software engineering in a team, using the concept of Task Patterns [9]. Task Patterns collect information artefacts created and used by developers when executing software development tasks of similar kind. These artefacts are organized with respect to their meaning for the described task class. With our approach we extend tools like Tasktop Pro for Eclipse [12] which support users in executing their tasks by functionalities for Task Pattern creation and enhancement for experience sharing. This is done by a structured re-use of the information collected by these tools for Task Patterns shared within a community.

This paper is structured as follows. First, we introduce our view on social software engineering and the possible fields of application for social information retrieval in software engineering projects. Second, task-centric software development and the Task Pattern concept are presented as foundation for the idea of using task-centric software engineering to support experience transfer. Third, Task Patterns for software engineering are presented as means for transferring personal experience in a collaborative development environment.

## 2 Conceptual Background

In this section we introduce the concept of social software engineering and show how social information retrieval (SIR) can be applied in software engineering projects.

### 2.1 Social Software Engineering

Software engineering - especially in Open Source - is carried out in (large) teams. The times when developers worked alone on their code seem definitely to be over. Surely this is not a trend of the recent years, as Gerald M. Weinberg stated in 1971 that software engineering is a social user-centered activity where communication plays a key function [15] and that is supported by computers. Computer supported cooperative work (CSCW) is an interdisciplinary field of research focussing on the connection of collaborative work and technical support for it. As Ellis et al. define it *CSCW looks at how groups work and seeks to discover how technology (especially computers) can help them work* [3, p. 39]. So CSCW is part of the research conducted in the area of Computer Mediated Communication (CMC). CMC researches how people and groups communicate using web technologies and services like e-mail, blogs or bulletin boards. Restricting this collaborative work in groups to the domain of software engineering leads to the research area of collaborative (or social) software engineering. Regional and temporal distribution of software development teams require specific techniques for communication and coordination, especially with respect to the domain of knowledge sharing. Collaborative development environments (CDEs) are providing optimal support for coordinating activities and communication in the software engineering process conducted in teams [1].

### 2.2 Social Information Retrieval in Software Engineering projects

Social Information Retrieval (SIR) deals with obtaining information by utilizing social processes. Information can be externalized artefacts as well as personal expertise, so SIR can allow the retrieval of stored data or domain experts. There are different techniques and methods for SIR [14]; techniques include collaborative filtering, subjective relevance judgements and social bookmarking. Methods for SIR comprise recommender systems, social navigation and social search. A mix of the methods and techniques yield an added value for the users of services. In software engineering projects the SIR needs structures to retrieve abstract information like documents and API desciptions as well as more concrete information like source code or best practise examples which fit the engineering requirements and guidelines of the specific team. A CDE can offer a social bookmarking approach allowing developers to bookmark important project-related documents and include these information in the ranking of search results. [6] show how methods and techniques of SIR can be successfully applied in the context of software requirements engineering.

With Task Pattern for software enginnering projects we propose a new type of SIR technique: building patterns from task execution. We will store and analyse tasks of developers, abstract from their specific context and create a Task Pattern from multiple task executions. This Task Pattern will be stored in a central Task Pattern Repository and can be accessed by every developer.

## 3 Tasks and Task Pattern to support organization and execution of tasks

Tasks have proven as effective and efficient means to structure knowledge work. A recent approach is task-centric software development, indicated by e.g. one million downloads of the task-centric development support tool MyLyn [12]. Still, task concepts in knowledge work and also software development are mainly utilized as mechanism to support a user in executing his tasks by organizing information objects. The Task Pattern concept has been proposed to re-use task execution knowledge to share experience and support the retrieval of structured information in a community.

### 3.1 Task-centric Software Engineering

Task-centric Software Development is the organization of programming activities based on tasks using e.g. a task management system. One respective application is MyLyn [5], an extension for the Eclipse IDE which observes the lines of code a user works on for a specific task. MyLyn is based on the following assumption: *The longer a user works on a specific element of the code in a task context, the higher is its relevance for the task.* Empirical studies [13] prove that this assumption holds. Thus the software code can be organized in terms of tasks. If a developer interrupts a coding task and has to come back to it later, MyLyn filters the code resources based on the earlier observed relevant lines of code thus increasing the speed of resource access.

MyLyn allows to share the relevant code parts for tasks in a given project in a community. This allows a community to understand certain lines of code as product of a coding task. This is a good first step, but lacks additional information. The process which lead to the product needs to be transferred additionally to provide more understanding, following Floyds process-product complementary view on software development [4]. Code examples might be worthless for a novice who can not access information describing involved technologies and frameworks. This information most probably has been used during the process of coding. This implicitly has been taken up by the developers of MyLyn, as they recently introduced Tasktop. Tasktop extends MyLyn beyond the observation of code relevant for a task: it observes the interaction with different kinds of information objects like e-mails, documents and web-resources as well. The data collected by MyLyn and Tasktop seems to be a valuable basis for task support beyond the individual organization. Both is integrated in Tasktop Pro for Eclipse. A structure to organize the data respectively can be Task Patterns which are introduced in the following.

### 3.2 Task Patterns to share work experience

Task Patterns are a structure to support individual task execution and to realize tool-supported experience sharing in knowledge work. They hint to working activities and information objects necessary to execute a specific class of tasks. For a user Task Patterns recommend work activities based on Abstraction Services. These Services support users in identifying subtasks, information objects or persons to collaborate with for a specific purpose in a given task class. They are capable to describe aspects of a task on different abstraction levels as described in [10].

Task Patterns are not centrally organized but originate from abstractions of different degrees on user task executions. These executions are explicit in task-centric information system[2] which can be extended by Task Pattern functionality. As such an interplay between a user task and the respective Task Pattern realizes an enhancement life-cycle [7]: by connecting task and Task Pattern and enriching the task by the abstracted pattern data each pattern re-use in the community enhances the pattern, as described in [10]. As such the Task Pattern is a structure for a knowledge worker to learn how to execute a task, meanwhile the structure is implicitly enriched by the knowledge worker himself. The resulting pattern is a community effort as it structures knowledge from real task executions in the community in a re-usable way. In the following we explain the concept of Task Patterns with Abstraction Services to support software engineering activities.

## 4 Task Patterns for Software Engineering

Software development tasks in general result in the creation of program code as product of the tasks. To create the respective program code developers often need to look up different information or learn specific techniques by using different information objects like API descriptions, wikis, development handbooks etc.

### 4.1 Structure and life-cycle of Task Pattern in Software Engineering

As described, task-centric software development uses tools like Tasktop Pro to automatically structure code and resources by means of tasks. We re-use this information for Task Patterns which mesh up the different types of information: code examples which stand for the product and abstract descriptions which help to understand the process which lead to the product. Currently these information types are not centrally organized, as code is spread across the development project source files and other resources need to be accessed via folder structure for files and bookmarks for web-resources. This leads to tedious information-retrieval activities by the developer and a retrieval process which often uses external data-sources like e.g. google code or development forums.

Next to Abstraction Services for information objects like documents, bookmarks or persons which help to structure documents or serve as expert finder, a

Code Abstraction Service serves as central collector for code examples. A Code Abstraction Service basically collects program code which was created to solve tasks described by the Task Pattern. It allows the user to identify relevant passages which can be re-used to execute a new task. Thus it serves as a cheat-sheet to highlight code solving a certain problem in the context of a program.

As different types of information are included in a Task Pattern, a developer can decide to what extent which type of support is necessary. Experts might get support by code examples meanwhile novices need to understand the context and design process itself and have to refer to abstract information. By combining both, Task Patterns allow developers to balance their information requirements.

The creation of a Task Pattern and the life-cycle of integrated Task Pattern use and Task Pattern enhancement is shown in figure 1. After finishing a development task which has been organized using Tasktop the user decides to transform the information into a Task Pattern. To create a Task Pattern a user only has to name a Task Pattern and decide which elements observed by Tasktop are to be wrapped up by Abstraction Services. Abstraction Services allow to structure similar kinds of information and to describe their purpose for the task (Abstraction layer in figure 1). The program code created during the execution of a task gets included in a Code Abstraction Service. The resulting Task Pattern is stored in a public Task Pattern Repository. This repository can be accessed by all developers which can make use of the pattern and enhance it by their respective activities.

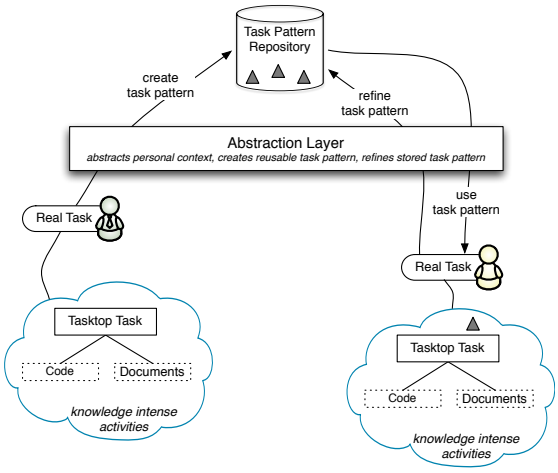


**Fig. 1.** Realization of a life-cycle

### 4.2 An example of Task Pattern in software engineering

The following example shows the application of Task Pattern. We assume that there is a community-embedded collaborative development environment (CCDE) that supports informal communication amongst developers and serves as project management tool for multiple different projects at the same time (cf. [8]). Furthermore we assume that the CCDE hosts multiple projects that connect with the social photo sharing site Flickr (`http://www.flickr.com`). So various developers from different projects try to upload photos to and display photos from Flickr using different programming languages. All developers are structuring their daily work using Tasktop and make use of the possibility to create, enhance and use patterns in the context of their local tasks. This already has resulted in a pattern on accessing and displaying Flickr photos. The pattern includes software code from 5 different tasks of this kind, realizing photo display in different user interfaces. An additional excerpt from the Flickr API and a "how-to" document are attached as well as a short e-mail discussion on caching of photo sets to increase the accessibility.

A developer wants to display Flickr photos. He queries the Task Pattern repository and identifies the described pattern based on its name. He attaches this pattern to his new task. Using Abstraction Services, the developer accesses the code examples but has problems in understanding specific parts of the implemented code. He refers to the attached documents which enable him to understand and re-use parts of the given code. Additionally he asks a colleague on a specific visualization technique including filters and receives a document on this. The document gets included in his tasktop system, as all his activities are tracked by the system. Due to the connection between task and Task Pattern the developer easily can attach the additional information to the pattern. The Code created by him gets automatically attached to the Code Abstraction Service.

### 4.3 Integrating Task Patterns into the IDE

As a realization we propose the extension of Tasktop Pro for Eclipse by interaction functionalities with Task Patterns and Abstraction Services as visible in the mockup fig. 2. A Task Pattern gives access to the Abstraction Services which provide access to documents, web resources, e-mails and example code (1). The example code can be used by an additional code view to browse the program code collected by the Abstraction Service during all different execution processes (2). The task list has to be extended by a mechanism to attach Task Pattern to tasks and the task list must show the attached Pattern with Abstraction Services to the activated tasks. This application is currently under development at the University of Paderborn.

## 5 Discussion

Task Patterns in Software Engineering have been introduced as structure which re-uses task information to realize experience sharing. Thereby Task Pattern

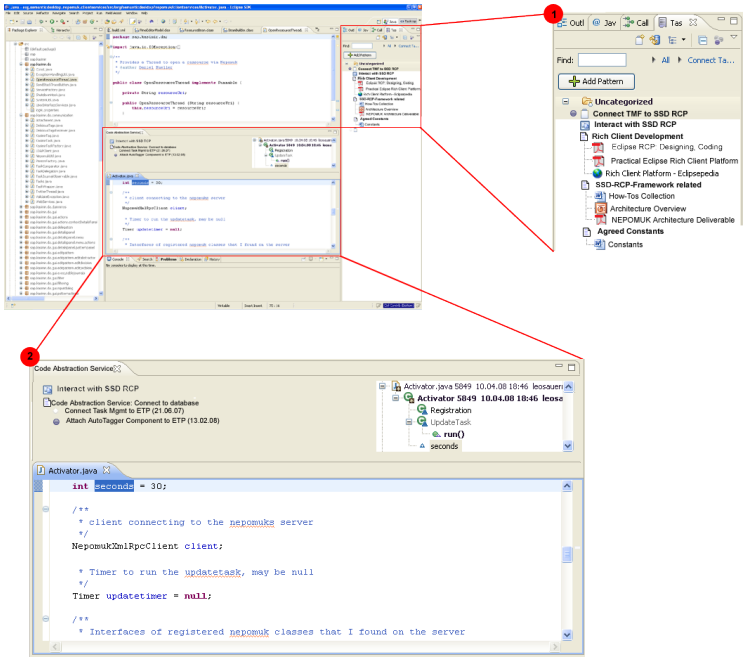


**Fig. 2.** Example for Code Abstraction Service in Eclipse

extend the scope of task-centric software development tools from an organizational and analytical scope towards a tool supported awareness of the execution process. This supports software development as social process, providing shared understanding of problems and awareness of different problem solving types.

The re-use of the Task Pattern concept shows its applicability to different, more specialized domains of knowledge work. The domain of software engineering shows a specific benefit: one can assume that the product of most tasks is code. This allows the easy identification of information resources which represent the process of problem understanding and solving including the resulting code as product. Thus, one can provide a combination of resources to support the process of problem based learning and the examples of working products. The re-use of a tool like Tasktop Pro especially allows the extensive automation of the involved Task Pattern life-cycle [10]. Task Patterns for software engineering focus on support of the individual solution of discrete, individual programming tasks. With embedding Task Patterns for software engineering in a CCDE developers can make use of social processes of recommendation and tagging and thus share experience on similar, re-occurring tasks in a structured form which integrates support for the understanding of the development process as well as for the generated product.

The presented integration of the Task Pattern approach for social software engineering is currently under development at the University of Paderborn within

the scope of a master's thesis. The results will be evaluated and used to enhance the social experience within software engineering projects and help users to find help from others.

## References


1. G. Booch and A. W. Brown. Collaborative development environments. *Advances in Computers*, 59:2–29, 2003.
2. T. Catarci, A. Dix, A. Katifori, and A. Poggi. Task-Centred Information Management. *LECTURE NOTES IN COMPUTER SCIENCE*, 4877:197, 2007.
3. Clarence A. Ellis, Simon J. Gibbs, and Gail Rein. Groupware: some issues and experiences. *Commun. ACM*, 34(1):39–58, 1991.
4. C. Floyd. STEPS-eine Orientierung der Softwaretechnik auf sozialverträgliche Technikgestaltung. *Berichte des German Chapter of the ACM; Vol. 29*, pages 500–503, 1987.
5. M. Kersten and G.C. Murphy. Using task context to improve programmer productivity. In *Proceedings of the 13th ACM SIGSOFT 14th international symposium on Foundations of software engineering*, pages 1–11. ACM Press New York, 2006.
6. S. Lohmann, S. Dietzold, P. Heim, and N. Heino. A web platform for social requirements engineering. In *Workshop Proceedings of Software Engineering 2009*, volume P-150 of *Lecture Notes in Informatics*, pages 309–316, 2009.
7. E. Ong, O. Grebner, and U.V. Riss. Pattern-based task management: pattern lifecycle and knowledge management. In *4 thConference of Professional Knowledge Management (WM 2007), Workshop Integrierte Wissensmanagement-Systeme (IKMS2007), Potsdam, Germany*, 2007.
8. Wolfgang Reinhardt and Sascha Rinne. An architecture to support learning, awareness, and transparency in social software engineering. In *Forthcoming: Special Track on Mashups for Learning, International Conference on Computer Aided Learning (ICL2009)*, 2009.
9. U. V. Riss, A. Rickayzen, H. Maus, and W.M.P. van der Aalst. Challenges for Business Process and Task Management. *Journal of Universal Knowledge Management*, pages 77–100, 2005.
10. B. Schmidt and Uwe V. Riss. Task patterns as means for experience sharing. In *Forthcoming: Proceedings of ICWL 2009*, 2009.
11. M. Schönström and S.A. Carlsson. Methods as knowledge enablers in software development organizations. In *The 11th European Conference on Information Systems (ECIS 2003)*, 2003.
12. Tasktop Technologies Inc. Tasktop pro. `http://tasktop.com/products/`, 03 2009.
13. J. Triesch, D.H. Ballard, M.M. Hayhoe, and B.T. Sullivan. What you see is what you need. *Journal of Vision*, 3(1):86–94, 2003.
14. Riina Vuorikari. Can social information retrieval enhance the discovery and reuse of digital educational content? In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 207–210, New York, NY, USA, 2007. ACM.
15. Gerald M. Weinberg. *The Psychology of Computer Programming.* John Wiley & Sons, Inc., New York, NY, USA, 1971.