Modeling Business Capabilities and Context Dependent Delivery by Cloud Services

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

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

Abstract. Contemporary business environments are changing rapidly, organizations are global, and cloud-based services have become a norm. Enterprises operating in these conditions need to have the capability to deliver their business in a variety of business contexts. Capability delivery thus has to be monitored and adjusted. Current Enterprise Modeling approaches do not address context-dependent capability design and do not explicitly support runtime adjustments. To address this challenge, a capability-driven approach is proposed to model business capabilities by using EM techniques, and to use model-based patterns to describe how software applications can adhere to changes in the execution context. A meta-model for capability design and delivery is presented with the consideration to delivering solutions as cloud services. The proposal is illustrated with an example case from an energy efficiency project. A supporting architecture for the capability development and the delivery in the cloud is also presented.

Keywords: Model-Driven Development, Capability, Context, Cloud Computing, Enterprise Modeling.

1 Introduction

Together with resources, company's capabilities have long been recognized as the primary sources of business profitability and competitive advantage [1]. In Enterprise Modeling (EM), business capability is the notion commonly used to describe the essential functions of an enterprise [2], [3]. Capabilities are then mapped to IT solutions, such as software services [4], [5] to deliver them to final customers. There is however an emerging challenge – in modern business environments that are global and Internet based, business capability delivery needs to be based on the application context. The goal of capability modeling is thus to classify functional abilities, to identify relevant contexts, and to align technology with business.

Furthermore, the dominance and volatility of the Internet shifts the problem solving focus to capturing instantaneous business opportunities, which increases the importance of non-functional aspects such as availability and scalability. The fact that

C. Salinesi, M.C. Norrie, and O. Pastor (Eds.): CAiSE 2013, LNCS 7908, pp. 369–383, 2013.
© Springer-Verlag Berlin Heidelberg 2013

the context of use for modern systems is not always predictable at the time of design is a further challenge resulting in the need for modern IS should have the capability to support different contexts. For example, airport operations use different patterns to cope with different levels of passenger flow at times of different events. Currently these patterns do not fully extend to customizing airport's IT services depending on a context. As a result, if many passengers are stranded at once due to bad weather, strike or accident, the IT services cannot cope with the surge in demand, which leads to webpages being unavailable, servers overloaded, networks congested, and long waiting times in telephone services.

In such situations, not even new scalable service delivery platforms, such as cloud computing, are adequately supportive. This is because cloud computing is a technology driven phenomenon, and there is little guidance for the development of cloud-based business applications [6]. To manage changing amounts of work, scaling of servers and data storages is not sufficient under conditions when the cloud service is to cope with changed business needs requiring new business processes and services to be engaged.

A capability-driven approach to business and IT development should be able to elevate such issues and to produce solutions that are fit for changing business contexts, while taking the advantage of emerging technology solutions. The objective of this paper is to present a proposal to model business capabilities by using EM techniques as a starting point for the development process, and to use model-based contextualized patterns to enable cloud services to adhere to changes in the execution context. Our vision is to apply enterprise models emphasizing business capabilities to create executable software with built-in contextualization patterns.

The research approach taken in this paper is conceptual and argumentative. Concepts used in EM, context representation and service specification are combined together to establish the design aspect of the capability meta-model, where its delivery aspect relies on the cloud computing components. Preliminary validation and demonstration of the capability modeling approach is performed using an example of designing a decision support system for optimizing energy flows in buildings.

The paper is organized as follows: Section 2 gives brief overviews of the prevalent research done on modeling business capabilities, context modeling, and cloud computing. Section 3 presents the capability meta-model, including design and delivery aspects. In Section 4, the meta-model is exemplified, using a business case. A brief guide to the methodology for development and implementation of the capability-based model is given in Section 5. Section 6 concludes the paper with a reflection on the results and future research directions.

2 Related Work

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

2.1 Capability Modeling

In a business context, *capability* refers to the resources and expertise that an enterprise needs to offer its functions. The notion of capability emerged in the beginning of the nineties in the context of developing firm's competitive advantage [7], [8], [9]. It was later adopted for Business-IT alignment [10]. The capability notion is also used in AI and in particular in Agent Oriented Programming (AOP) in order to describe an aspect of agent's state, i.e. what activities is the agent able to do at certain time [11].

Lately the notion of business capability has gained a growing attention, due to a number of factors: the notion directs business investment focus, it can be used as a baseline for business planning, and it leads directly to service specification and design [4]. More specifically, capability is used to describe what a business can do, but not how; technical terms are not used – instead, capabilities are mapped to IT deployments through IT architectures.

Following these briefly explained relations of capability to business and IT, the argument in [4] that the notion gets the most value when incorporated into a larger view of an enterprise's ecosystem becomes comprehensible. Thus, the notion has been over time captured by Enterprise Architecture to present the core of the business architecture. In TOGAF [2], for instance, the "architectural vision" describes high-level capabilities as meeting the business goals and the stakeholder concerns. Capability is defined and assessed on different levels, e.g. for the enterprise as a whole or individual segments such as architectural functions. In ArchiMate [3], capability is modeled through the "business function" entity of the business layer, grouping required skills and resources, and being used by one or more business processes. In SOA [5], 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.

Our understanding of capability coincides with the above described proposals; however, we go beyond them by proposing a meta-model showing how capability is related to the business design (i.e. goals and processes), its dependence on situational context, as well as mechanisms for the delivery through IT, which are not, or at least not explicitly, addressed in the discussed proposals.

2.2 Context Modeling

The notion of *context* refers to situational cognition; as such, it is used to fully describe the conditions of a situation. Schilt et al. [13] scope context with "where you are", "who you are with", and "what resources are nearby", whereas Pascoe [14] defines it as the subset of physical and conceptual states of interest to a particular entity. Subsequently arguing that the former definitions are too specific, Dey [15] defines context as "any information that can be used to characterize the situation of an entity". In computational frameworks, one of the first considerations of context-sensitivity is found in AOP, where the modalities of agent's state such as obligations or capabilities, are affected by a context, such as network being up [11], [12].

Many categorizations of context have been proposed for purposes such as a generic understanding and enumeration, as well as for its application in computing. In [16], Dey and Abowd distinguishes context types, such as, location, identity, activity and time, while Gross and Specht [17] defines the four dimensions of the context, namely,

location, identity, time, and environment. Arguing that context is a broad, inaccurate, and non-delimited concept, in [18] Hervas proposes a more refined classification. The proposal facilitates elicitation of a context by identifying users, environment, services and devices, further refined by what, who, where, when and why. E.g., a user determines a context, by eliciting not only the user itself, but, in addition, what he/she is doing, when he/she is doing, etc.

All above discussed context categorizations set the focus to an entity, or more specifically, to a user. In contrast, in our research, there is a need to model the context surrounding the delivery of a business. Thus, the presented categorizations have not been applicable. However, the two-dimensional context framework of Hervas has been considered as an inspiration when creating our meta-model.

2.3 Cloud Modeling

Relying on service-based sharing of resources such as storage, hardware and applications, cloud computing has facilitated coherence of the resources and economies of scale through its pay-per-use business model. From the customer's perspective the cloud technology offers a means to increase capacity or add capabilities on the fly, without investing in new infrastructure, training new personnel, or licensing new software.

Being conceptualized in such a way, the cloud technology endeavors provide two main features: virtualization and scalability on demand [6], [19]. Virtualization is achieved by offering various resources through a unified abstract interface to a number of users. This further requires for scalability - addition, or withdrawal of resources, according to demands, where the interface to users is constant. The scalability of resources further requires a smooth integration with offered applications to enable their transparent elasticity, i.e. the power according to the needs of users. Cloud services are offered as three basic models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Lately, delivering of business processes as cloud services has been also proposed [20].

According to [6], the main challenges to adoption and the growth of cloud computing are the obstacles to availability of services, security, and performance, lack of customizability, lack of integration with in-house resources, etc. So far, such challenges have been mainly addressed by providing targeted technical solutions, when possible. Our approach starts by setting up business requirements for the cloud using capabilities thus facilitating developers to structure their systems for delivering cloud services capable of adhering to changes in the business environment.

3 Capability Modeling

Capability driven development of business has two perspectives - design and delivery, sometimes refereed to as design-time and runtime. Design is addressed by enterprise modeling, i.e. by eliciting business goals, Key Performance Indicators (KPI), designing generic business processes and resources, as well as by specifying capabilities, relevant context sets and patterns. Capability delivery is addressed in the meta-model by specifying actual context situations, as well as the services for pattern

delivery. In [22], we have proposed a design for capability. In what follows, we have further elaborated that proposal, and also added the perspective of capability delivery.

3.1 Capability Design

Enterprise Modeling. This part starts with the representation of Goals, and the Processes realizing these goals using required Resources (Figure 1). These are essential components of business planning, and their relationships as presented in the figure are common to many EM approaches, for instance EKD [23], [24]. Furthermore, Key Performance Indicators (KPI) should be set up to measure the achievements of goals [25]. The main components in the meta-model needed for planning business variability are Capability and Context.

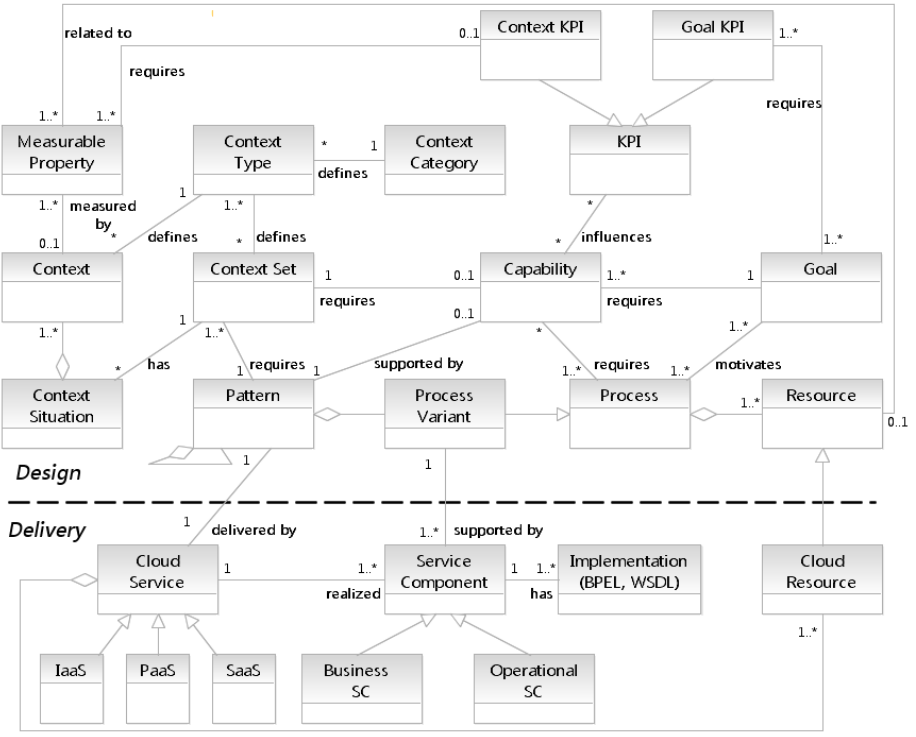

Fig. 1. Meta-model for capability design and deployment in cloud

In essence, Capability formulates the requirements for the ability of accomplishing a Goal, realized by applying a solution described by a capability delivery Pattern. This realization requires certain business Processes, Process Variants and Resources, such as infrastructure or IT components. The distinguishing characteristic of Capability is that it is designed to deliver a business solution for specific Context Sets that are represented by Context Situations at runtime. It essentially links together business Goals, related

business Processes and Resources, with delivery solutions by distinguishing the business contexts in which certain Patterns capturing business Process Variants should be applied.

Context Modeling. Following the related research results outlined in Section 2.2, the context encompasses the information characterizing the situation in which a business capability should be provided. Thus, it may include those who provide the business, or are its customers; the resources used in the business exchanges, as well as surrounding conditions. For example, the provisioning of mobile services is related to consumers, infrastructure, and regulations having certain conditions in times and locations. We have elaborated a framework for capturing this in Table 1.

Table 1. Context framework - Context categories and measurable properties

Context Category	Relevance	Availability	Feature	Time	Location
Subjects Organization Customers Partners Competitors	What is subject doing?	Is subject available?	Characteristic or quantity of subject	When does subject perform process?	Where is subject located?
Objects Infrastructure Artefact Service	How is object used?	Is object available?	Characteristic or quantity of object	When is object used?	Where is object located?
Environment Regulations Standards Weather	What is the influence of environment?	Is environment concept available?	Characteristic or quantity of environment	When is environment concept applicable?	Where is environment concept located?

In the meta-model, Subjects, Objects and Environment from Table 1 are modeled as Context Category (Figure 1). For each category, its relevance to a business capability is to be assessed. If found relevant, then it is important to capture the measurable information of a category – if it is available, which features it has, when (time), and where (location) it is used. A Context Category is concretized by eliciting a number of the relevant Context Types, such as "weather information supplier" for Category "Partners". A Capability is designed to be adequate for certain context situations represented by a Context Set, i.e. a range of Context Types (such as "weather information supplier", "pricing information supplier", "EU customers", etc.) Each Context Set can be materialized with a number of Context Situations according to the specification of the set by observing or measuring different individual Contexts. The Context KPI component defines desired KPIs that can be related to Contexts and measured using the Measured Property component, which is of a vital importance for monitoring capability delivery. In this regard, we envision that in real application cases a collection of Measurable Properties might have to be established to measure a specific Context (Table 1).

Capability Delivery Pattern Modeling. The Pattern component describes an actual solution for realizing a Capability (Figure 1). Each pattern describes how a certain Capability is to be delivered within a certain Context Situation and what resources, process, and IS components are needed. Patterns typically describe which Process

Variants should be used in accordance to a Context Set. At runtime patterns are applied 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 pattern applied for capability realization is still valid for the current context situation. If the pattern is not valid, then capability realization should be dynamically adjusted by applying a different pattern, by reconfiguring the existing pattern (i.e., changing a utilized process, reassigning resources etc.), or by aggregating several patterns into a new pattern. Technically, the context information is captured using a context platform in a standardized format.

3.2 Capability Delivery

The design of the capability meta-model as described in the previous section can be implemented in different ways, where the delivery using the cloud-based services is the objective of this work.

A consideration in regard to the delivery of business capabilities in the cloud may start when the Goals are modeled. In addition to those of the core business, certain goals will set up the objectives in regard to the access to the offered business capabilities; in case of the access through cloud, they will concern the functionality and the quality of cloud's implementation. Thus, aside from setting the basic goals for facilitating the delivery of capabilities through the cloud, recalling Section 2.3, availability of services, performance, security, and integration with in-house ICT resources, can also be of high importance to enable constant and smooth delivery of cloud services. Once these objectives are modeled as goals in a capability model and prioritized appropriately, they are linked with concrete Goal KPIs, which will set further requirements for certain Capabilities.

Once a capability Pattern requiring a delivery in the cloud is designed, it will be realized through a Cloud Service, which will offer a IaaS, PaaS, SaaS, or a combination of those models (see Figure 1). Following further common proposals for the cloud architecture [20], [21], the implementation of a chosen cloud model will be supported by Service Components, where one or more are used to realize a required Process Variant. Here, Business Service Components represent the set of business-related services, such as Order Management, Pricing, etc. Operational Support Services represent the set of management and technical-related services, such as Service Delivery Catalog, Service Automation Management, Virtualization Management, etc. [20]. Both the design and the reusability of those service components is enabled through the capability model, i.e. through Patterns with their related Process Variants and Resources.

Concerning the management of resources in the capability delivery in the cloud, the meta-model enables the selection of the needed resources for a delivery through Cloud Resources. At runtime, the resources are monitored through context-related KPIs to facilitate the change in their use, as the context changes.

4 Case Study

To exemplify the proposed approach for capability modeling and aligning with cloud services, we present a case from the EU 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. Both, long term strategic investments in building energy technologies, as well as short term operational planning are considered. The EnRiMa DSS should perform the following core functionality:

(1) Importing data from various sources, such as, pricing data, weather data from sensors and forecasts, as well as operational data from Building Energy Management Systems (BEMS),
(2) Setting up parameters for decision making,
(3) Setting target temperature for optimization,
(4) Optimizing energy flows in a building, as well as
(5) Viewing optimization results and feeding temperature set points into BEMS.

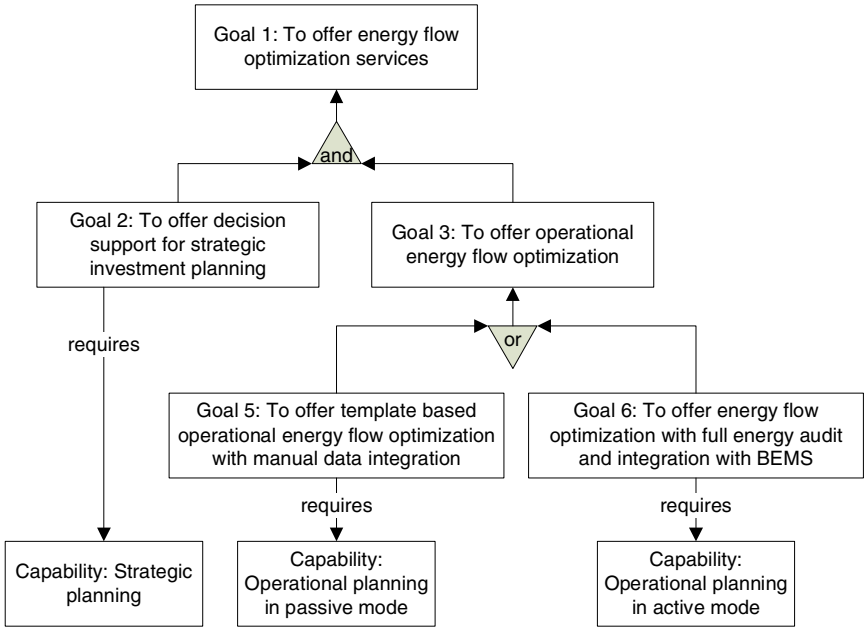

Fig. 2. Business goals and capabilities for key variants of the EnRiMa DSS

In this paper we analyze the delivery aspects of the DSS with respect to using cloud services. The DSS needs to exchange data with the BEMS, where a key challenge is the use of ICT in the building. Older buildings have older ICT infrastructure and may not have real BEMS; newer building most often have BEMS with a possibility to integrate with the installed energy technologies. As a result, in older building the data exchange may require manual interventions. Figure 2 shows a Goal Model fragment addressing the two alternatives of offering operational energy flow optimization.

Goal 5 and 6 define the alternatives depending on the presence of, and of the integration possibilities with the BEMS. By analyzing the goals and application contexts we can elicit three capabilities:

- *Strategic planning* providing the building operator and other stakeholders with decision support for investment planning based on historical data and scenarios for prices and weather.
- *Operational planning in passive mode* providing operational planning based on template based energy audit and human assisted or manual input of data into DSS and manual input of temperature set points for the next 24 hours into the system used for controlling.
- *Operational planning in active mode* providing operational planning based on full energy audit, and daily update of sensor data and automatic transfer of set-points into the BEMS using the BACnet/IP protocol.

Tables 2 to 4 present different context sets for the capabilities shown in Figure 2, as obtained using the context categories from Table 1 as guidelines.

Table 2. Context Set for Capability: Strategic Planning

Context Type	Relevance	Availability	Feature	Time	Location
Subjects					
Weather data provider	Provides weather data	Yes, as cloud service	Date of last update	Every day	Not relevant
Pricing data provider	Provides pricing data	Yes, as cloud service	Date of last update	Every day	Not relevant
Objects					
BEMS	For data input and output	No	Not relevant	Not relevant	At the building site
Sensors	For building environment, weather	No	Sensor readings	Every 15 min	At the building site

Table 3. Context Set for Capability: Operational planning in passive mode

Context Type	Relevance	Availability	Feature	Time	Location
Subjects					
Weather data provider	Provides weather data	Yes, as cloud service	Date of last update	Every day	Not relevant
Pricing data provider	Provides pricing data	Yes, as cloud service	Date of last update	Every day	Not relevant
Building operator	Update energy usage and weather data	Yes	Not relevant	Every 24 hours	At the building or remotely
Objects					
BEMS	For data input and output	No	Not relevant	Not relevant	At the building site
Sensors	For building environment and weather	No	Sensor readings	Every 15 min	At the building site

For the sake of brevity we have addressed only a subset of contexts that influence the decisions to deploy the DSS as cloud service or locally. The capabilities and contexts were elicited collaboratively and iteratively by analyzing the goals and generic processes of deploying the DSS (not shown here).

Table 4. Context Set for Capability: Operational planning in active mode

Context Type	Relevance	Availability	Feature	Time	Location
Subjects					
Weather data provider	Provides weather data	Yes, as cloud service	Date of last update	Every day	Not relevant
Pricing data provider	Provides pricing data	Yes, as cloud service	Date of last update	Every day	Not relevant
Building operator	Enters optimization targets, e.g desired temp.	Yes	Not relevant	Every 24 hours	At the building or remotely
Objects					
BEMS	For data input and output	Yes	Not relevant	Not relevant	At the building site
Sensors	For building environment and weather	Yes	Sensor readings	Every 15 min	At the building site
Environment					
Communicatio n protocol BACnet/IP	To integrate with the BEMS	Yes	Not relevant	Not relevant	Local ICT systems

In the reminder of this section we will discuss capability delivery options.

The delivery of the capabilities in Figure 2 and their contexts shown in Tables 2-4, requires the activation of different process variants for the base processes such as Payment, Data Isolation, etc. For example, the delivery of the capability “Operational planning in active mode” requires integrating DSS with the local BEMS to feed back the temperature set-points, while the capability “Operational planning in passive mode” makes use of a manual process variant for adjusting the system.

As indicated in the meta-model (Figure 1), the selection of delivery for the service components also partially determines the process variants that need to be used. To exemplify how the notions of process variants and patterns can be used to determine the consequences of the delivery options, and hence the needed process variants, we here discuss the options of having the EnRiMa DSS service components deployed locally or as cloud services. To start with, delivery via a local installation or via the cloud can be considered as two separate delivery patterns. Each of these patterns has their own sub-patterns with associated process variants (see the meta-model, Figure 1). To exemplify how the choice of a delivery pattern (cloud or local) affects the choice of sub-patterns, we use three base processes. For each of the three processes we describe the process variants to be used in the cloud delivery case and in the case of a local installation, as well as their implications for the EnRiMa case:

Payment process. Variants: pay-per-use, one-time fee. For cloud delivery it is rational to let the users of the DSS to pay a monthly or yearly fee, since the use of a cloud platform will incur cost for the provider. For a local installation it can be assumed the organization buying the service will provide own hardware, thus allowing a one-time fee. In the EnRiMa case it is likely that other services (e.g. energy audits) will be performed by the organization providing the DSS, thus this points toward using a flexible pay-per-use fee.

Data isolation process. Variants: single-tenancy, multi-tenancy. While using a local installation there is no need to separate data belonging to different organizations – each organization will have their own data storage. However when using the cloud delivery, it is efficient to use the same data storage, according to a multi-tenancy model. For the cloud delivery pattern it thus makes sense to have a separate service component that ensures that all data stored are tagged with the correct organizational origin. In the case of the EnRiMa DSS, the system will not handle sensitive data. However other applications might consider an operational service component that performs encryption when running in a multi-tenancy environment.

Deployment and update process. Variants: Local installation scripts, cloud upload. The deployment of software to the cloud and locally can differ, thus to support both cloud delivery and local installations there is a need to have separate process variants for each. Some cloud platform, such as Amazon EC2 IaaS service, support the upload of pre-configured virtual machines, while others, such as Google App Engine PaaS service requires the service components to have specific format compliant with the platform. In the EnRiMa DSS case the DSS user interface services component is developed in a format that is easily transferrable to Google App Engine PaaS, however the optimization algorithms will need specific software, thus requiring a more flexible IaaS deployment.

The above process variants are examples of how an enterprise wishing to consider the cloud capability delivery needs to find variants of their processes, and eventually package them as patterns for efficient reuse and development of new capabilities. Further areas to consider is how the run-time performance is monitored, that is, how it is ensured that the capability are delivered within the context it was designed for. For this purpose the defined context-KPIs can be used.

5 Overview of Capability Driven Development

The capability driven approach based on the capability meta-model proposed in Section 3 is supported by a development methodology and a development environment. The development methodology, as initially outlined in [22], consists of three cycles, namely, Design, Delivery and Updating. The design cycle covers the design perspective of the capability development, the delivery cycle covers the delivery perspective of the capability development, and the updating cycle uses capability delivery experiences to create new and update existing capability delivery patterns. The three cycles are supported by the capability driven development environment (Figure 3). The main components of the environment are the capability

design tool for the capability design, cloud services for the capability delivery, and the context platform for capturing context data.

The capability design cycle starts with capability modeling as defined by the meta-model. Existing enterprise models and/or architecture models will be used at this stage, and the capability designs integrated with those models. The modeling is supported by the capability Modeling Module, which is integrated with the overall enterprise architecture proving information about available assets and resources as well as enterprise goals and processes. The capability modeling purpose is to capture business goals, KPIs and business processes on a generic level. The capability modeling is followed by identification of appropriate patterns for capability delivery. The patterns are stored in the Repository of Patterns, and the Composition Module provides means for combing individual patterns. The capability delivery is composed in a way to allow for multiple process execution variants [26]. At the design stage, the context platform provides specifications of available context data and these are mapped with the capability context set. The Integration Module binds patterns with the corresponding executable components.

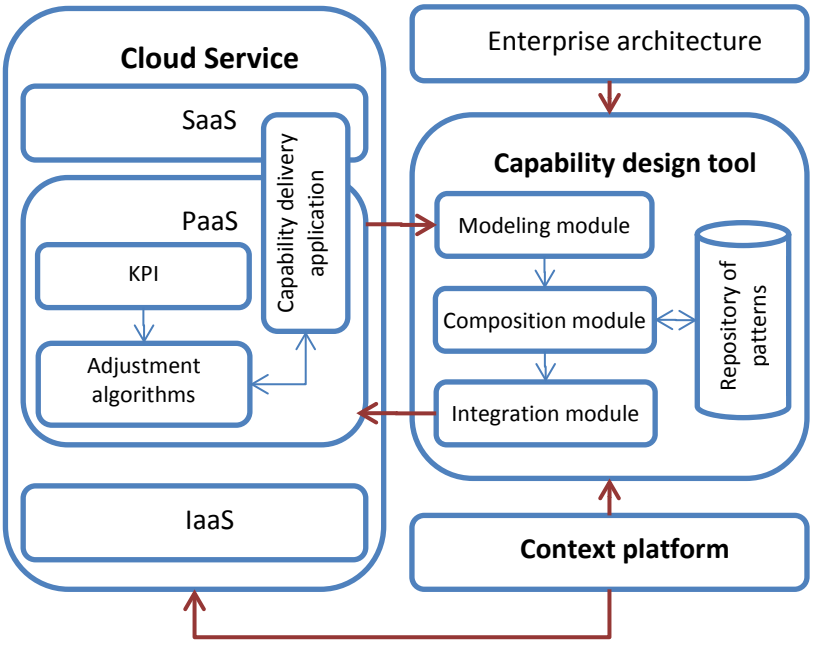

Fig. 3. Capability driven development environment

The result of the capability design cycle is a business capability developed and software implementing this capability, called Capability Delivery Application (CDA), which is delivered as a cloud service. Technical requirements towards the cloud service can be represented using a service provisioning blueprint [27]. Based on the models and available platforms the best option for deployment can be selected. As discussed in the EnRiMa case description in the previous section, the deployment

could be made to different cloud platforms. The selection of platform is supported by the use of patterns. Given that each pattern is associated with a set of measurable properties, it is possible to match a defined deployment environment (such as Google App Engine) to the given context.

The context is provided by the context platform [28] in a standardized format (e.g. XCoA). CDA is deployed in the cloud platform, and some parts of capability delivery can be provided as cloud based software services. During the delivery cycle, CDA uses KPI indicators for monitoring capability delivery and run-time adjustment algorithms (AA) to change capability delivery with regards to the changing context.

In response to the changes in the context, the adjustment algorithms can switch from one capability delivery process variant to another or provision additional computational resources from the cloud services. If the context changes cannot be accommodated in the run-time and the patterns used are no longer applicable for the given context, the capability design cycle is repeated and capability delivery patterns are updated in the repository of patterns.

6 Conclusion and Future Work

We have proposed to support the design of business capabilities by using enterprise modeling techniques as a starting point, and to employ model-based patterns to describe how the software application can adhere to changes in the execution context.

Concerning the delivery of business capabilities, we have considered the cloud architecture, with the following motivations – the rationale for cloud computing lies in the significant scalability and resource virtualizations, but economic viability forces cloud providers to manage suppliers' contracts based on actual demands. Our meta-model for capability design and deployment resolves this by enabling the delivery of varying business capabilities according to different business contexts. Furthermore, cloud offerings need business level assessments and the optimization of the usability in different business contexts. With the proposed approach business modelers will be able to motivate the business requirements for the cloud using goals and KPIs in accordance to changing contexts. The solution will also facilitate developers to structure their systems for delivering cloud platforms to meet required business contexts.

Development of enterprise business capabilities will be achieved by modeling them according to the capability meta-model (Section 3) and supported by the capability development environment (Section 5). The modeling process will be based on the EM process (see e.g. [23]), but more specific modeling guidelines for designing capabilities, modeling context, and creating patterns will be elaborated as future work. The modeling is based on EM components understandable to business stakeholders, such as goals, KPIs, processes, and resources and in principle is independent of any specific EM language. The linkage of the available enterprise components with different business contexts is done relying on the principle of reuse and execution of software patterns with the principle of sharing best practices of organizational patterns. In our meta-model, patterns represent reusable solutions in terms of business process, resources, and supporting IT components (e.g. cloud services) for delivering a specific type of capability in a given context.

The aim of this research is to contribute to the business-driven application development and the emergence of new kinds of interoperable cloud-based services thus stimulating innovation and performance in businesses.

Regarding future work, our main interest is set to the tool support for capability driven development, which will at the run-time facilitate dynamic adjustments of capabilities according to changing contexts, by reconfiguring the use of processes and service components in accordance to available patterns. Among the key challenges to be addressed are (1) the process of capturing, creating, collecting feedback about, and managing patterns, (2) the implementation of algorithms for their dynamic adjustment, as well as (3) the deployment for different cloud platforms.

References

1. Barney, J.B.: Firm Resources and Sustained Competitive Advantage. Journal of Management 17(1), 99–120 (1991)
2. Open Group Standard: TOGAF - Enterprise Architecture Methodology, Version 9.1, http://www.opengroup.org/togaf/ (last accessed December 07, 2012)
3. Open Group Standard: ArchiMate - Modeling Language for Enterprise Architecture, Version 2.0, https://www2.opengroup.org/ogsys/catalog/c118 (last accessed December 07, 2012)
4. Ulrich, W., Rosen, M.: The Business Capability Map: Building a Foundation for Business/IT Alignment. Cutter Consortium for Business and Enterprise Architecture, http://www.cutter.com/content-and-analysis/resource-centers/enterprise-architecture/sample-our-research/ea110504.html (last accessed December 07, 2012)
5. OASIS: Reference Architecture Foundation for Service Oriented Architecture Version 1.0, Committee Specification Draft 03 / Public Review Draft (July 02-06, 2011), http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra.pdf (last accessed November 01, 2012)
6. Armbrust, M., et al.: A View on Cloud Computing. Communications of the ACM 53(4), 50–58 (2010)
7. Wikipedia, the Free Encyclopedia. Business Capability Specification (Accenture), http://en.wikipedia.org/wiki/Accenture (last accessed November 15, 2012)
8. Collis, J.D.: How Valuable are Organizational Capabilities? Strategic Management Journal 15, 143–152 (1994); Issue: Special Issue: Competitive Organizational Behavior
9. Teece, D.J., Pisano, G., Shuen, A.: Dynamic capabilities and strategic management. Strategic Management Journal 18(7), 509–533 (1997)
10. Bhatt, G.D., Grover, V.: Types of Information Technology Capabilities and Their Role in Competitive Advantage: An Empirical Study. Journal of Management Information Systems 22(2), 253–277 (2005)
11. Shoham, J.: Agent-Oriented Programming. International Journal of Artificial Intelligence 60, 51–92 (1993)
12. Vieira, R., Moreira, Á.F., Bordini, R.H., Hubner, J.: An Agent-Oriented Programming Language for Computing in Context. In: Debenham, J. (ed.) Professional Practice in Artificial Intelligence. IFIP, vol. 218, pp. 61–70. Springer, Boston (2006)

13. Schilit, B., Adams, N., Want, R.: Context-aware Computing Applications. In: First International Workshop on Mobile Computing Systems and Applications, pp. 85–90 (1994)
14. Pascoe, J.: Adding Generic Contextual Capabilities to Wearable Computers. In: Proceedings of 2nd International Symposium on Wearable Computers, pp. 92–99 (1998)
15. Dey, A.: Understanding and Using Context. Personal and Ubiquitous Computing 5(1), 4–7 (2001)
16. Dey, A., Abowd, G.: The Context Toolkit: Aiding the Development of Context Aware Applications. In: Proc. on Workshop on Software Engineering for Wearable and Pervasive Computing, pp. 68–80 (2000)
17. Gross, T., Specht, M.: Awareness in Context-Aware Information Systems. Mensch and Computer 5(8), 173–182 (2001)
18. Hervas, R., Bravo, J., Fontecha, J.: A Context Model based on Ontological Languages; a proposal for Information Visualisation. Journal of Universal Computer Science (J. UCS) 16(12) (2010)
19. Foster, I., Yong, Z., Raicu, I., Lu, S.: Cloud Computing and Grid Computing 360-Degree Compared. In: Proceedings of Grid Computing Environments Workshop, GCE, pp. 1–10 (2008)
20. Behrendt, M., et al.: Introduction and Architecture Overview. IBM Cloud Computing Reference Architecture 2.0, https://www.opengroup.org/cloudcomputing-/uploads/40/23840/CCRA.IBMSubmission.02282011.doc (last accessed December 07, 2012)
21. National Institute of Standards and Technology: NIST Cloud Computing Reference Architecture, Version 1 (March 28, 2011), http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/Meeting12A ReferenceArchitectureMarch-282011/NIST_CCRATWG_029.pdf (last accessed December 07, 2012)
22. Stirna, J., Grabis, J., Henkel, M., Zdravkovic, J.: Capability Driven Development – an Approach to Support Evolving Organizations. In: Sandkuhl, K., Seigerroth, U., Stirna, J. (eds.) PoEM 2012. LNBIP, vol. 134, pp. 117–131. Springer, Heidelberg (2012)
23. Bubenko Jr., J.A., Persson, A., Stirna, J.: 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 (2001)
24. Loucopoulos, P., Kavakli, E., Prakash, N., Rolland, C., Grosz, G., Nurcan, S.: Requirements Engineering: Panacea or Predicament. UMIST, Manchester (1997)
25. Chen, C.: An objective-oriented and product-line-based manufacturing performance measurement. International Journal of Production Economic 112(1), 380–390
26. Lu, R., Sadiq, S., Governatori, G.: On managing business processes variants. Data & Knowledge Engineering 68, 642–664 (2008, 2009)
27. Nguyen, D.K., Lelli, F., Taher, Y., Parkin, M., Papazoglou, M.P., van den Heuvel, W.-J.: Blueprint Template Support for Engineering Cloud-Based Services. In: Abramowicz, W., Llorente, I.M., Surridge, M., Zisman, A., Vayssière, J. (eds.) ServiceWave 2011. LNCS, vol. 6994, pp. 26–37. Springer, Heidelberg (2011)
28. Gomes, D., Gonçalves, J.M., Santos, R., Aguiar, R.: XMPP based Context Management Architecture. In: Proceedings of the IEEE GLOBECOM Workshop, December 6-10, pp. 1372–1377 (2010)