An Outlook on Patterns as an Aid for Business and IT Alignment with Capabilities

Janis Stirna[1] and Kurt Sandkuhl[2]

[1] Department of Computer and Systems Sciences, Stockholm University
Forum 100, SE-16440, Kista, Sweden
js@dsv.su.se
[2] Institute of Computer Science, University of Rostock
Albert-Einstein-Str. 22, 18059, Rostock, Germany
kurt.sandkuhl@uni-rostock.de

Abstract. Patterns have established themselves as a useful and practicable instrument for capturing reusable solutions to reoccurring problems in a multitude of domains. This paper discusses three cases of pattern application – at Riga City Council, Kongsberg Automotive, and Proton Engineering, An outlook on how pattern based approaches should be developed to support business and IT alignment and the concept of capability as means to deliver context dependent organizational solutions is also presented.

Keywords: Patterns, alignment, best practices, capability.

1 Introduction

In the process of developing or customizing information systems (IS) we are frequently faced with questions such as: what is the best IT solution to this organizational problem, how should this piece of best practice or experience be used, is it of any value, what can it be used for, when can it be used and by whom. These questions address various aspects of IT use, from management, e.g. concerning IT governance frameworks, to development, e.g. how to customize a particular system component to support company's business process. To answer these questions two main aspects of a knowledge artifact are of importance – what is the problem it addresses and what is the solution it provides.

Alexander [1] defined such problem-solution pairs as patterns – "a problem which occurs over and over again in our environment and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice". Following this principle patterns have been introduced in IS design, data modeling, and in IS analysis. The common objective is to capture, store and communicate reusable artifacts, such as fragments of code or models. Including a set of patterns in a text book on system analysis and design is a de facto standard nowadays. The pattern concept has also been successfully used in organizational development and knowledge management under

L. Iliadis, M. Papazoglou, and K. Pohl (Eds.): CAiSE 2014 Workshops, LNBIP 178, pp. 148–158, 2014.

the term organizational patterns. An overview of six such application cases can be found in [2]. Pattern users appreciate the problem-solution principle of structuring knowledge and can immediately relate their problems to what the patterns address. There are however challenges – at the moment patterns are to a large extent used by people with IT development knowledge, but to ensure efficient business and IT alignment they should also be used by domain experts and business developers.

The objective of this paper is to discuss how the existing drawbacks of pattern methodologies and tools can be improved to support business and IT alignment and the concept of capability.

The rest of the paper is structured as follows. Section 2 gives a brief overview to pattern use in information systems and computer science. Section 3 describes three pattern usage cases. Section 4 discusses the future use of patterns for business and IT alignment while section 5 outlines the main principles of Capability Driven Development and ponders on challenges pertinent to pattern use. Section 6 presents concluding remarks.

2 Pattern Use in Computer Science

Since more than a decade, patterns have been popular in Computer Science and have been used for numerous areas, such as software design, information modeling and business process design. Although there is no generally accepted definition of the term pattern, most publications in the field get some inspiration from Christopher Alexander's definition (see section 1). Whilst Alexander's focus is on the solution, many pattern approaches in computer science concentrate more on capturing proven practices or an advice for how to approach certain problems.

The seminal book on patterns in the area of information system engineering was published by the "Gang of Four" [3] and focuses on software design patterns. Many other books followed, basically offering patterns for all phases of the software development process, including analysis patterns [4], data model patterns [5], software architecture patterns [6, 7], test patterns, etc. The pattern idea was adapted in other areas of Computer Science, like workflow patterns [8], ontology patterns [9], groupware patterns [19] and patterns for specific programming languages [10].

Patterns have also been adopted for organizational design and knowledge management purposes, c.f. for instance [11] and [12]. Furthermore, patterns and anti-patterns have also been used to capture best practices of enterprise modeling in the attempt to improve model quality [13].

Despite the many different fields addressed by these different pattern types, they share certain common characteristics:

- They are based on experiences and deeply rooted in the practice of the field,
- They are not meant to be used blindly as they are. The core idea within the pattern must be understood first and the pattern adjusted or tailored for the specific application case
- They do not only help to build software, processes or models, but also to communicate work approaches within a team or among different stakeholders.

Different approaches can be taken in order to develop or to discover patterns. The existing literature in the field (see above for a selection) suggests at least four possible ways that can be used depending on the nature of the problem and the overall vision for pattern application:

- Pattern detection: use (a large number of) existing development in the area under consideration (e.g. enterprise models, software designs, etc.) and analyze them for recurring parts
- Pattern derivation: use knowledge from related areas (e.g. process models, information flow diagrams, enterprise models) and derive patterns from this knowledge
- Pattern construction: use expert knowledge in the domain and construct patterns based on this knowledge
- Community-based pattern development: use communities of people with knowledge in the field (on the web, wikis, in conferences (e.g. PLoP) or associations) to develop patterns.

In terms of working with patterns we have to consider that there are two dimensions of reuse - design for reuse and design with reuse. By design for reuse we mean the process of identifying valuable solutions in existing or newly created models and creating reusable components, i.e. patterns, from them. By design with reuse we mean the process of creating new organizational designs, e.g. enterprise models, by identifying existing patterns, adapting the solutions, and integrating them with the new solutions created in the project.

3 Examples of Pattern Application Cases

This section presents three pattern allocation cases – in Riga City Council (Latvia), Kongsberg Automotive (Sweden), and Proton Engineering (Sweden) exemplifying the diverse applicability of the pattern concept. All three organizations used patterns for capturing what can be regarded as organizational best practices or know-how. Hence, it is important to point out that the pattern repositories will only create the expected impact if the organizations have supporting processes and roles for knowledge capturing, packaging, storing, searching and applying. Without such a supporting foundation any pattern collection, no matter how competitive and initially innovative, will quickly become obsolete and forgotten. These cases also show that there is a strong demand for supporting the solutions proposed by patterns with information systems.

In 2002 patterns were applied in the Riga City Council (RCC). The RCC wanted to develop an employee knowledge sharing portal where best practices structured according to the pattern format would play a central role (see [14] for details). The RCC had six internal pilot cases located in different organizational units. Patterns were developed by modeling and pattern experts in consultation with stakeholders from the RCC. The resulting patterns included enterprise model fragments, general drawings as well as multimedia content. They were structured by a content

management system (CMS) that was able to automatically suggest hyperlinks based on similar entries in the repository. The users interacted with the patterns via the web interface of the CMS.

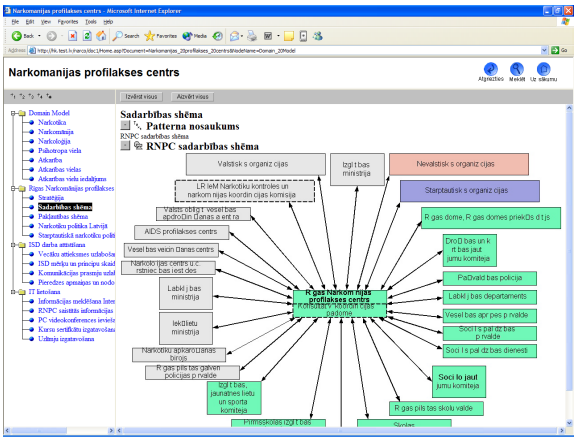

Fig. 1. An example pattern (in Latvian) showing collaborating organizations of the Riga Drug Abuse Prevention Centre

Figure 1 shows an example pattern that explains the collaboration structure within the RCC and how it is supported by various information systems of the involved organizations. This is described by text and conceptual models in the pattern (not shown in the figure). The objective of these patters was to document and share the existing knowledge about RCC's work processes and different best practices used. Some patterns described which information is available in which information system of the RCC or its municipal companies. Beyond that it was up to the pattern users to elaborate the needed connections themselves or ask the RCC's IT department to do it. This can be seen as a drawback that probably contributed to low usage of the pattern repository.

Kongsberg Automotive used patterns from 2006-2008 within the EU-FP6 project MAPPER (Model-adapted Process and Product Engineering) for supporting collaborative engineering in networked manufacturing enterprises by capturing reusable organizational knowledge with Active Knowledge Models (AKM). MAPPER developed c.a. 20, so called, task patterns, which included process, product, organization structure and resources for specific recurring organizational tasks (see [15] for details). The significant difference of the MAPPER project is that the patterns developed were linked to IS components in the METIS tool and the AKM platform, which made the organizational solutions achieved by applying patterns executable. The more or less instant transition from a pattern to a running system was one of the advantages of the MAPPER approach. Figure 2 shows a pattern for establishing a material specification on the left and a functioning workflow system the behavior of which is defined by the pattern.

The drawbacks hindering the adoption of the approach were: (1) the patterns developed in the project only covered a limited area of company's needs and (2) development of new patterns by the domain experts was considered to be too advanced for people without IT development knowledge.

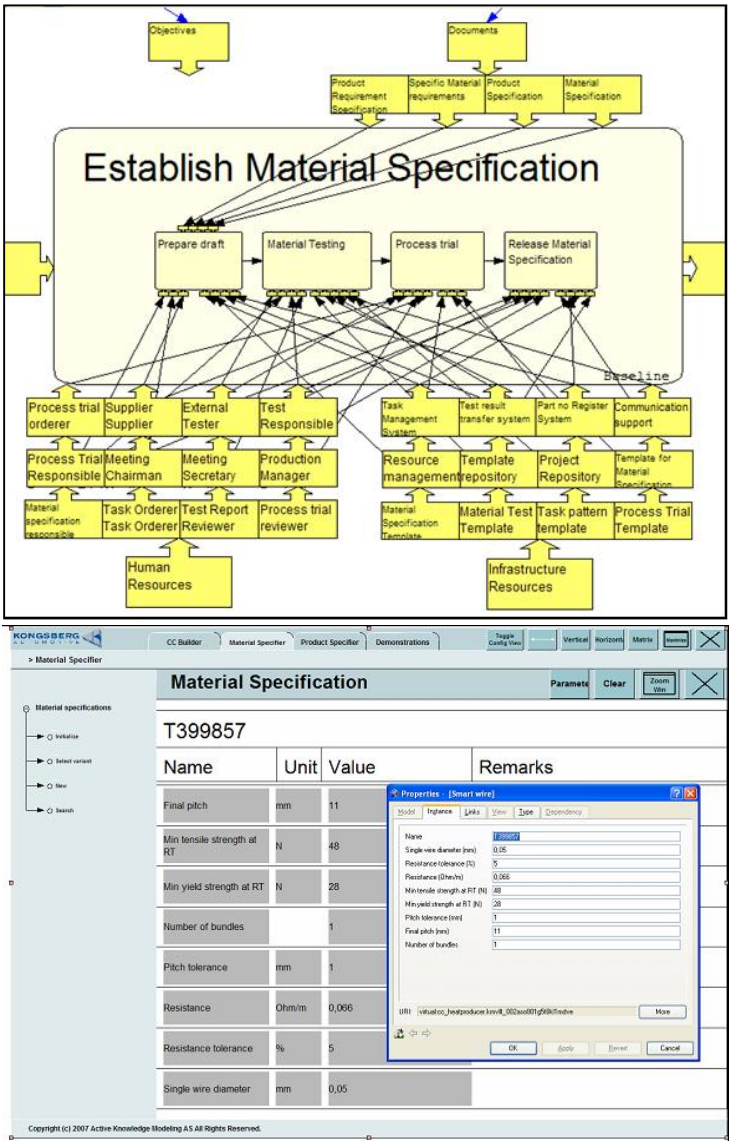

Fig. 2. A pattern in the Metis tool (above) and executed in the AKM platform (below)

Proton Engineering developed and applied information demand patterns within the infoFLOW project during 2009-2012. Proton is a sub-supplier to different first-tier

suppliers in automotive and telecommunication industries who performs various surface treatment services of metal components. Surface treatment in this context includes different technical or decorative coatings to achieve certain functionality or appearance. The patterns were developed for engineering change management (ECM) in the production process. The challenge is to handle the continuously incoming change specifications for products manufactured for many different OEMs in the automotive industry. Not implementing the changes in time would lead to products with wrong characteristics and economic consequences.

An information demand analysis of a specific part of the ECM process (from quotation to production planning) was performed, which resulted in several information demand patterns. *An information demand pattern addresses a recurring information flow problem that arises for specific roles and work situations in an enterprise, and presents a conceptual solution to it.* An information demand pattern consists of a number of essential parts used for describing the pattern: pattern name, organisational context, problems addressed, conceptual solution (consisting of information demand, quality criteria and timeline), and effects.

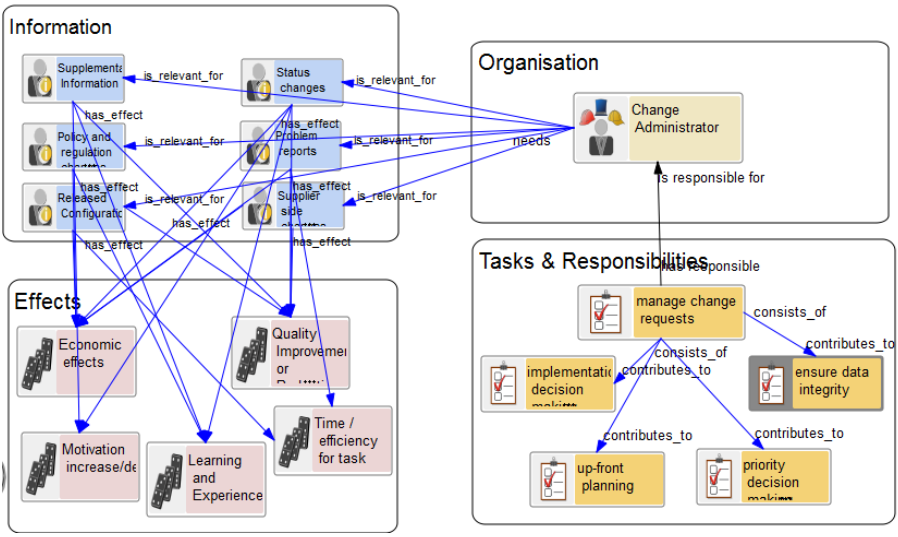

Fig. 3. Visual model of the information demand pattern for the role "change administrator"

Figure 3 shows an example of the information demand pattern for the change administrator role, which is represented as textual description accompanied by a visual model fragment. The advantage of these patterns can also be seen as a disadvantage: they are intended for decision makers in enterprises and focuses on transferring knowledge about how to solve organizational problems related to information flow. Hence, the patterns can be easily understood by domain experts who get hints for how to their problems. However, the domain experts can only take the basic structure of the solution for the problem and have to design the implementation of the solution without the pattern providing details regarding how this should be done.

4 Challenges for Pattern Use in Business and IT Alignment

Sections 2 and 3 discussed the current state of pattern usage in organizations. This section summarizes the current challenges and discuses issues for future work.

There are patterns that present relatively stable knowledge that is unlikely to change soon, e.g. the Gang of Four patterns [3] present a set of foundational solutions for object-oriented design. But there also are other kinds of patterns that are only useful if they take into account the latest IT developments, e.g. the Yahoo pattern case in [16]. Consequently, successful pattern applications require equal attention to (1) the patterns themselves; (2) the process that supports their development and (3) pattern application including user feedback and constant update.

In the future it might become increasingly important to indentify which parts of a pattern require updating. In some cases when patterns are used by a large group the challenge is to discover the needs of the group and to amalgamate all the different feedback. A grassroots approach of allowing each user to suggest candidate patterns in a development environment could therefore be helpful. As an additional benefit the company would be able to see the different solutions that the employees are using and assess the current situation, indentify gaps, as well as plan development actions.

Patterns reside in repositories, content management systems, tools, wikis and other kinds of collaboration platforms. Relationships among patterns as well as with other internal and external information sources are usually established manually by the pattern developers. This is a tedious and often neglected task with little or no automation support. As a result it is not done thoroughly and users are left wondering what else (e.g. other supporting technologies, policies, risks) is relevant to the proposed solution and where the relevant information can be found. While there are tools that are able to automatically find and recommend hyperlinks, such functionality is not sufficiently developed and widely used. It should also be extended towards automatic web service discovery.

Most patterns are described in text supported by code or model fragments. Regardless of the technology used for representation, users have to search, assess suitability, and decide by themselves how to apply the proposed solution. There are contributions that include patterns in development environments, e.g. design patterns in CASE tools such as ModelMaker or the EUREQA approach and tool presented in [17]. But currently this is done only with design patterns addressing IS design problems, similar to the Gang of Four patterns, with software patterns containing reusable code, or with workflow patterns. While this is good starting point, a more explicit connection should be established between various business problems and IT solutions. In the future model driven development tools should be able to connect business problems to patterns and executable components as well as allow users to create and add their own. This would improve traceability and transition between the business requirement elicitation and IS development. In essence, we should be striving towards a modern kind of patterns that are business problem-solution-execution triplets.

5 Outlook on Patterns for Capability Delivery

In the recently started FP7 project CaaS – "Capability as a Service for digital enterprises" patterns are used within capability design and delivery. The ethos of the project is to facilitate development of business solutions requiring customization as the context of delivery changes. The CaaS project aims to facilitate configuration of business services and development of executable software to monitor the fitness of purpose of these services to evolving business contexts and where necessary to adjust these services according to the context. Patterns will be used for delivering context dependent organizational capabilities as specified in the meta-model of the project's approach [18].

Capability is the ability and capacity that enable an enterprise to achieve a business goal in a certain context. It describes the capability of the business that will be designed and delivered. Capability formulates the requirements for the ability of accomplishing a business goal, realized by applying a solution described by a capability delivery pattern. Patterns are reusable solutions for reaching business Goals under specific situational contexts. The context defined for the Capability (Context Set) should match the context in which the Pattern is applicable.

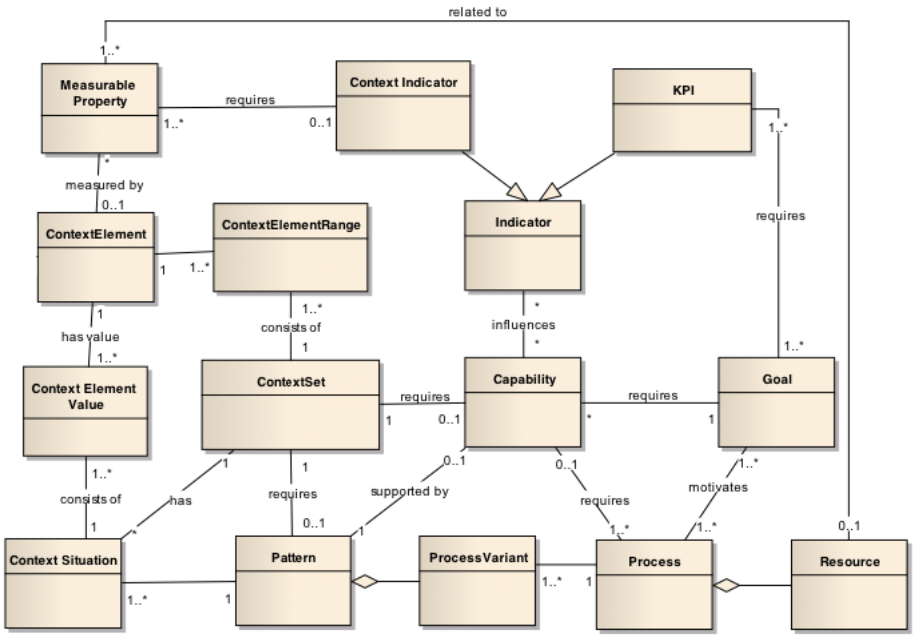

Fig. 4. Capability meta-model [18]

In the CaaS project patterns will represent reusable solutions in terms of business processes, resources, roles and supporting IT components (e.g. code fragments, web service definitions) for delivering a specific type of capability in a given context.

The Context indicators are used to monitor at runtime whether the pattern applied for delivering the capability is valid for the specific context situation. If the pattern is not valid, then the capability delivery should be dynamically adjusted by applying a different pattern, by reconfiguring the existing pattern or by aggregating several patterns into a new pattern. More about the overall vision of Capability Driven Development and the CaaS project is available in [19] and [20].

Concerning the use patterns the following challenges will have to be addressed in terms of model representation, way of modeling, IS support and organizational support.

In terms of model representation: the approach will be model driven and hence patterns will have to be represented in a model form. The relationships between pattern and its application context (represented by context set in the meta-mode) and the solution the pattern proposes (represented by a process variants) will have to be supported by methodological guidelines concerning issues such as efficient ways of modeling context, modeling process variants as part of patterns etc.

In terms of the way of modeling: both dimensions of reuse will have to be supported – pattern discovery and development from existing enterprise models as well as using patterns in constructing solutions for capability delivery. The main challenges are keeping up to date links between patterns and the initial models they are based on, as well as managing traceability between the capability delivery solutions and the patterns that are included in each solution.

In terms of IS support: patterns will be executable in the sense that they will be linked with information system components and will be used adjusting the application at run time depending on the changes in context. Hence, the pattern concept should be able to deal with run time adjustment algorithms and monitoring context and resources.

In terms of organizational support: Patterns have proven to be useful as reusable organizational solutions and hence they should be supported by an organizational knowledge cycle that supports activities such as creation, capturing, packaging, storing, sharing, applying knowledge artifacts (patterns) as well as transforming and innovating leading to creation of new knowledge artifacts [21]. These activities will have to be supported by the Capability Driven Development approach in order to ensure that the target organization is using patterns that are up to date and valuable for the organization. To achieve this, specific work procedures, e.g. for collecting feedback, and responsibilities will have to be established.

6 Concluding Remarks and Future Work

This paper presents three cases of pattern application that we consider being typical in a setting where organizational knowledge needs to be integrated with IT support. To a large extent patterns have been used for improving business and IT alignment by offering means for capturing, documenting and sharing best practices. As the application cases discussed in the paper suggest, patterns have the potential of linking business solutions with IT solutions. The current state of the art in the respect is,

however, not offering practicable solutions, especially in cases when business solutions should be tailored according to context changes. To this end a proposal to use the concept of capability has emerged that further extends the use of patterns.

Another area of future work will be related to the different usage scenarios of patterns. Development of patterns on the CaaS project indicated the possibility to distinguish between solution-oriented and design-oriented patterns. Solution-oriented patterns incorporate how to solve a context related problem at runtime, i.e. the degree of freedom in these patterns is reflected in how to combine executable components. This characteristic puts such patterns close to software services, but there is still an important difference: services work as solution elements on their own with defined interfaces and deterministic behavior. Solution-oriented patterns need composition and decision logic at runtime, and hence we needs to know how to compose and configure them. Design-oriented patterns are closer to the "traditional" meaning of software, ontology and workflow patterns: they capture the core structure and elements of a design solution, which needs to be incorporated into a business service design. Examples are parts of processes or variations for defined context settings.

Both pattern types have been identified in CaaS; examples were developed and their differences exposed. However, the connection between the pattern types has to be explored in more detail. Design-time patterns can be further developed and combined at design time to solution-patterns. Such a development process basically is similar to a conventional engineering process. But there also is a potential of using design time patterns to model the decision logic for how solution-patterns have to be combined and configured at runtime. To this end we need to capture the flow of the decision process and part of the logic. Elaborating this connection will be part of the future work.

Furthermore, we will investigate the possibility to use the same pattern representation for both types. On a very general level, we will need the typical "gang-of-four" pattern structure of context, problem, solution and effects. For the problem and solution part we will furthermore have to link both pattern types to business services. Whether or not this can be done with the same mechanisms needs further exploration.

Acknowledgement. This work has been partially supported by the EU-FP7 funded project no: 611351 CaaS - Capability as a Service for Digital Enterprises.

References

1. Alexander, C.: A pattern language. Oxford University Press, New York (1977)
2. Niwe, M., Stirna, J.: Organizational Patterns for B2B Environments –Validation and Comparison. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. LNBIP, vol. 29, pp. 394–406. Springer, Heidelberg (2009)
3. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software Architecture. Addison Wesley, Reading (1995)
4. Fowler, M.: Analysis patterns: Reusable object models. The Addison-Wesley series in object-oriented software engineering. Addison-Wesley, Menlo Park (1997)

5. Hay, D.C.: Data model patterns: Conventions of thought. Dorset House, New York (1995)
6. Fowler, M.: Patterns of enterprise application architecture. The Addison-Wesley signature series, vol. 2. Addison-Wesley, Boston (2003)
7. Buschmann, F., Meunier, R., Rohnert, H., Schmidt, D.C., Henney, K., Sommerlad, P., Stal, M.: Pattern-oriented software architecture. Wiley series in software design patterns. Wiley, New York (2000)
8. Van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases 14, 5–51 (2003)
9. Blomqvist, E., Sandkuhl, K.: Patterns in Ontology Engineering: Classification of Ontology Patterns. In: Proc. of ICEIS 2005 (2005)
10. Beck, K.: Smalltalk best practice patterns. Prentice Hall, Upper Saddle River (1997)
11. Rolland, C., Stirna, J., Prekas, N., Loucopoulos, P., Persson, A., Grosz, G.: Evaluating a Pattern Approach as an Aid for the Development of Organisational Knowledge: An Empirical Study. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, pp. 176–191. Springer, Heidelberg (2000)
12. Persson, A., Stirna, J., Aggestam, L.: How to Disseminate Professional Knowledge in Healthcare. Journal of Cases on Information Technology 10(4), 41–64 (2008)
13. Stirna, J., Persson, A.: Anti-patterns as a means of focusing on critical quality aspects in enterprise modeling. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. LNBIP, vol. 29, pp. 407–418. Springer, Heidelberg (2009)
14. Mikelsons, J., Stirna, J., Kalnins, J.R., Kapenieks, A., Kazakovs, M., Vanaga, I., Sinka, A., Persson, A., Kaindl, H.: Trial Application in the Riga City Council, deliverable D6, IST Programme project no. IST-2000-28401, Riga City Council, Riga, Latvia (2002)
15. Sandkuhl, K., Stirna, J.: Evaluation of Task Pattern Use in Web-based Collaborative Engineering. In: Proc. of the 34th EUROMICRO. IEEE (2008) ISBN 978-0-7695-3276-9
16. Malone, E., Leacock, M., Wheeler, C.: Implementing a pattern language in the real world: A Yahoo! case study. Boxes and Arrows (2005), http://boxesandarrows.com/implementing-a-pattern-library-in-the-real-world-a-yahoo-case-study/ (accessed March 28, 2014)
17. Austrem, P.G.: The EUREQA Approach: A Method and Tool for Leveraging Design Patterns in End-User Development, PhD thesis, Univ. of Bergen, Norway (2011)
18. Bērziša, S., Bravos, G., Gonzalez Cardona, T., Czubayko, U., España, S., Grabis, J., Henkel, M., Jokste, L., Kampars, J., Koc, H., Kuhr, J., Llorca, C., Loucopoulos, P., Juanes Pascual, R., Sandkuhl, K., Simic, H., Stirna, J., Zdravkovic, J.: Deliverable 1.4: Requirements specification for CDD, CaaS – Capability as a Service for Digital Enterprises, FP7 project no 611351, Riga Technical University, Latvia (2014)
19. Zdravkovic, J., Stirna, J., Henkel, M., Grabis, J.: Modeling Business Capabilities and Context Dependent Delivery by Cloud Services. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 369–383. Springer, Heidelberg (2013)
20. Egido, J.C., González, T., Llorca, R., Grabis, J., Stirna, J., Zdravkovic, J.: Deliverable 1.3: Vision of the CDD methodology, CaaS – Capability as a Service for Digital Enterprises, FP7 project no 611351. Everis, Spain (2013)
21. Persson, A., Stirna, J., Aggestam, L.: How to Disseminate Professional Knowledge in Healthcare – The Case of Skaraborg Hospital 3, 42–64 (2008), Journal of Cases on Information Technology 10 (2008) ISSN: 1548-7717