Capability Patterns as the Enablers for Model-based Development of Business Context-aware Applications

Janis Stirna[1], Jelena Zdravkovic[1], Martin Henkel[1], Janis Kampars[2]

[1]Department of Computer and Systems Sciences, Stockholm University, Forum 100, SE-16440, Kista, Sweden
[js, jelenaz, martinh]@dsv.su.se
[2]Information Technology Institute, Riga Technical University
Kalku iela 1, Riga, Latvia
Janis.Kampars@rtu.lv

Abstract. Lately the notion of capability has emerged in information system engineering as the means to support development of context dependent organizational solutions and supporting IT applications. To this end a Capability Driven Development (CDD) approach has been proposed. As key part of CDD is the concept of *patterns* that is used to support the capability design from existing solutions as well as the adjustment of the capability delivery at run-time. A pattern template and meta-model is presented together with the CDD lifecycle that incorporates pattern development and use. The initial experiences of use of the patterns as part of CDD at three industrial use cases are also presented.

Key words: Enterprise Modeling, Capability Modeling, Patterns.

1 Introduction

Enterprises are facing the need to adapt their businesses according to various situations in which their applications need to be used. To answer this challenge an EU FP7 project "Capability as a Service in digital enterprises" (CaaS) has been initiated [1]. The aim of CaaS is support of the capture and analysis of changing business context in design of information systems (IS) using the capability notion.

In the specification and design of services using business planning as the baseline, capability is seen as a fundamental abstraction to describe what a core business does [2] and, in particular, *as an ability and capacity for an enterprise to deliver value, either to customers or shareholders, right beneath the business strategy* [3, 4]. The key rationale behind a capability driven approach to software development is to make IS designs more accessible to business stakeholders by enabling them to use the capability notion to describe their business needs more efficiently.

An important problem in the prevailing Model Driven Development (MDD) paradigm for IS development mostly relies on the models defined on a relatively low abstraction level. In contrast, Enterprise Modeling (EM) captures organizational knowledge and provides the necessary motivation and input for designing IS. According to practitioners [5], EM needs to be placed in the context of solving problems, which, due to today's geo-political situations and monetary crises, are becoming more complex, and more dynamic. Hence the organization needs to have a high discipline to bring EM and its IS solutions just-in-time and just-enough. Dynamics of business requires support from modeling and IS; at the same time that dynamics prevents optimal use of modeling because organizations cannot really match the pace of change with a right approach to EM. Models are often large and complex, expensive and taking lot of effort to produce. And hence they are not matching the dynamics of the organizational change; even in the best case when a valuable artifact is produced, it is typically not aligned with dynamics and therefore not used. That means the investment is lost and the business will not do such an investment again [5]

A solution to the described problem is in the CaaS project initiative seen by integrating the principle of reuse and execution of software patterns with the principle of sharing best practices of organizational patterns [6, 7]. Hence, *capability patterns* are organizational design proposals that can be executed, adapted, and reused. The patterns can thus represent reusable solutions in terms of business process, services, resources, roles and supporting IT components (e.g. code fragments, web service definitions) for delivering a specific type of capability in a given context. They can be adapted and reused. The main objective of the capability pattern notion is to facilitate enterprise-size components carrying a software solution for an organizational problem in a given business context, in both the design and run times.

The rest of the paper is organized as follows. Section 2 briefly describes the origin and existing pattern efforts within the IS engineering discipline, and the Capability Meta-Model. Section 3 starts by describing the content of the capability pattern and further describes the life cycle for Capability Driven Development (CDD) from the capability pattern perspective. Section 4 presents our experiences of using developing patterns as part of CDD at three industrial use cases as part of the CaaS project. Section 5 provides concluding remarks and issues of future work.

2 Theoretical Foundations and Related Work

In this section brief overviews of the topics and the results related to the research of this paper are presented.

2.1 Pattern Origin and Use

In the process of capturing, packaging, storing, and sharing knowledge we are frequently faced with questions such as: how should this piece of best practice or experience be represented, is it of any value, what can it be used for, when can it be used and by whom. These questions address the two main aspects of a knowledge artifact – what is *the problem* it addresses and what is *the solution* it provides [8]. Following this definition, pattern based approaches have established themselves in software programming, software design, data modeling, and in systems analysis, see e.g. [9, 10] with the common objective to capture, store and communicate reusable artifacts, such as fragments of code or diagrams.

The pattern concept has been further extended and applied in organizational development and knowledge management under the term organizational patterns [11]. According to this principle, patterns have been successfully applied in a number of projects for knowledge sharing purposes,

As for the EM discipline, its projects create a great deal of models. They have various purposes, e.g. some are created only to capture a particular idea or document the discussion of the stakeholders while others document a specific business design, for instance, as business processes, service coordination, or data structures. The majority of models are created in a design situation and once completed they reflect good solutions and best practices for dealing with a specific business problem or corporate intention, some of which can be captured and represented as patterns.

In section 3 we will explain the use of patterns within the CDD – their content and the use differ from / go beyond existing pattern proposals in the way that a) their granularity and the scope corresponds to another enterprise notion, i.e. capability, b) their life-cycle spans from business analysis, through design, and to run-time when upon monitoring the patterns are replaced, added, removed or adapted.

2.2 Capability Meta-model

The theoretical and methodological foundation for pattern use in capability-oriented software applications is provided by the core capability meta-model (CMM) in Figure 1, and in details presented in [12]. CMM is developed on the basis of requirements from the industrial project partners, and related research on capabilities. Within Capability Driven Development (CDD), patterns are envisioned as reusable solutions for reaching business goals under specific situational context. Individually, they are intended to describe best practices for businesses, and in a collection to form a repository of capability delivery patterns.

In brief, the meta-model has three main sections: a) Enterprise model representing organizational designs with Goals, KPIs, Processes (with concretizations as Process Variants) and Resources; b) Context represented with Context Situation instantiating a set of elements (such as Context Element, Range, and Set) under which the solutions should be applied, including Context Indicators for measuring the context properties (Measuring Property); and c) Patterns for delivering Capability by reusable solutions for reaching Goals under different Context Situations.

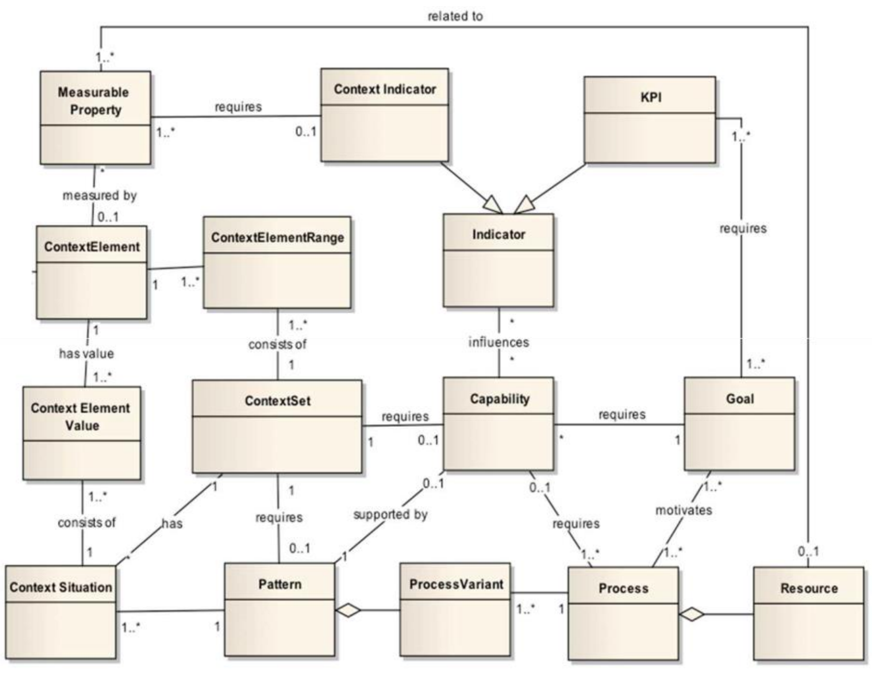


Fig. 1. A core meta-model for supporting Capability Driven Development.

The Pattern component describes an actual solution for realizing a Capability. Each pattern describes how a certain Capability is to be delivered within a certain Context Situation and what Processes Variants and Resources are needed to support a Context Set. Pattern consists of Patterns: Pattern may represent a complete solution or it can consist of several “sub-patterns” each of them focusing on a specific part of the solution. At runtime patterns are used according to the Context Situations representing a set of actual context values with their Measurable Properties. The Context KPIs are used to monitor at runtime whether the capability realization through patterns is still valid for the current context situation. If a capability pattern is not valid, then the capability realization should be dynamically adjusted, for example by applying a different pattern.

3 Pattern Representation and Way of Working

The purpose of including patterns in the CDD approach is to capture and represent reusable knowledge in the model form that can be used for capability design and delivery. This section will discuss the form used to represented patterns, including an elaborated meta-model for concepts that are specific to patterns (section 3.1), and the way of working with patterns with respect to the CDD lifecycle (section 3.2).

3.1 Pattern Structure and Meta-model

Patterns are typically described according to a pattern template. We have chosen a template that is fairly simple and consists of a few fields that are commonly used in pattern descriptions (see Table 1). The information represented in this template is also represented in the meta-model (see **Fig. 1**); terms highlighted in Courier refer to concepts in the meta-model

The components of the pattern need to be linked with the rest of the components in the CDD meta-model as shown in figure below. It shows how the different components of the pattern template are connected to other elements in the CDD. More specifically, the solution body can be a business process model fragment (`Process Variant`) or a fragment of another model type. If the `Solution Body` is a process model fragment then gateways are used as `Process Variant Variation Points`. For other model types there is a possibility to specify a generic `Capability Delivery Variation Point`. At this stage of the project the variation points are specified in the Capability Design Tool, but at later stages they will be linked to executable services in the Capability Delivery Application, as well as with `Capability Adjustments` that will be performed by the Capability Navigation Application according to the actual `Context Element Values` at run-time.

Table 1. Pattern template used for CDD

Name of the field	**Purpose of the field**
Name:	Each pattern should have a `name` that reflects the problem/solution that it addresses. Names of patterns are also used for indexing purposes.
Problem:	Describes the issues that the pattern wishes to address within the given context and forces.
Context:	Describes the preconditions under which the problem and the proposed solution seem to occur. This can initially be expressed in free text and later represented by creating a `Context Set` that encompasses the permitted `Context Element Ranges` of `Context Elements` that influence the applicability and variability of the solution proposed by the `Pattern`.
Solution:	Describes how to solve the `problem` and to achieve the desired result. The solution field consists of a textual `solution description` and a `Solution Body` in the form of a model fragment. Currently we focus of `Process Variants` expressed using the BPMN. The `Process Variants` may also contain `Variation Points`. Other model types are also possible to use in order to represent the solution
Usage Guidelines:	Presents a set of usage tips to the potential user of the pattern about how the pattern can be tailored to fit into particular situations or to meet specific needs of an organization. As well as how the

	`Solution Body` should be integrated with an existing capability design.
Keywords:	A few keywords are defined for each pattern in order to facilitate search and retrieval.
Adjustment algorithms:	Links to specific `Capability Adjustments` for a specific `Variation Aspect`.

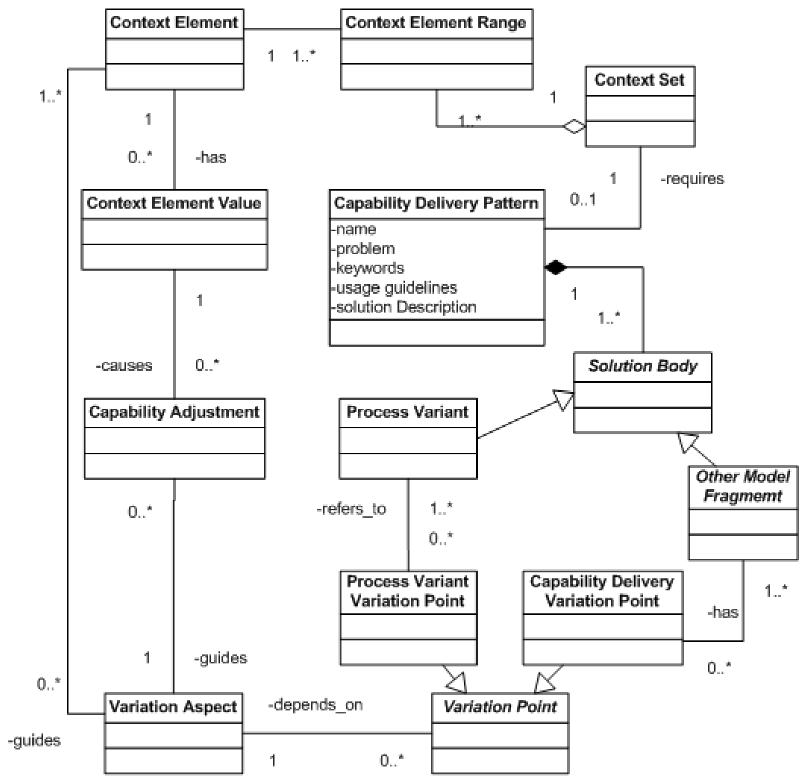


Fig. 2. Elaboration of the meta-model components of related to specifying pattern.

We illustrate the above meta-model with a pattern example from one of the use cases of the CaaS project at company Fresh T Limited, UK (FreshTL). The overall capability is to ensure maritime compliance of ships of one of FreshTL's customers. The capability requires a process for maritime compliance (process 1, in Fig. 3). In a development situation this process can be taken from a pattern repository and used as best practice to deliver the capability. It is dependent on a number of external factors (context) one of them is if the ship is in an environmentally sensitive or protected geographic area. In such cases the process for generic compliance should be changed with a specific process addressing environmental compliance. This adjustment is depended on context element (CE) "area". Hence, the applicability of the pattern that provides the process variant for rule compliance for emissions and environmental compliance is determined by CE: area and CE range: Environmentally sensitive or protected. The connection between the main process used for capability delivery and the process variant is achieved via variations aspects (VA) that determine which context elements determine choosing which patterns.

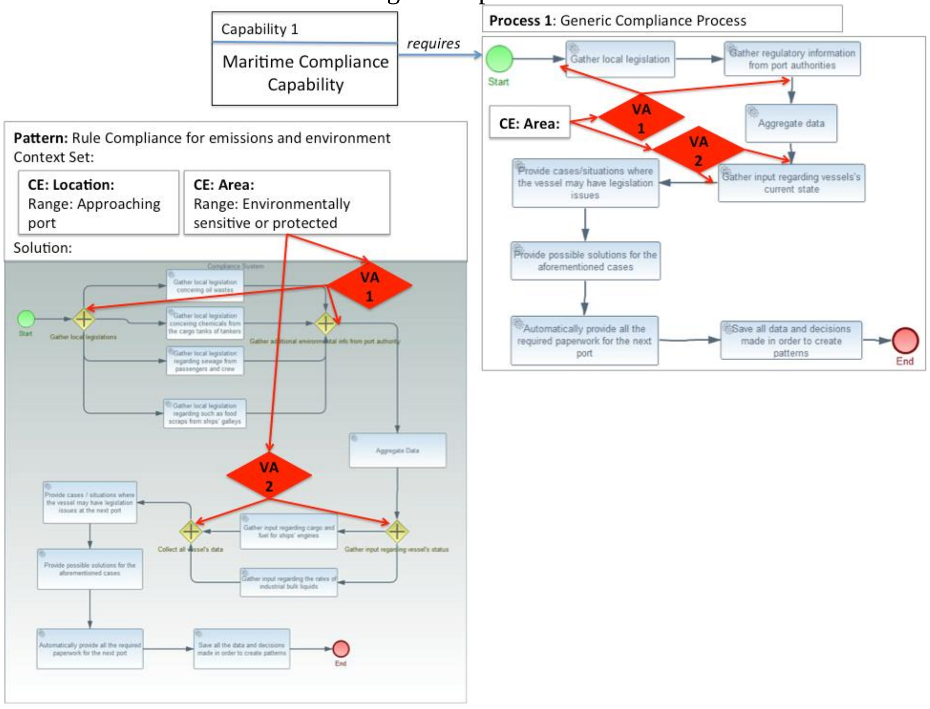

Capability 1
Maritime Compliance Capability
requires
Process 1: Generic Compliance Process
Start
Gather local legislation
Gather regulatory information from port authorities
CE: Area:
VA 1
VA 2
Aggregate data
Provide cases/situations where the vessel may have legislation issues
Gather input regarding vessels's current state
Provide possible solutions for the aforementioned cases
Automatically provide all the required paperwork for the next port
Save all data and decisions made in order to create patterns
End
Pattern: Rule Compliance for emissions and environment
Context Set:
CE: Location:
Range: Approaching port
CE: Area:
Range: Environmentally sensitive or protected
Solution:

Fig. 3. Example of pattern use for specifying a design for adjusting according to context.

3.2 Pattern Creation, Use, and Update

The overall Capability Driven Development (CDD) process includes three cycles (see **Fig. 4**): 1) capability design; 2) capability delivery; and 3) capability refinement/updating. As explained in section 2, capability delivery patterns are the solutions for delivering capabilities meant to be sufficiently general and reusable in long-term.

Figure 4 and the text below explain how the patterns are created, used and updated within the CDD process.

The capability development cycle starts with *Enterprise Modeling*, i.e. by a business request for a new capability - the request might be initiated by strategic business planning, changes in context, or discovery of new business opportunities requiring reconfiguration of existing or the creation of new goals, business processes, and other EM elements. This is followed with a *formalized definition of requested capabilities and definition of the relevant contexts* according to CMM (section 2).

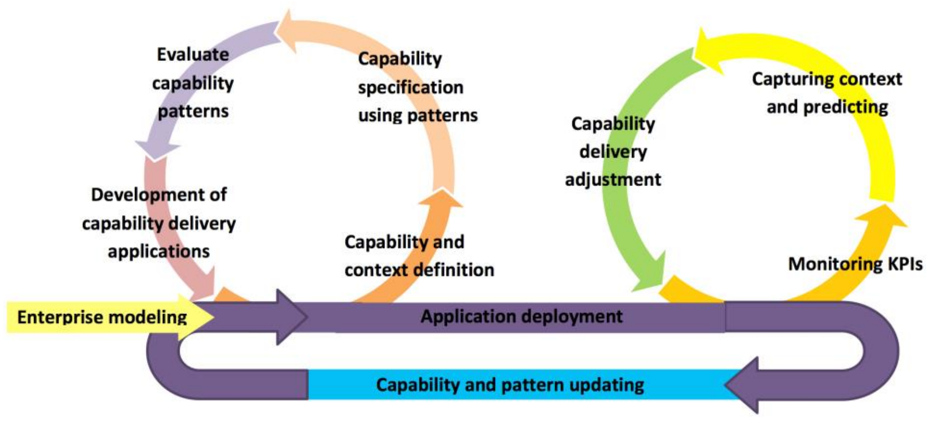

Evaluate capability patterns
Capability specification using patterns
Capability delivery adjustment
Capturing context and predicting
Development of capability delivery applications
Capability and context definition
Monitoring KPIs
Enterprise modeling
Application deployment
Capability and pattern updating

Fig. 4. A life-cycle for CDD with a focus on the pattern's perspective.

For a capability that is successfully defined, its delivery is then *specified by a capability pattern* assigned either by selecting it from a pattern repository or by creating a new pattern. A prototype of the repository is illustrated in section 4. A new pattern is created following its model-based structure in details presented in the next section. The candidate pattern can be composite, i.e. include other patterns (see CMM, Figure 1) – this means that the pattern modeler has to match the needed capability to its exact solution; for instance a capability "Maritime Compliance Capability" would require a matching capability pattern, which can in turn be composite, i.e. containing aggregated patterns such as "Rule Compliance Pattern for vessel's cargo", "Rule Compliance for emissions and environment", "Rule Compliance Pattern for medical issues" etc.

The capability designed is assessed by evaluation of the pattern for its business and technical feasibility. If capability delivery is deemed feasible, business structures and software development enabling capability delivery are put forward.

A capability delivery application (CDA) is either developed or an existing CDA is linked to the capability design. This is done by following the development process used by a company, where the model of the selected capability pattern (if not earlier developed) provides the model-based requirements for the context data and the processes (or other model types) that are to be developed and integrated using company's technology architectures, platforms and software services.

The delivery cycle starts with the deployment of the CDA in a target environment; the capability delivery navigation application (CNA) *monitors the fulfilment of capability delivery KPIs.* If for the mentioned compliance delivery pattern example a KPI defined as the number of ships compliant, then, since it must be 100% at all times, its fulfilment is continuously followed, and in conjunction with possible changes in the context (for example, near to environmentally protected areas, having special medical cases on ship).

If capability is not delivered as requested by the KPIs, the delivery adjustment algorithms are invoked to modify the capability delivery by replacing the pattern in use, by calculating if the changes are become such to require another capability pattern or that one or more of other aggregated patterns to support the required KPI values are needed.

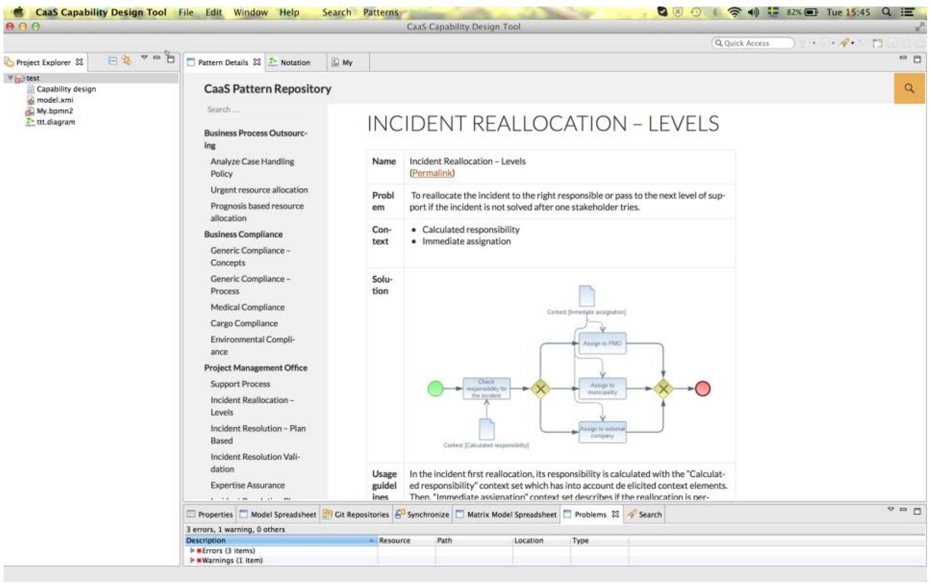
The capability refinement and pattern updating is initiated according upon the results of capability delivery monitoring and adjustment. These results indicate validity of selected capability delivery patterns and capability delivery patterns are, if indicated, updated accordingly. This is considered due to an expected need to account for additional context factors because typically not all relevant factors (such as all relevant context elements, or all variation points) can be identified during the first development iteration. Initially, the run-time adjustment algorithms attempt to modify execution of the CDA by replacing patterns; if that is not sufficient, the capability definition is refined, additional elements are included, as well as capability delivery patterns are redesigned or new patterns added.

4 Experiences of Developing and Applying Patterns for CDD

This section briefly explains the current use of patterns in the CaaS project and summarizes our experiences at the three industrial use cases, namely

1. At Everis (Spain) for service promotion capability, marriage registration capability, SOA platform capability
2. Fresh T Limited (UK) for maritime compliance capability
3. SIV AG (Germany) for standard business processes execution capability.

Patterns have been captured and documented in a web-based pattern repository and integrated with the Capability Development Tool.



CaaS Pattern Repository

Business Process Outsourcing
- Analyze Case Handling Policy
- Urgent resource allocation
- Prognosis based resource allocation

Business Compliance
- Generic Compliance – Concepts
- Generic Compliance – Process
- Medical Compliance
- Cargo Compliance
- Environmental Compliance

Project Management Office
- Support Process
- Incident Reallocation – Levels
- Incident Resolution – Plan Based
- Incident Resolution Validation
- Expertise Assurance

INCIDENT REALLOCATION – LEVELS

Name	Incident Reallocation – Levels (Permalink)
Problem	To reallocate the incident to the right responsible or pass to the next level of support if the incident is not solved after one stakeholder tries.
Context	• Calculated responsibility • Immediate assignation
Solution	
Usage guidelines	In the incident first reallocation, its responsibility is calculated with the "Calculated responsibility" context set which has into account de elicited context elements. Then, "Immediate assignation" context set describes if the reallocation is per-

3 errors, 1 warning, 0 others
Errors (3 items)
Warnings (1 item)

Fig. 5. Example of a pattern opened via the Capability Design Tool

The reason for maintaining a pattern repository within CDD is to enable retrieval of patterns for understanding business capabilities, and for using them in different occasions and domains. Reuse through a capability pattern repository intends to facilitate a systematic practice of designing capabilities, so that their similarities between business units, organizations and industries can be exploited to achieve benefits in business performance.

Concerning the use of the pattern template and the meta-model, the current version of patterns should be seen as an initial version. The fields in the template are complete but we have not used the modeling approach to connect the representation of context sets to the actual context elements and relevant variations aspects. This is because at the initial stages the users did not have a complete view on what context elements could be available at run-time and how they could influence the capability delivery. The project has been primarily performing capability design and the capability delivery processes is only beginning now. As a result the process variants in the patterns are not yet connected to the variation points and variation aspects. More of this knowledge will be discovered once the patterns will be connected to the Capability Delivery Applications and the experiences of the use gathered.

At this stage of the project we have mainly been focusing on the procedural aspects of capability design and delivery. I.e. the solutions offered by the patterns are expressed by process variants in all patterns except one that contains a conceptual model of key concepts of compliance. At later stages we envision developing more patterns that contain business rules and capability adjustment algorithms.

Concerning the pattern development lifecycle and its alignment with the CDD lifecycle, the pattern development process was iterative. The candidate patterns were developed by the use case experts and then suggestions for improvement were provided by the method experts. For most patterns it took 3-4 iterations to reach a reasonable level of quality. The main areas that needed improvement were (1) making the context description specific to pattern application and the variation of the process that the patterns deals with, as well as (2) the granularity of patterns, i.e. initially the patterns created were either too large, hence not being suitable for adjustments, or too atomic, addressing a very small problem.

The CDD process foresees that relevant patterns are selected in during capability design or that the Capability Navigation Application is able to suggest relevant patterns for adjustment as part of monitoring the execution context at runtime. At this stage of the project, there were no patterns available at the time when the capability design begun. Even if some use case partners had some reusable process models, and other components that they were reusing in their business applications, consequently, the current set of patterns is created on the basis on the initial business processes developed from scratch for the purpose of capability design. Once the pattern repository becomes more mature and contains a significant amount of patterns for a specific application sector the capability design will be mostly done by selecting and configuring existing patterns.

5 Conclusions and Future Work

In this study we have discussed the use of patterns for supporting Capability Driven Development of context based business solutions as part of an ongoing EU research project CaaS. The pattern concept is used to support designing capability delivery from reusable model fragments. More specifically, we have presented the format and meta-model used to represent patterns and the way of working with patterns for developing capability designs. We have also reported initial experiences of developing patterns as part of the industrial use case work of the project.

The current experiences show the potential of using patterns for capturing reusable solutions in order to support capability design, as well as to use patterns for adjustment of capability delivery at run time. Amongst future work issues are, making the patterns integrated with the Capability Navigation Application for monitoring and suggesting adjustment, integrating patterns with context modeling and monitoring in order to support finding of relevant patterns, as well as establishing organizational processes for pattern capture and feedback at the industrial use case companies.

Acknowledgments. This work has been performed as part of the EU-FP7 funded project no: 611351 CaaS − Capability as a Service in Digital Enterprises.

References

1. EU FP7 CaaS Project. Capability as a service for digital enterprises. http://caas-project.eu/
2. Ulrich, W. and Rosen, M. The business capability map: Building a foundation for business/it alignment. Cutter Consortium for Business and Enterprise Architecture (2012)
3. OPENGROUP TOGAF - enterprise architecture methodology, version 9.1, http://www.opengroup.org/togaf/ (2012)
4. OPENGROUP Archimate - modelling language for enterprise architecture, v2.0, https://www2.opengroup.org/ogsys/catalog/c118 (2012)
5. Stirna, J., Zdravkovic J., Interview with Sladjan Maras on “Challenges and Needs in Enterprise Modeling”, Business & Information Systems Engineering, February 2015, Volume 57, Issue 1, pp 79-81
6. Stirna, J., Grabis, J., Henkel, M., and Zdravkovic, J.: Capability Driven Development – an Approach to Support Evolving Organizations. In: proc. of IFIP WG8.1 Working Conf. on the Practice of Enterprise Modeling (PoEM), Springer LNBIP, 117-131 (2012)
7. Zdravkovic, J., Stirna, J., Henkel, M., and Grabis, J. (2013) Modelling Business Capabilities and Context Dependent Delivery by Cloud Service. In Proc. of CAiSE’2013, LNCS 7908, 369-383, Springer (2013)
8. Alexander C., (1977). A pattern language, Oxford University Press, New York.
9. Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995).Design Patterns: Elements of Reusable Object-Oriented Software Architecture. Addison Wesley, Reading, MA
10. Fowler, M. (1997), Analysis patterns: Reusable object models, The Addison-Wesley series in object-oriented software engineering, Addison-Wesley, Menlo Park, Calif.
11. Rolland, C., Stirna, J., Prekas, N., Loucopoulos, P., Persson, A. and Grosz, G. (2000), “Evaluating a Pattern Approach as an Aid for the Development of Organisational

Knowledge: An Empirical Study", in Wangler, B. and Bergman, L. (Eds.), Advanced Information Systems Engineering, Lecture Notes in Computer Science, Vol. 1789, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 176–191.

12. Bērziša S., Bravos G., Gonzalez T., Czubayko U., España S., Grabis J., Henkel M., Jokste L., Kampars J., Koç H., Kuhr J., Llorca C., Loucopoulos P., Juanes R., Pastor O., Sandkuhl K., Simic H., Stirna J., Valverde F., Zdravkovic J. Capability Driven Development: An Approach to Designing Digital Enterprises, Business & Information Systems Engineering, February 2015, Volume 57, Issue 1, pp 15-25