Capability Driven Development – An Approach to Support Evolving Organizations

Janis Stirna, Jānis Grabis, Martin Henkel, Jelena Zdravkovic

Capability Driven Development – an Approach to Support Evolving Organizations

Janis Stirna[1], Jānis Grabis[2], Martin Henkel[1], Jelena Zdravkovic[1]

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

Abstract. The need for organizations to operate in changing environments is addressed by proposing an approach that integrates organizational development with information system (IS) development taking into account changes in the application context of the solution – Capability Driven Development (CDD). A meta-model for representing business and IS designs consisting of goals, key performance indicators, capabilities, context and capability delivery patterns, is been proposed. The use of the meta-model is exemplified by a case from the energy efficiency domain. A number of issues related to use of the CDD approach, namely, capability delivery application, CDD methodology, and tool support also are discussed.

Keywords. Enterprise modeling, capabilities, capability driven development, model driven development

1 Introduction

In order to improve alignment between business and information technology, information system (IS) developers continuously strive to increase the level of abstraction of development artifacts. A key focus area is making the IS designs more accessible to business stakeholders to articulate their business needs more efficiently. These developments include object-orientation, component based development, business process modeling, enterprise modeling (EM) and software services design. These techniques are mainly aimed at capturing relatively stable, core properties of business problems and on representing functional aspects of the IS [1]. However, the prevalence and volatility of the Internet shifts the problem solving focus to capturing instantaneous business opportunities [2] and increases the importance of non-functional aspects. Furthermore, the context of use for modern IS is not always predictable at the time of design; instead as IS should have the capability to support different contexts. Hence, we should consider the context of use and under which circumstances the IS, in congruence with the business system, can provide the needed business *capability*. Hence, system's capability is determined not only during the design-time but also at run-time when the system's ability to handle changes in

contexts is put to test. The following anecdotal evidence can be used to illustrate importance of capabilities. A small British bakery was growing successfully and decided to promote their business by offering their cupcakes at a discount via collective buying website, Groupon. As a result it had to bake 102 000 cupcakes and suffered losses comparable to its yearly profit. The bakery did not have mechanisms in place to manage the unforeseen and dramatic surge in demand - it did not have the capability of baking 102 000 cupcakes nor mechanisms for foreseeing the consequences. Another example is a mobile telecommunications company offering telephone services over its network, similar in all respects to traditional fixed-line providers. Such a service consists of the same home telephone, with an additional box between the telephone and the wall. However, unlike ordinary fixed-line telephony, it cannot connect to emergency services (112) in the event of a power outage. In this case the provided capability is unstable in a changing context.

A capability-driven approach to development should be able to elevate all such issues and to produce solutions that fit the actual application context.

From the business perspective, we define a *capability as being the ability to continuously deliver a certain business value in dynamically changing circumstances*. Software applications (and their execution environments) are an integral part of capabilities. This means that it is important to tailor these applications with regard to functionality, usability, reliability and other factors required by users operating in varying contexts. That puts pressure on software development and delivery methods. The software development industry has responded by elaborating Model Driven Development (MDD) methods and by adopting standardized design and delivery approaches such as service-oriented architecture and cloud computing. However, there are a number of major challenges when it comes to making use of MDD to address business capabilities:

- *The gap between business requirements and current MDD techniques*. Model driven approaches and tools still operate with artifacts defined on a relatively low abstraction level.
- *Inability to model execution contexts*. In complex and dynamically changing business environments, modeling just a service providing business functionality in very limited context of execution is not sufficient.
- *High cost for developing applications that work in different contexts*. Software developers, especially SMEs, have difficulties to market their software globally because of the effort it takes to adhere to localization requirements and constraints in the context of where the software will be used.
- *Limited support for modeling changes in non-functional requirements*. Model driven approaches focus on functional aspects at a given time point, rather than representing evolution of both functional and non-functional system requirements over time.
- *Limited support for "plasticity" in applications*. The current context-aware and front-end adaptation systems focus mainly on technical aspects (e.g., location awareness and using different devices) rather than on business context awareness.
- *Limited platform usage*. Limited modeling support for defining ability the IS to make use of new platforms, such as cloud computing platforms. Cloud computing is a technology driven phenomenon, and there is little guidance for development of cloud based business applications.

We propose to support the development of capabilities by using EM techniques as a starting point of the development process, and to use model-based patterns to describe how the software application can adhere to changes in the execution context. Our vision is to apply enterprise models representing enterprise capabilities to create executable software with built-in contextualization patterns thus leading to *Capability Driven Development (CDD)*.

The objective of this paper is to present the capability meta-model, to discuss its feasibility by using an example case, and to outline a number of open development issues related to practical adoption of the CDD approach.

The research approach taken in this paper is conceptual and argumentative. Concepts used in enterprise modeling, context representation and service specification are combined together to establish the capability meta-model. Preliminary validation and demonstration of the CDD approach is performed using an example of designing a decision support system for optimizing energy flows in a building. Application of the meta-model is outlined by analyzing its role in development of capability delivery applications. The CDD methodology is proposed following the principles of agile, iterative and real-time software development methodologies.

The remainder of the paper is organized as follows. Section 2 presents related work. In section 3 requirements for CDD are discussed. Section 4 presents the CDD meta-model. It is applied to an example case in section 5. Section 6 discusses aspects of development methodology need for the CDD approach. The paper ends with some concluding remarks in section 7.

2 Related Work

In the strategic management discipline, a company's resources and capabilities are long-time seen as the primary source of profitability and competitive advantage – [3] has united them into what has become known as the resource-based view of the company. Accordingly, Michael Porter's value chain identifies top-level activities with the capabilities needed to accomplish them [4]. In Strategy Maps and Balanced Scorecards, Kaplan and Norton also analyze capabilities through the company's perspectives, e.g. financial, customers', and other [5]. Following this, in the research within Business-IT alignment, there have been attempts to consider resources and capabilities as the core components in enterprise models, more specifically, in business value models [6, 7]. However, in none of these works, capabilities are formally linked to IS models. In the SOA reference architecture [8] capability has been described as a business functionality that, through a service, delivers a well-defined user need. However, in the specification, not much attention is given to the modeling of capability, nor it is linked to software services. In the Web Service research, capability is considered purely on the technical level, through service level agreements and policy specifications [9].

In order to reduce development time, to improve software quality, and to increase development flexibility, MDD has established itself as one of the most promising software development approaches. However, [10] show that the widely practiced MDD specialization - Model Driven Architecture [11] and following methodologies,

mainly assume requirements as given a priori. [12] and [13] indicate that MDA starts with system analysis's models. They also survey various methods for integrating requirements into an overall model-driven framework, but do not address the issue of requirements origination. There is a limited evidence of MDA providing the promised benefits [14]. Complexity of tools, their methodological weaknesses, and too low abstraction level of development artifacts are among the main areas of improvement for MDD tools [15].

Business modeling and Enterprise Modeling (EM) [16] has been used for business development and early requirements elicitation for many years, but a smooth (nearly automated) transition to software development has not been achieved due to immaturity of the existing approaches and lack of tools. Enterprise-wide models are also found in [17], where the enterprise architecture of ArchiMate is extended with an intentional aspect capturing the goals and requirements for creating an enterprise system. A comparable solution is developed in [18], where a generic process is presented for linking i* and the OO-Method as two representatives of Goal-Oriented Requirements Engineering (GORE) and MDD, respectively. In [19] a recent analysis of the current state in this area is presented, as well as proposed a meta-model for integrating EM with MDD.

Model driven approaches also show promise to development of cloud-based applications, which has been extensively discussed at the 1st International Conference on Cloud Computing and Service Sciences, c.f. [20, 21]. However, these investigations currently are at the conceptual level and are aimed at demonstrating a potential of MDD for cloud computing. A number of European research project, e.g. REMICS and SLA@SOI have been defined in this area.

Methods for capturing context in applications and services have achieved high level of maturity and they provide a basis for application of context information in software development and execution. [22] describe MDD for context-aware applications, where the context model is bound to a business model, encompassing information about user's location, time, profile, etc. Context awareness has been extensively explored for Web Services, both methods and architectures, as reported in [23]. It is also studied in relation to workflow adaptation [24]. Lately, [25] has suggested a formal context model, compounded by ontologies describing users, devices, environment and services. In [26] an extension to State charts to capture context dependent variability in processes has been proposed.

Non-functional aspects of service-oriented applications are controlled using QoS data and SLA. Dynamic binding and service selection methods allow replacing under-performing services in run-time [27]. However, QoS and SLA focus only on a limited number of technical performance criteria with little regard to business value of these criteria.

In summary, there are a number or contributions in addressing the problem of adjusting the IS depending on the context, however business capability concept is not explicitly addressed in the context development.

3 Requirements for Capability Driven Development

In this section we discuss a number of requirements motivating the need for CDD.

Currently the business situation in which the IS will be used is predetermined at design time. At run-time, only adaptations that are within the scope of the planned situation can usually be made. But in the emerging business contexts we need rapid response to changes in the business context and development of new capabilities, which also requires run-time configuration and adjustment of applications. In this respect a capability modeling meta-model linking business designs with application contexts and IS components is needed.

Designing capabilities is a task that combines both business and IS knowledge. Hence both domains need to be integrated in such a way that allows establishing IS support for the business capabilities.

Current EM and business development approaches have grown from the principle that a single business model is owned by a single company. In spite of distributed value chains and virtual organizations [28] this way of designing organizations and their IS still prevails. The CDD approach would aim to support co-development and co-existence of several business models by providing "connection points" between business models based on goals and business capabilities.

Most of the current MDD approaches are only efficient at generating relatively simple data processing applications (e.g. form-driven). They do not support e.g. complex calculations, advanced user interfaces, scalability of the application in the cloud. CDD should bring the state of the art further by supporting the modeling of the application execution context; this includes the ability to model the ability to switch service providers and platforms. Furthermore, the capability approach would also allow deploying more adequate security measures, by designing overall security approaches at design-time and then customizing them at deployment and run-time.

4 Foundation for Capability Driven Development

The capability meta-model presented in this section provides the theoretical and methodological foundation for the CDD. The meta-model is developed on the basis of industrial requirements and related research on capabilities. Initial version of such a meta-model is given in Figure 1. The meta-model has three main sections:

- *Enterprise and capability modeling*. This focuses on developing organizational designs that can be configured according to the context dependent capabilities in which they will be used. I.e. this captures a set of generic solutions applicable in many different business situations.
- *Capability delivery context modeling*. Represents the situational context under which the solutions should be applied including indicators for measuring the context properties.
- *Capability delivery* patterns representing reusable solutions for reaching business goals under different situational contexts. The context defined for the capability should match the context in which the pattern is applicable in.

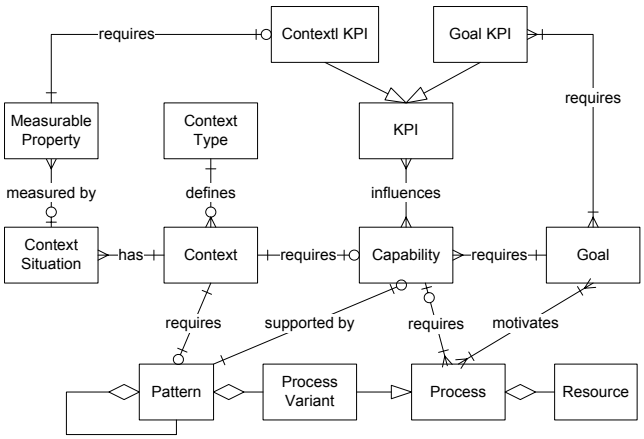

Fig. 1. The initial capability meta-model

4.1 Enterprise and Capability Modeling

This part covers modeling of business goals, key performance indicators (KPI), and business processes needed to accomplish the goals. We also specify resources required to perform processes. The associations between these modeling components are based on the meta-model of EM approach EKD [29]. The concept of capability extends this meta-model towards being suitable for CDD.

Capability expresses an ability to reach a certain business objective within the range of certain contexts by applying a certain solution. Capability essentially links together business goals with patterns by providing contexts in which certain patterns (i.e. business solutions) should be applicable.

Each capability supports or is motivated by one business goal. In principle business goals can be seen as internal means for designing and managing the organization and capabilities as offerings to external customers. A capability requires or is supported by specific business processes, provided by specific roles, as well as it needs certain resources and IS components. The distinguishing characteristic of the capability is that it is designed to be provided in a specific context. The desired goal fulfillment levels can be defined by using a set of goal fulfillment indicators – Goal KPIs.

4.2 Context Modeling

The context is any information that can be used to characterize the situation, in which the capability can be provided. It describes circumstances, i.e. context situation, such as geographical location, platforms and devices used and as well as business conditions and environment. These circumstances are defined by different context types. The context situation represents the current context status. Each capability delivery pattern is valid for a specific set of context situations as defined by the pattern validity space. The context KPIs are associated with a specific capability delivery pattern. They represent context measurements, which are of vital importance for the capability delivery. The context KPI are used to monitor whether the pattern chosen for capability delivery is still valid for the current context situation. If the

pattern is not valid, then capability delivery should be dynamically adjusted by applying a different pattern or reconfiguring the existing pattern (i.e., changing delivery process, reassigning resources etc.). Technically, the context information is captured using a context platform in a standardized format (e.g. XCoA). Context values change according to a situation. The context determines how a capability is delivered, which is represented by a pattern.

4.3 Capability Delivery Pattern

A pattern is used to: "describe a problem that 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" [30]. This principle of describing a reusable solution to a recurrent problem in a given context has been adopted in various domains such as software engineering, information system analysis and design [31] as well as organizational design. Organizational patterns have proven to be a useful way for the purpose of documenting, representing, and sharing best practices in various domains (c.f. [32]).

In the CDD approach we amalgamate the principle of reuse and execution of software patterns with the principle of sharing best practices of organizational patterns. Hence, *capability delivery patterns are generic and abstract design proposals that can be easily adapted, reused, and executed*. Patterns will represent reusable solutions in terms of business process, resources, roles and supporting IT components (e.g. code fragments, web service definitions) for delivering a specific type of capability in a given context. In this regard the capability delivery patterns extend the work on task patterns performed in the MAPPER project [33].

Each pattern describes how a certain capability is to be met within a certain context and what resources, process, roles and IS components are needed. In order to provide a fit between required resources and available resources, KPIs for monitoring capability delivery quality are defined in accordance with organization's goals. KPIs measure whether currently available resources are sufficient in the current context. In order to resolve resource availability conflicts, conflict resolutions rules are provided.

5 Example Case

To exemplify the proposed approach we model a case of a building operator aiming to run its buildings efficiently and in an environmentally sustainable manner. The case is inspired by the FP7 project EnRiMa – "Energy Efficiency and Risk Management in Public Buildings" (proj. no. 260041). The objective of the EnRiMa project is to develop a decision support system (DSS) for optimizing energy flows in a building. In this paper we envision how this service will be used after the DSS will be operational. The challenge that the capability driven approach should address is the need to operate different buildings (e.g. new, old, carbon neutral) in different market conditions (e.g. fixed energy prices, flexible prices), different energy technologies (e.g. energy storage, photovoltaic (PV)), and with different ICT technologies (e.g. smart sensors, advanced ICT infrastructure, closed ICT infrastructure, remote

monitoring, no substantial ICT support). The EnRiMa DSS aims to provide building specific optimization by using customized energy models describing the energy flows for each building. The optimization can be based on using building data from the on-site buildings management systems, for example giving the current temperature and ventilation air flow. The project also aims to provide a DSS that can be installed on-site or via deployment in the cloud.

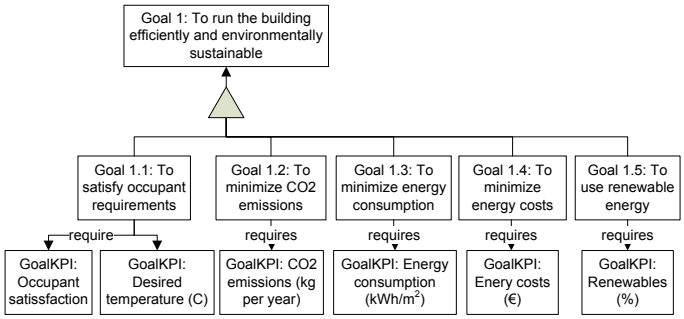

Fig. 2. A generic goal model for a building operator

5.1 Enterprise Modeling

The top goal is refined into a number of sub-goals, each lined to one or several KPIs. This is a simplification; in real life there are more sub-goals and KPIs to consider than figure 2 shows. In this particular case the decomposition of the top goal into the five sub-goals should be seen in conjunction with the KPIs. I.e. the building operator wants to achieve all of the sub-goals, but since that is not possible for each particular building the owner defines specific KPIs to be used for the optimization tasks.

In summary, KPIs are used for designing the capabilities to set the level of goal fulfillment that is being expected from the capabilities. In the capability driven approach presented here we use indicators to define different level of goal fulfillment that we can expect.

Processes are central for coordinating the resources that are needed for a capability. In this case there are processes that are executed once e.g. for the initial configuration of the system and then-re executed when the context changes. We here include four basic processes:

Energy audit and configuration process. As a part of improving the energy efficiency of a building there is a need to perform an energy audit and to configure the decision support system with general information on building. The energy audit will result in a model of the building energy flows, for example to determine how much of the electricity that goes to heating, and to determine the technical equipment (such as boilers) efficiency level. Besides the energy flow there is also a need to configure the system with information of the glass area of the building, hours of operation and so on. Depending on the desired capability the process can take a number of variants, ranging from simple estimation to full-scale audits. Note that if the context changes, for example if the installed energy technology in the building changes, there is a need

to repeat the configuration. We here define two variant of this process: Template based – using generic building data to estimate energy flows, Full energy audit – doing a complete energy flow analysis, leading to a detailed model of the building

ICT infrastructure integration process. To continuously optimize the energy efficiency of a building there is a need to monitor the building behavior via its installed building management system. For example, by monitoring the temperature changes the cooling system can be optimized to not compensate for small temperature fluctuations. This process can take several variants, depending on the context in the form of the building management system ability to integrate with external systems. In this case we define two variants: Manual data entry – data entered manually, Integration – data fetched directly from the building management system. The actual integration process depends on which building management system is installed (e.g. Siemens Desigo system).

Deployment process. Depending on the access needs the decision support system can be executed at the building site, at a remote locations, or in a cloud platform provided by an external provider. Process variants: On-site, External, Cloud provider.

Energy efficiency monitoring and optimization process. This process is at the core of delivering the capability, i.e. monitoring, analyzing and optimizing the energy flows is what can lead to a lower the energy consumption. A very basic variant, addressing a simple context is to just monitor for failures in one of the building systems. A more advanced variant, catering to highly automated buildings is to perform a daily, automated, analysis to change the behavior of the installed building technologies. Process variants: Passive monitoring – monitoring for failures, Active optimization – performing pro-active optimizations based on detailed estimations

Depending on the context the variants of these processes can be activated, this will be described in the next section.

5.2 Context Modeling

The DSS can be deployed to a wide range of contexts. To exemplify the varying conditions we here describe two simplified context types:

Older building, low ICT Monitoring – where the building got a low degree of ICT integration abilities, and the overall desire of the building owner is to monitor the buildings energy usage and minimize costs.

Modern building, high ICT infrastructure – where integration with the building system is possible, a building model allowing continuous optimizations is possible, and the building owner wants to balance CO_2 emissions and cost minimization.

Each of these context types can be addressed by capabilities (see figures 3 and 4) that guide through selecting the right processes or process variants; this will be further described in the section on patterns. The examples here present the enterprise models at design-time. To detect a context change at runtime we define a set of context-KPIs. These allow us to monitor the goal fulfillment at runtime by comparing the measurable situational properties. For example, Context KPI: Energy consumption 200 kWh/m^2 should be compared with the actual energy consumption (see figure 3).

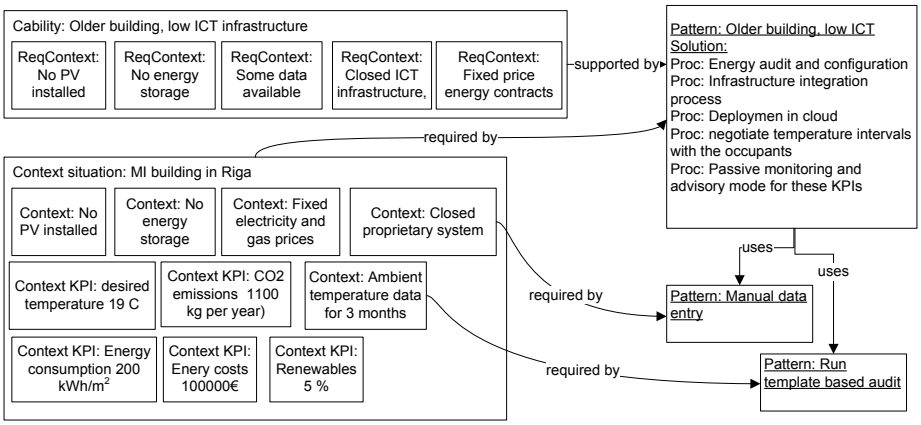

Fig. 3. Capability, context and capability delivery pattern for "Older building with low ICT infrastructure"

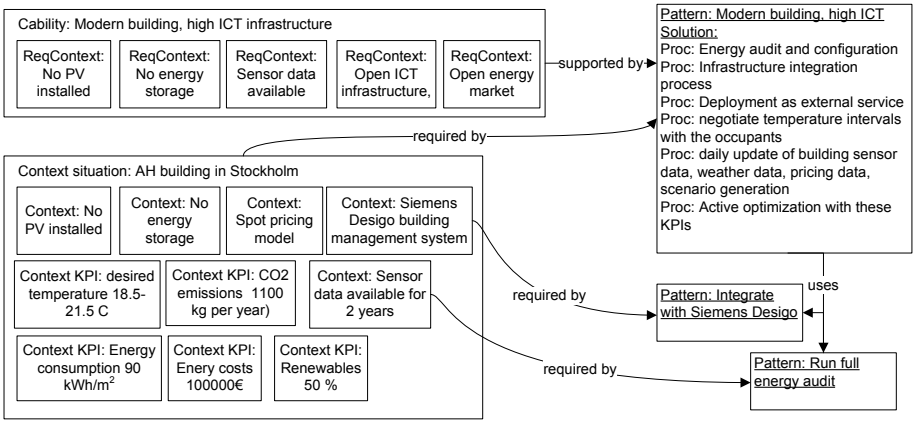

Fig. 4. Capability, context and capability delivery pattern for "Modern building with high ICT infrastructure."

The patterns shown here omit details such as forces and usage guidelines, e.g. explaining how to apply and use the processes and/or executable services. In a real life case they should be developed and included in the pattern body.

5.3 Capability Delivery Patterns

The EnRiMa DSS will be used to balance various, often contradictory, operator goals, e.g. to lower the energy costs in buildings and to reduce CO_2 emissions. Each building however is different, and thus the context of execution for the system will vary. Therefore we design a set of process variants. The role of capability delivery patterns is to capture and represent which process variants should be used at which contexts delivering which capabilities. For example, if a building has Siemens Desigo building management system, then a pattern describing how to integrate it with the EnRiMa

DSS and which executable components (e.g. web-services) should be used. If the building has closed system, then manual data input should be used instead. Table 1 shows two capabilities and their relation to variants of the energy audit and integration with the existing ICT systems of the building. Moreover we identify those context KPIs that can be of use when monitoring the process execution.

Table 1. Example of two context patterns, each making use of process variants.

Capability delivery pattern contains:	*Capability: Old building, low ICT*	*Capability: Modern building, high ICT*
ICT infrastructure integration process	Pattern: Manual data entry	Pattern: Integrate with Siemens Desigo
Energy audit and configuration	Pattern: Template based audit	Pattern: Run full energy audit

6 Discussion

In this section we will discuss issues pertinent to usage of CDD, namely capability delivery application (CDA), CDD methodology, and tool support.

6.1 Capability Delivery Application

A company requesting a particular capability represents it using the concepts of CDD meta-model. The main principle of CDD is that, in comparison to traditional development methods, the software design part is supported by improving both the analysis side and the implementation side. From the analysis side, the capability representation is enriched and architectural decisions are simplified by using patterns. From the implementation side, the detailed design complexity is reduced by relying on, for example, traditional web-services or cloud-based services. The resulting CDA is a composite application based on external services.

Figure 5 shows three conceptual layers of the CDA: (1) Enterprise Modeling layer; (2) design layer; and (3) execution layer. The EM layer is responsible for high level of representation of required capabilities. The design layer is responsible for composing meta-capabilities from capability patterns, which is achieved by coupling patterns with executable services. The execution layer is responsible for execution of the capability delivery application and its adjustment to the changing context.

The requested capability is modeled using the EM techniques and according to the capability meta-model as described in this paper. The patterns are analyzed in order to identify atomic capabilities that can be delivered by internal or external services by using a set of service selection methods. These service selection methods are based on existing service selection methods [34]. Availability of internal services is identified by matching the capability definition against the enterprise architecture, and a set of the matching rules will have to be elaborated.

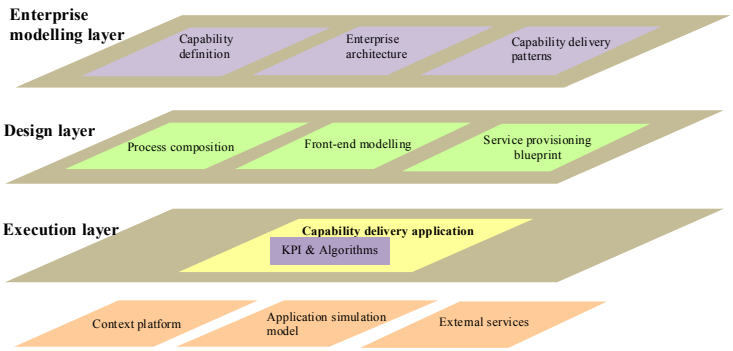

Fig. 5. Layered view of capability delivery application

A process composition language is used to orchestrate services selected for delivering the requested capability. The process composition model includes multiple process execution variants [35]. The capabilities are delivered with different front-ends, which are modelled using an extended user interface modelling language. The external services used in CDA should be able to deliver the requested performance in the defined context. The necessary service functionality and non-functional requirements corresponding to the context definition are transformed into a service provisioning blueprint [36], which is used as a starting point for binding capability delivery models with executable components and their deployment environment. The service provisioning blueprint also includes KPIs to be used for monitoring the capability delivery. We envision that the CDA is deployed together with its simulation model and run-time adjustment algorithms based on goal and context KPIs. The key task of these algorithms is enacting of the appropriate process execution variant in response to the context change.

Business capabilities also could be delivered using traditional service-oriented and composite applications. However, the envisioned CDA better suites the requirements of CDD by providing integration with enterprise models and built-in algorithms for dynamic application adjustment in response to changing execution context.

6.2 The Process of Capability Driven Development

To support development of CDA, a CDD methodology is needed. It is based on agile and model driven IS development principles and consists of the CDD development process, a language for representing capabilities according to the CDD meta-model, as well as modeling tools. The main principles of the CDD methodology should be:

- Use of enterprise models understandable to business stakeholders,
- Support for heterogeneous development environment as opposed to a single vendor platform,
- Equal importance of both design-time and run-time activities with clear focus on different development artifacts,
- Rapid development of applications specific to a business challenge,
- Search for the most economically and technically advantageous solution,

An overview of the envisioned CDD process is shown in Figure 6. It includes three main capability delivery cycles: 1) development of the capability delivery application; 2) execution of the capability delivery application; and 3) capability refinement and pattern updating. These three cycles address the core requirements of the CDD by starting development with enterprise level organizational and IS models, adjustment of the capability delivery during the application run-time and establishing and updating capability delivery patterns.

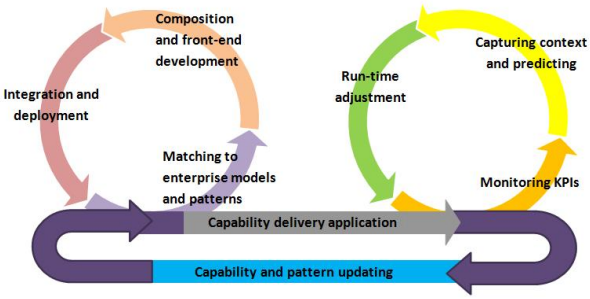

Fig. 6. Capability Driven Development methodology

CDD should also encompass run-time adjustment algorithms because the capability is delivered in a changing context, where both business (e.g., current business situation (growth, decline), priorities, personnel availability) and technical (e.g., location, device, workload) matters. Once the CDA is developed and deployed, it is continuously monitored and adjusted according to the changing context. Monitoring is performed using KPIs included in the system during the development and adjustment is made using algorithms provided by the CDD methodology.

Tool support also is important for CDD. EM is a part of CDD and for this purpose a modeling tool is needed. It should mainly address the design phase because at runtime tools provided by the target platform will be used.

We are currently planning to develop an open source Eclipse based tool for CDD and will use Eclipse EMF plug-in and other relevant plug-ins as the development foundation. Models are built on the basis of extensions of modeling languages such as EKD, UML and executable BPMN 2.0.

7 Concluding Remarks and Future Work

We have proposed an approach that integrates organizational development with IS development taking into account changes in the application context of the solution – Capability Driven Development. We have presented a meta-model for representing business designs and exemplified it by a case from the energy efficiency domain. This in essence is research in progress, and hence, we have also discussed a number of issues for future work related to use of the CDD approach, namely, capability delivery application, CDD methodology, and tool support also are discussed.

The two important challenges to be addressed are availability of patterns and implementation of algorithms for dynamic adjustment of CDA. In order to ensure pattern availability an infrastructure and methods for life-cycle management of

patterns is required. In some cases, incentives for sharing patterns among companies can be devised. That is particularly promising in the field of energy efficiency. There could be a large number of different adjustment algorithms. Elaboration and implementation should follow a set of general, open principles for incorporating algorithms developed by third parties.

The main future directions are throughout validation of the capabilities meta-model and formulation of rules for matching required capabilities to existing or envisioned enterprise resources represented in a form of enterprise models and architectures.

References

1. Wesenberg, H. (2011) Enterprise Modeling in an Agile World, P. Johannesson, J. Krogstie, A. L. Opdahl (Eds.): in proc, of PoEM 2011, Springer LNBIP 92, p.126-130
2. Deloitte (2009) Cloud Computing: Forecasting Change, Deloitte Consulting, https://www.deloitte.com/assets/Dcom-Global/Local%20Assets/Documents/TMT/cloud_-_market_overview_and_perspective.pdf
3. Barney J.B, (1991) Firm Resources and Sustained Competitive Advantage. Journal of Management, 17(1), 99–120.
4. Porter, M.E.: Competitive Advantage: Creating and Sustaining Superior Performance, Free Press, New York (1985)
5. Kaplan, R.S., and Norton, D.P.: Strategy Maps: Converting Intangible Assets into Tangible Outcomes. Harvard Business School Press, Boston, MA (2004)
6. Osterwlader, A., Pigneur, Y. (2003) Modeling value propositions in e-Business. Proceedings of the 5th International Conference on Electronic Commerce, ICEC 2003. ACM Conference Proceedings Series 50, 2003, ISBN 1-58113-788-5
7. Kinderen, de, S., Gordijn, J., Akkermans, H. (2009) Reasoning about customer needs in multi-supplier ICT service bundles using decision models. In Proceedings of the 11th International Conference on Enterprise Information Systems, ICEIS 2009, 131-136
8. OASIS, (2011) Reference Architecture Foundation for Service Oriented Architecture Version 1.0, Committee Specification Draft 03 / Public Review Draft 02 06 July 2011, http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra.pdf
9. Papazoglou, M. P. and Yang, J. "Design Methodology for Web Services and Business Processes", Proc. Third International Workshop on Technologies for E-Services (TES 03), LNCS, Vol. 2444, Springer-Verlag 2003, pp 54-64.
10. Asadi, M. and Ramsin, R (2008), MDA-Based Methodologies: An Analytical Survey, I. Schieferdecker and A. Hartman (Eds.): ECMDA-FA 2008, LNCS 5095, 419–431.
11. Kleppe, A., Warmer, J. and Bast, W., MDA Explained. Addison-Wesley Professional, 2003.
12. Loniewski, G., Insfran, E. and Abrahao, L. (2010), A Systematic Review of the Use of Requirements Engineering Techniques in Model-Driven Development, D.C. Petriu, N. Rouquette,Ø. Haugen (Eds.):MODELS 2010, Part II, LNCS 6395, pp. 213–227.
13. Yue, T., Briand, L.C. and Labiche, Y. (2011) A systematic review of transformation approaches between user requirements and analysis models, Requirements Engineering 16, 75–99
14. Mohagheghi, P. and Dehlen, V. (2008) Where Is the Proof? - A Review of Experiences from Applying MDE in Industry, I. Schieferdecker and A. Hartman (Eds.): ECMDA-FA 2008, LNCS 5095, 432–443.
15. Henkel, M., Stirna J., (2010) Pondering on the Key Functionality of Model Driven Development Tools: the Case of Mendix, P. Forbrig and H. Günther (eds.), in proc. of Business Informatics Research (BIR 2010), Springer LNBIP 64, ISBN 978-3-642-16100-1

16. Nilsson, A. G., Tolis, C. and Nellborn, C. (Eds.) (1999), Perspectives on Business Modelling: Understanding and Changing Organisations, Springer-Verlag
17. ArchiMate, An enterprise modeling language from the Open Group. http://www.opengroup.org/archimate/
18. Pastor, O., & Giachetti, G. (2010). Linking Goal-Oriented Requirements and Model-Driven Development. In Intentional Perspectives on Information Systems Engineering (pp. 257–276). Springer
19. Zikra, I., Stirna, J., Zdravkovic, J. (2011) Bringing Enterprise Modeling Closer to Model-Driven Development. In Proceedings of 4th Working Conference on Practice of Enterprise Modeling, PoEM 2011, Springer LNBIP 92, 268-282, , ISBN 978-3-642-24848-1
20. Esparza-Peidro, J., Munoz-Escoí, F.D. (2011) Towards the next generation of model driven cloud platforms, CLOSER 2011 - Proceedings of the 1st International Conference on Cloud Computing and Services Science, 494-500
21. Hamdaqa, M., Livogiannis, T., Tahvildari, L. (2011) A reference model for developing cloud applications, CLOSER 2011 - Proceedings of the 1st International Conference on Cloud Computing and Services Science, 98-103
22. Vale, S., Hammoudi, S. (2009) COMODE: A framework for the development of context-aware applications in the context of MDE, Proceedings of the 2009 4th International Conference on Internet and Web Applications and Services, ICIW 2009, 261-266
23. Sheng., Q., Yu, J., Dustar, S., Eds. (2010) Enabling Context-Aware Web Services: Methods, Architectures, and Technologies. Chapman and Hall/CRC, ISBN 1-43980-985-2
24. Smanchat, S., Ling, S., Indrawan, M. (2008) A survey on context-aware workflow adaptations, MoMM2008 - The 6th International Conference on Advances in Mobile Computing and Multimedia, 414-417
25. Hervas, R., Bravo, J. and Fontecha, J. (2010) A Context Model based on Ontological Languages; a proposal for Information Visualisation. Journal of Universal Computer Science (J.UCS) Vol. 16/12
26. Liptchinsky, V., Khazanlin R., Truong H.-L., Dustdar S., (2012) A Novel Approach to Modeling Context-Aware and Social Collaboration Processes, J. Ralyté, X.Franch, S. Brinkkemper, S. Wrycza (eds.), in proc. of CAiSE 2012, Springer LNCS 7328
27. Comuzzi, M., Pernici, B. (2009), A framework for QoS-based Web service contracting, ACM Transactions on the Web 3 (3), 10-52
28. Davidow, W.H., and M.S. Malone. (1992) The Virtual Corporation: Structuring and Revitalizing the Corporation for the 21st Century, Harper Collins Publishers
29. Bubenko, J. A. Jr., Persson, A. and Stirna, J. (2001). User Guide of the Knowledge Management Approach Using Enterprise Knowledge Patterns. Deliverable D3, IST Programme project Hypermedia and Pattern Based Knowledge Management for Smart Organisations, project no. IST-2000-28401, Royal Institute of Technology, Sweden.
30. Alexander C., (1977). A pattern language, Oxford University Press, New York.
31. Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) Design Patterns: Elements of Reusable Object-Oriented Software Architecture. Addison Wesley
32. Niwe, M. Stirna, J. (2009) Organizational Patterns for B2B Environments-Validation and Comparison, Halpin et al. (eds.) in Enterprise Business-Process and Information Systems Modeling, Springer LNBIP 29, ISBN 978-3-642-01861-9
33. Sandkuhl K., Stirna J., (2008) Evaluation of Task Pattern Use in Web-based Collaborative Engineering, Proc. of the 34th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), EUROMICRO, IEEE, ISBN 978-0-7695-3276-9
34. Chen, L., Song, Z.-L., Zhang, Y., Miao, Z. (2011), A method for context-aware web services selection, International Journal of Advancements in Computing Technology 3 (7), 291-298
35. Lu, R., Sadiq, S., Governatori, G. (2009) On managing business processes variants, Data & Knowledge Engineering 68, 642–664.
36. Nguyen, D.K., F. Lelli, Y. Taher, M. Parkin, M.P. Papazoglou, and W. van den Heuvel (2011) Blueprint Template Support for Engineering Cloud-Based Services, W. Abramowicz et al. (Eds.): ServiceWave 2011, LNCS 6994, 26–37.