Development of Capability Driven Development Methodology: Experiences and Recommendations

Janis Stirna[1], Jelena Zdravkovic[1], Jānis Grabis[2], Kurt Sandkuhl[3]

[1]Department of Computer and Systems Sciences, Stockholm University, PO Box 7003, 164 07, Kista, Sweden
[2]Institute of Information Technology, Riga Technical University, Kalku 1, Riga, Latvia
[3]Institute of Computer Science, Rostock University, 18051 Rostock, Germany
[js, jelenaz]@dsv.su.se, grabis@rtu.lv, kurt.sandkuhl@uni-rostock.de

Abstract. The field of Information Systems (IS) and Enterprise Modeling (EM) is continuously striving to address the challenges of the practice by developing new methods and tools. This paper presents experiences and lessons learned from the Method Engineering of the Capability Driven Development (CDD) methodology. The CDD methodology supports organizations operating in dynamic environments and integrates EM with information system (IS) development taking into account changes as the application context. The main focus is on presenting the CDD meta-model and the associated development activities as well as sharing the experience and recommendations for developing similar methods and tools.

Keywords: Enterprise Modeling, Meta-modeling, Method Engineering, Capability Driven Development

1 Introduction

Information Systems (IS) have to dynamically adapt to new and unexpected, often drastic, business opportunities and threats. To respond to this challenge of continuous adaptation, the EU FP7 project "Capability as a Service in digital enterprises" (CaaS) [1] developed a methodology for capturing and analyzing the influence of the business application context on the IS using the notion of capability. The concept of capability is generally used as an abstraction to define what a core business does [2]. For instance, "*an ability and a capacity for an enterprise to deliver value, either to customers or shareholders, right beneath the business strategy"* [3], or *"the ability of one or more resources to deliver a specified type of effect or a specified course of action"* [4]. The CaaS project has developed an integrated methodology for context-aware business and IT solutions: *Capability Driven Development* (CDD). It consists of a modeling language and guidelines for the way of working. The areas of modeling performed as part of CDD are Enterprise Modeling (EM), context modeling, variability modeling, adjustment algorithms, and patterns for capturing best practices.

The development of the CDD methodology followed principles defined during analysis of use case requirements and documented in [5]:

- The CaaS project should not develop a single methodology mandatory for all business cases, but a reference methodology for using in majority of cases and pathways of extending the reference methodology to proprietary methodologies.
- All concepts of the methodology should be based on a common meta-model.
- The CDD methodology should not be a monolithic block but component-oriented to allow flexible use of selected method components depending on the intentions of an organization and a particular development situation.
- Integration of existing methods or method components should be given preference before substituting them with new.
- The CDD methodology is to be supported by the CDD Environment, a part of which is the *Capability Design Tool* (CDT) implemented in Eclipse.

The objectives of this paper are (i) to report on the process that led to development of the CDD methodology, (ii) to share the experiences method development, and (iii) to formulate a set of guidelines for development of EM methods.

The research approach followed the principles of design science [8] and consisted of several design and evaluation cycles. The proposed CDD methodology has been applied and validated in 4 use case companies of the CaaS project.

The rest of the paper is organized as follows. Section 2 presents the research approach taken, while section 3 gives a background to method development. The process of method development that took place is presented in section 4 while the CDD meta-model and the CDD process is summarized in section 5. Section 6 presents the experiences and recommendations for method development, while section 7 summarizes and provides concluding remarks.

2 Research Methodology

Within IS engineering, Design Science Research (DSR) is a problem-solving paradigm which aims to resolve problems by creating innovative scientific artifacts through development- and evaluation cycles within an operating context (organizational domain, social setting, environment, etc.) [8]. The creation of artifacts evolves iteratively and incrementally through a research process and results in a practical solution. The DSR process consists of the explication of the problem, an outline of the artifact with the related requirements, artifact's design and development, as well as its demonstration, evaluation, and communication.

Our research concerns the CDD methodology as the main *design artifact*, for enabling IS development to capture changes in business context through variability terms and to accordingly adapt using adjustment algorithms. This design artifact is composite because the components of the methodology, e.g. the meta-model, are also design artifacts. Furthermore it is closely related to another design artifact of the project – the CDD Environment. This paper presents the experience of applying the DSR paradigm to the development CDD methodology and environment. The DSR process of constructing the artifact according to needs of multiple stakeholders was iterative and incremental. Participatory modeling workshops, focus-groups sessions, and questionnaires were the main techniques used for requirements elicitation. The artifact was developed and validated during a number of design cycles, notably, two cycles of initial feasibility design and analysis [5], three parallel cycles of application of the methodology at the use case companies leading to the reference CDD methodology [6], followed by a number of design-validation cycles for development of extensions of the methodology (available from [1]).

3 Background to Method Development

Method engineering (ME) is *the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems* [9]. One of the first efforts in modeling of modeling methods (meta-modeling) was proposed by [10] and development of customizable tools for supporting various modeling approaches (meta-tools) by [11]. The main motivation was to search and adopt or tailor existing methods, as well as, to develop a new method by designing its modeling language, way of working, and tool support. The efforts concentrating on tool support became known as Computer Aided Method Engineering, c.f. [12, 13].

The need to adapt methods and tools according to organizational needs has been addressed by Situational Method Engineering (SME) [14]. Recently it has been systematically presented in [15]. In a nutshell, SME is an ME approach that includes all aspects of creating, using and adapting an IS development method based on local conditions. This is achieved by designing method parts, i.e. method chunks [16], supporting the realization of some specific IS development activity as well as by tailoring by extracting a set of appropriate method parts assessed based on local situational factors (e.g. the business sector, or size of the business). Each method part is represented according to a same template and adheres to a unique meta-model.

Another practicable ME approach was proposed in [17]; it sets a high attention on the elaboration of method parts such as the procedures for meta-modeling, i.e. for choosing appropriate concepts for inclusion.

Since the CDD methodology aimed for creating a new method for IS development based on the notion of capability because any similar has not existed, the core concern was to correctly identify the main method parts and their relevant concepts. For that reason, the approach described in [17] has been chosen. It proposes that methods are to be described in terms of the following aspects:

- *Purpose*: every method component has to clearly state its purpose, e.g. what modeling or problem-solving task it supports. Furthermore, a method usually describes the procedure for the modeling task from a particular perspective (e.g. business goals, process), which influences what is considered important when following the procedure. This perspective should also be stated explicitly.
- *Overview* to method components describes the relationships between the individual method components, i.e. which components are to be used and under what conditions, as well as the sequence of the method components (if any).

- *Method component* defines in operational terms what are the modeling language (in terms of concepts and notation) and procedure to be used. The *concepts* specify what aspects of reality are regarded as relevant in the modeling process, i.e. what is important and what should be captured a model. These relevant concepts and their relationships should be named in the method component and explained if necessary. The *procedure* describes how to identify the relevant concepts in a method component. It may also state prerequisites, resources, input, output, and tool support. In some cases it includes guidelines of modeling and assessing model quality. The *notation* specifies how the result of the procedure is to be documented, i.e. the graphical symbols, providing appropriate representation for each concept and for the relationships between them.
- *Forms of cooperation*: many modeling tasks require a range of specialist skills or cooperation between different stakeholder and developer types, i.e. roles in the project. The skills and roles are described along with the responsibilities and the forms of cooperation, e.g. who will take responsibility for each task or method component, will it be participatory or analyst-driven modeling.

The conceptualization of the relevant aspects is an important concern when designing method components. This is typically done using meta-modeling to specify a modeling language in a declarative manner, to generate a tool for its support. A key challenge is to organize ME and tool development in such a way that it is based on common modeling constructs and structure. In this regard MOF meta-modeling architecture [18] defining four modeling layers – from M3 (meta-meta model layer) to M0 (instance layer). Modeling languages are typically specified at M2 (meta-model layer). Once they are used to describe models reflecting reality, M1 model layer is populated. When the models at M1 level are instantiated M0 level is reached.

The CDD methodology has been defined by an M2 model, and as we show in section 5, we have obtained it starting from a conceptual M2 model to enable communication as well as to reach the agreement for the meta-model requirements among the methodology's stakeholders.

4 Overview of the Process of Method Development

The following phases of method development took place: (i) requirements elicitation and analysis of the business motivators; (ii) method development first iteration – base line methodology, (iii) second iteration of method development, focusing on fine-tuning the base line methodology and the creation of regular CDD methodology as well as elaborating method extensions; (iv) integration of the method extensions and packaging for exploitation – final version of the CDD methodology.

4.1 Requirements Elicitation

The motivation for the CDD methodology development was analyzed in the initial requirements elicitation phase of the project. This was done by interviews with the use case companies, survey with a large number of external companies, as well as by several iterations of methodology development and capability designs for the four use case companies in order to validate the initial versions of the modeling language. This allowed us to elaborate the overall goals (see Fig. 1) and requirements for the CDD methodology, define an initial conceptual meta-model for representing capability designs, and to outline method components. Results of this work are reported in [5].

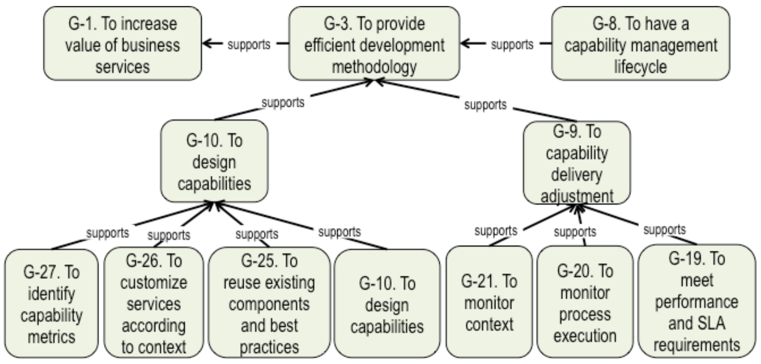
G-1. To increase value of business services
supports
G-3. To provide efficient development methodology
supports
G-8. To have a capability management lifecycle
supports
supports
G-10. To design capabilities
G-9. To capability delivery adjustment
supports
supports
supports
supports
supports
supports
supports
G-27. To identify capability metrics
G-26. To customize services according to context
G-25. To reuse existing components and best practices
G-10. To design capabilities
G-21. To monitor context
G-20. To monitor process execution
G-19. To meet performance and SLA requirements

Fig. 1. A goal model fragment for the CDD methodology, adapted from [5]

4.2 Development of the of the CDD Methodology – Base Line

The CDD methodology defines both aspects that comprise a modeling methodology – (1) *the modeling language* in terms of concepts, relationships, and notations used to represent the modeling product, i.e. the models of capability designs created, and (2) the way of working, the procedures and tools used, to arrive at a capability design of good quality i.e. *the modeling process*. The CDD methodology consists of a number of interlinked method components [6] described according to the framework of [17].

The CDD method components were divided into *upper-level method components* and *method extensions*. At this stage the upper-level components were designed according to the requirements and business goals elicited in the previous phase and the initial versions were documented. The resulting methodology was denoted, *base line methodology* and it included the following components:

- *Capability Design Process.* It describes an overview on how to design capabilities by using process models, goal models and other types of models.
- *Enterprise Modeling.* The component guides the creation of enterprise models that are used as input for capability design. We have incorporated the 4EM approach for the purpose of this component.
- *Context Modeling.* It describes the method components needed for analyzing the capability context, and the variations needed to deal with variations.
- *Reuse of capability design.* This component contains guidelines for the elicitation and documentation of patterns for capability design.
- *Run-time Delivery Adjustment* for development of capability runtime adjustments including implementation of capability delivery adaptation algorithms.

The base line methodology was applied and validated by application in the following use case companies:

- SIV AG (Germany) for business processes outsourcing (BPO) and execution capability.
- Fresh T Limited (UK) for maritime compliance capability.
- CLMS Ltd (UK) for collaborative software development using the MDD technology and i-Symbiosis application in particular.
- Everis (Spain) for service promotion capability, marriage registration capability, government SOA platform management capability.

4.3 Development of the of the CDD Methodology – Regular Methodology

The base line methodology was applied in the use case companies of the CaaS project and the application results contributed to further improvements and development of the next version of the CDD methodology, denoted *regular methodology.*

The main tasks at this stage was development of new subcomponents for the upper-level components, defining additional and more detailed procedures for the ways of working, as well as refinement of the meta-model, e.g. changes of multiplicities representing model integrity rules, and introducing new components needed for representing information needed by the newly developed method subcomponents. In addition, *method extensions* addressing specific business challenges to which the regular methodology can be applied were developed as part of the process of applying the base line methodology (c.f. [1]). The main purpose of the method extensions is to broaden the range of problems to which CDD can be applied. There following method extensions were developed:

- The *Capability Ready Business Services* covers the transition from textual instructions and activity descriptions to process models. With this extension many more BPO services can be designed as capabilities.
- The *Prepare Local and Global Optimization* improves service delivery by balancing the local optimization of services provided to a client and global optimization from a Business Service Provider (BSP) perspective.
- The *Evolutionary Development of Business Information Exchange Capability* helps organizations to develop capabilities in the case when pre-existing capability delivery solution must be tailored to the needs of a new client.
- The *Integration of CDD and MDD* for analyzing the potential for integrating MDD and CDD concepts. MDD is sharing a common ground with the CDD approach because both use models for analysis and design.
- The *Analysis of Capability Relationships* is proposing an analysis of capability relationships and mapping capabilities to delivered services including those offered by external partners.
- The *Predictive analysis* describes capability delivery adjustment using predicted context values to attain proactive behavior.
- The *Capacity evaluation* evaluates capability delivery capacity requirements to determine capability's suitability to context ranges.

4.4 Development of the Final version of the CDD Methodology

At this stage the upper level method components and method extensions had been applied and tested in several iterations in the use case companies and hence were considered relatively stable, i.e. only minor refinements to the documentation were performed, e.g. for eliminating redundancies and inconsistencies in the documentation, improving the understandability of the definitions.

Considering the project's aim to deliver a method for practical use, an additional method component to *Support Executive Decision Making* for the adoption of CDD in organizations. This method component defines the steps for CDD adoption as well as specifies the organizational roles needed for its successful and long-term use. The final version of the CDD methodology is reported in [19].

CDD was also analyzed with respect to the current EM and Enterprise Architecture contributions that include the concept of capability for similar purposes [20].

5 Overview of the CDD Meta-model

For the purpose of providing background, this section briefly presents the Capability Meta-model and the CDD way of working, the two aspects to which the experiences and lessons learned are related. The theoretical and methodological foundations for CDD are provided by the conceptual *core CaPability Meta-model* (CPM) in Fig. 2, c.f. [6] for details. CPM was developed on the basis of requirements from the industrial project partners and related research. It has three main sections:

a) *Enterprise model* representing organizational designs with Goals, KPIs, Processes (with concretizations as Process Variants) and Resources;

b) *Context model* represented with Context Set for which a Capability is designed and Context Situation at runtime that is monitored and according to which the deployed solutions should be adjusted. Context Indicators are used for measuring the context properties (Measuring Property); and

c) *Patterns and variability model* for delivering Capability by reusable solutions for reaching Goals under different Context Situations. 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.

Note that this is a simplified version of the CPM showing the key components of CDD; the version including definitions of components and associations is available in [19].

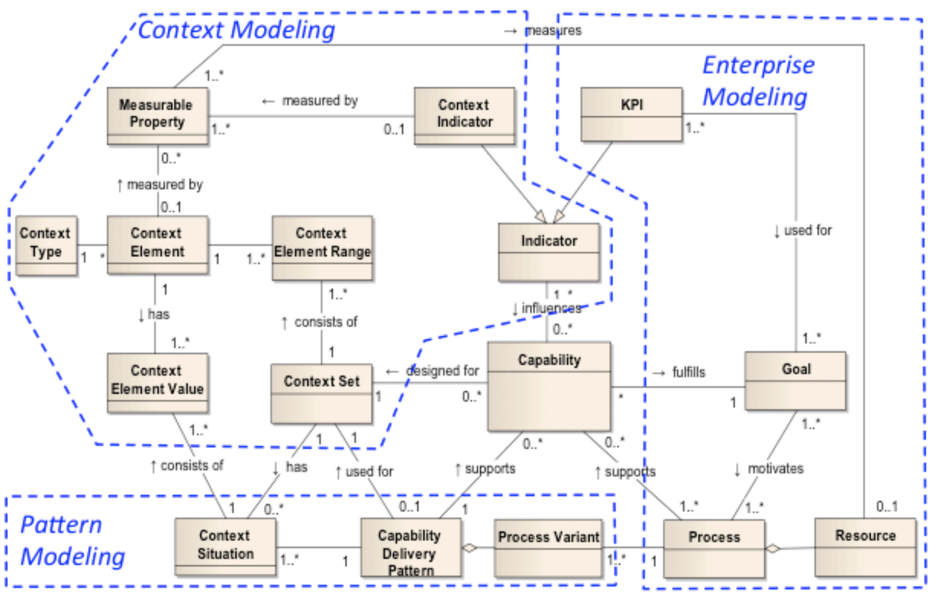

Fig. 2. A core meta-model for supporting Capability Driven Development [5]

Table 1. Concepts of the core capability meta-model

Concept	**Description**
Capability	Capability is the ability and capacity that enable an enterprise to achieve a business Goal in a certain context (represented by Context Set).
KPI	Key Performance Indicators (KPIs) are measurable properties that can be seen as targets for achievement of Goals.
Context Set	Context Set describes the set of Context Elements that are relevant for design and delivery of a specific Capability.
Context Element Range	Context Element Range specifies boundaries of permitted values for a specific Context Element and for a specific Context Set.
Context Element	A Context Element is representing any information that can be used to characterize the situation of an entity.
Measurable Property	Measurable Property is any information about the organization's environment that can be measured.
Context Element Value	Context Element Value is a value of a specific Context Element at a given the runtime situation, calculated from several Measurable Properties.
Goal	Goal is a desired state of affairs that needs to be attained. Goals should typically be expressed in measurable terms such as KPIs.
Process	Process is series of actions that are performed to achieve a result. A Process supports Goals, has input and produces output in terms of information and/or material.
Pattern	Patterns are reusable solutions for reaching business Goals under specific situational contexts. The context defined for the Capability (Context Set) should match the context in which the Pattern is applicable.
Process Variant	Process variant is a part of the Process, which uses the same input and delivers the same outcome as the Process in a different way.

The overall CDD process includes three cycles – capability design; capability delivery; and capability refinement/updating. It usually 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 new business opportunities requiring reconfiguration of existing or the creation of, e.g. new goals, business processes. This is followed by *capability design* – a formalized definition of requested capabilities and of relevant contexts, linking with relevant capability delivery patterns, as well as with supporting IT applications. The designed capability is then deployed and executed; this process is called *capability delivery*.

6 Experiences of the CDD Methodology Development

This section presents our findings in terms of development of the CDD methodology and Environment.

6.1 Iterative and incremental development of the meta-model

The work started by iterative development of the CPM in Fig. 2. It was used throughout the methodology development process. At first it presented the overall vision of the project consortium, the main components of a capability design, such as, capability, context, KPI, business process, which became clear in the early stages of the project. This was then validated in several iterations of, first instantiating the CPM, c.f. [5], and later applying to model the capability designs of the use case companies. All components in the meta-model had a textual description according to the following fields, component's name, description, purpose explaining why it should be used, associations including their purpose, and attributes.

There were four major versions of the CPM (**Fig. 3**) based on the initial version of the CDD methodology as well as 4EM that provided a core set of EM concepts. The initial CPM was developed prior to considering use cases, documented in [21]. It introduced concepts distinctive to capability, namely, context, indicators, patterns, and variants.

The first CPM development iteration within the project focused on the refinement of the existing concepts. This was based on high-level use case requirements for CDD. Initial capability models were developed as instantiations of the CPM. These capability models were further elaborated during analysis and design of the use cases, and this information was used in the second version. The main group of concepts added concern variability modeling as the use case partners found it important to represent contextual causes of variability in their capability models.

The next use case development stage focused on the actual implementation and delivery of capabilities and the CPM was extended to represent capability delivery aspects, which are the main executable parts of the capability design. The concepts of calculations and adjustments were added [22]. The former concerns transformation of contextual and performance data into context elements and indicators while the latter concerns definition of the capability delivery adaptation logics.

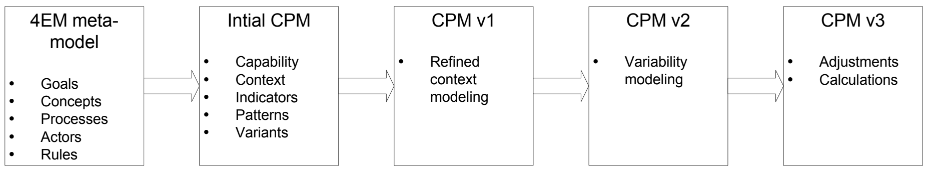

Fig. 3: Evolution of the CPM

In addition to the aforementioned iterations and the use case validations, further refinements were introduced after interactions with the CDD Environment development team. Some modeling components were difficult to understand by some method and tool developers in the project and discussing them from the point of view of meta-model and creating examples of capability models based on the meta-model proved useful. The most frequent changes were of multiplicities representing integrity constraints of the CDD methodology. There was a need to balance capability analysis and high level design needs with capability implementation and delivery needs. It was decided that the CPM v1 is used to communicate capability design concepts to business owners and analysts, while elements representing implementation details are defined in a separate view of the capability model (CPM v3).

Meta-modeling was instrumental to the process of designing method components with a clear purpose and precise semantics. This process was iterative during which CPM constructs and definitions were discussed in the method developer team to reach common understanding. A notable characteristic of CDD in comparison with other methods is that the same meta-model components are used by a considerably large number of method components and extensions. E.g. the constructs related to context modeling are used by almost all methodology components.

Development of all new method components started with the inclusion of the new modeling components in the CPM, which included certain restructuring and defining links to the existing components.

In summary, the following recommendations can be formulated:

- Develop the meta-model in design-validate iterations;
- Develop textual descriptions of the meta-model;
- Relate all method components to the common meta-model.

6.2 Method Components and Integration with Existing Methods

One of the initial decisions of methodology development was to base the CDD on existing methods and method components. The concept of method component proved to be very suitable because it allowed the development of support for the core tasks of CDD concurrently and in a modular fashion based on common principles.

It was assumed that capability design is based on EM and the CPM contains elements commonly used in EM. Hence, it was decided to incorporate an existing EM approach for the CDD tasks that are aligned to EM tasks. The 4EM approach was chosen because of three primary reasons: (1) 4EM sub-models are similar to method components and they are suited for modeling the perspectives of an organizational design (goals, business rules, process, concepts, actors and IS requirements) that are relevant for capability design, (2) the 4EM meta-model is formally defined, and (3) two of 4EM developers and authors of [7] participated the CDD method development.

Once the initial assessment of the suitability of 4EM was done, we investigated how the elements in the CPM correspond to the elements of the 4EM sub-models. Fig. 4 depicts this on a conceptual level with the dashed links showing the correspondence between the modeling perspectives of CDD and 4EM. The links show which sub-models of CDD can be supported by 4EM in the way that they use and where needed extend the 4EM models.

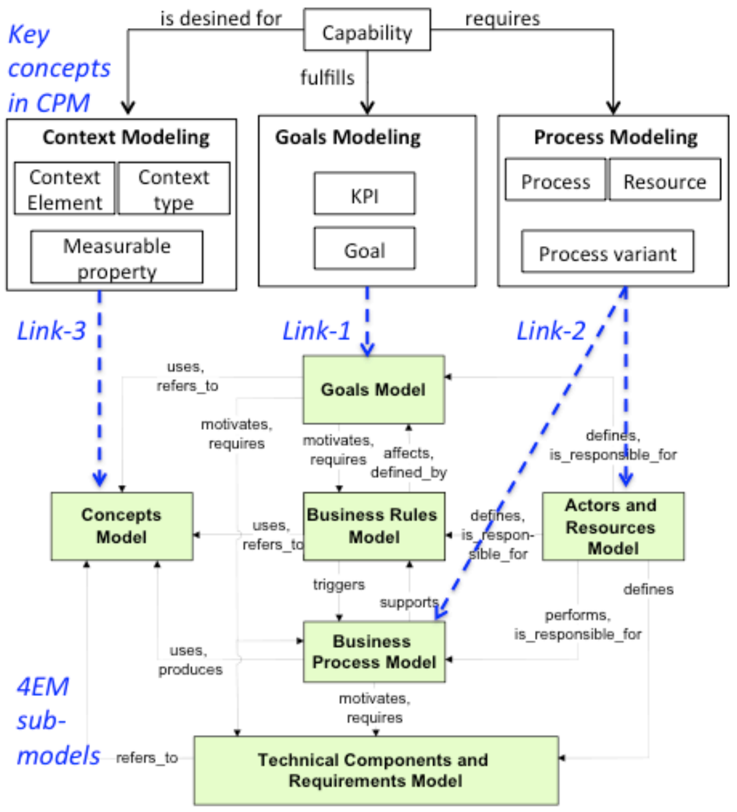

Fig. 4 Alignment of sub-models of capability meta-model and 4EM framework

Link-1: CPM goals and KPIs represent the intentional dimension of capability design and they correspond to the 4EM Goals Model components, namely, goal, problem, opportunity, cause, and relationships, namely, supports, hinders, and AND and OR refinement. Hence 4EM Goals Model was incorporated in CDD.

Link-2: Capability design is specified in terms of business process, process variants and resources, which can be addressed by 4EM Business Process Model and Actors and Resources Model. However, considering that the use case companies were more acquainted with the BPMN and that CDD Environment was developed in the Eclipse

Environment for which there was an available BPMN plug-it it was decided to use BPMN instead, which considerably reduced the implementation costs of the tool.

Link-3: CPM constructs for representing capability context, such as context element, context indicator, and measurable property define properties of things and phenomena, which makes them, in principle, appropriate for modeling with the 4EM Concepts Model. However, analyzing the use case requirements led to a conclusion that a specific modeling guidance is needed and it was decided to develop a dedicated modeling component and a distinct notation for context modeling.

In summary, the following recommendations can be formulated:

— Structure method into components (which and consist of sub-components);
— Consider competence of the method and tool development team;
— Assess the suitability of existing method components; components with similar modeling languages and notations should be considered for the inclusion, components requiring new ways of working might be too difficult to include;
— Consider tool implementation, e.g. available components, ease of use.

6.3 Use of Various Meta-models with Different Purposes

The purpose of the CPM in Fig. 2 is to present the modeling components of CDD and how they are related conceptually. It also includes the main integrity constraints based on association multiplicities, e.g. that each capability is motivated by exactly one goal. This version was extensively used in discussions with the use case partners and within the methodology development team. It was the main reference model for the development of the methodology steps.

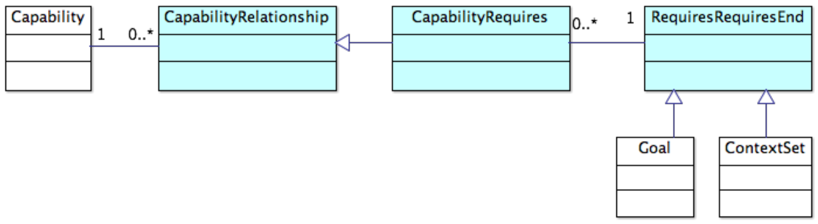

Fig. 5. A fragment of the language meta-model showing Capability relationships with Goal and Context Set

The core meta-model, is however insufficiently detailed for developing a modeling language to the full extent as well as to develop a supporting modeling tool. Hence a *language meta-model* containing detailed components the modeling language was created. Fig.5 shows a fragment of the language meta-model for relationships “Capability fulfills a Goal” and “Capability is designed for Context Set” in the CPM (Fig.2). The relationship names “fulfills” and “is designed for” are changed to “requires”, because it was deemed that the latter reflects the true nature of this relationship more precisely because a capability that is not associated to any goal or any context set would be seen as incompletely designed. The main difference between the language meta-model and core meta-model is that associations and association roles are modeled as classes to specify which association types are permitted between which modeling component types. The language meta-model was developed analytically – by considering the purpose of each component in the CPM and how it could be represented by a modeling language taking the constructs and notation of 4EM a starting point. The resulting meta-model was also useful in discussions between method developer and tool developer teams. It was later extended to represent information needed for other parts of the CDD methodology, such as variables and calculations for adjustment algorithms, which were not part of the modeling language but were needed for capability monitoring at runtime.

The CPM represented integrity and quality constraints assumed to be useful in the CDD methodology, e.g. each capability requires exactly one context set. This, however, does not take into account temporal states of the model, i.e. in an incomplete model, once a capability is placed in a model it will exist without a link to a context set until such a context set is created and an association to it is defined.

The language meta-model essentially served as the reference point for development of the CDD Environment, but it was not useful for conceptual discussions, e.g., when developing the different method components. Referring to MOF levels of meta-models, the language meta-model followed the principles of M2 level, while Ecore (meta-model of the Eclipse Modeling Framework) provided M3 level components.

In summary, the following recommendations can be formulated:

- Develop several meta-models in parallel – core meta-model for discussions and method development and language meta-model for tool development
- Assess integrity constraints and quality criteria built in the meta-model.

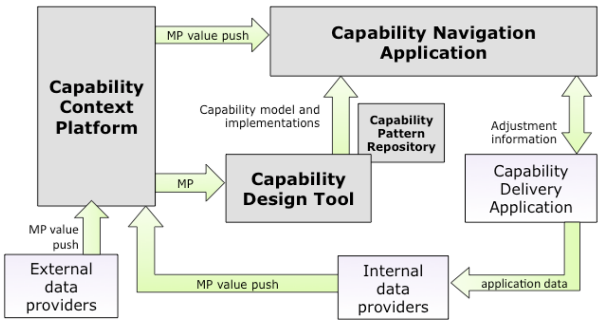
Capability Context Platform
MP value push
Capability Navigation Application
Capability model and implementations
Capability Pattern Repository
Adjustment information
MP
Capability Design Tool
Capability Delivery Application
MP value push
External data providers
MP value push
Internal data providers
application data

Fig. 6. Components of the CDD Environment

6.4 Development of the CDD Environment

The language meta-model was subsequently implemented in the CDD Environment consisting of a number of components (see fig.6.). *Capability Design Tool (CDT)* is an Eclipse based graphical modeling tool for supporting the creation of models according to the capability meta-model. It supports the CPM and its modeling notation. *Capability Navigation Application (CNA)* uses capability models to monitor the capability context by receiving the values of measurable properties (MP) and handle run-time adjustments. CNA manages information at run-time defined according to the meta-model. *Capability Context Platform (CCP)* distributes internal and external context information to the CNA. It aggregates MPs into context elements for models in CDT; it provides runtime values for external context elements (external data providers - Internet, other organizations, individuals); it also allows defining new context elements based on the existing MPs, and specifying KPIs based on MPs for monitoring. *Capability Delivery Application (CDA)* represents the business applications that are used to support the capability delivery. This can be a custom-made or configured, e.g. an ERP, system. The CNA communicates or configures the CDA to adjust for changing contexts during capability design and delivery. It also receives MP values from the data providers internal to the organization. *Capability Pattern Repository (CPR)* stores reusable capability designs. It supports the part of the capability meta-model that is related to patterns and business processes.

The case of developing the CDD Environment differs from the more traditional cases of developing tool support for modeling methods where a modeling language is implemented only in a modeling tool. Because the CDD environment also included other components, for example, the parts of the meta-model related to runtime monitoring and adjustments had to be supported by other components of the CDD environment. Similarly, the CCP was used for monitoring measurable properties and context elements that had to be structured according to the meta-model. To simplify deployment of the CDD environment, a cloud-based version of the environment was also created supporting the final version of the methodology. Virtual instances of CDT, CCP, and CNA are hosted on the common Apache CloudStack platform and CDT was made accessible using web browser by means of desktop virtualization.

In summary, the following recommendations can be formulated:

- Use the language meta-model for tool development
- Include in the meta-model components that are not modeled in a traditional way, such as for runtime data, adjustments
- Consider that the meta-model will be used even outside the modeling tool
- Use cloud based tools and services to support deployment

7 Concluding Remarks

The process of CDD methodology development followed the principles of DSR. The main focus in this paper has been set on the development of the modeling language using meta-modeling with a particular effort on integration with concepts of the 4EM approach and on supporting the development of a modeling tool. A number of experiences and recommendations have also been presented. The CDD methodology and environment have been validated in real life capability design projects at four use case companies as part of design-evaluation cycles of the project. The presented recommendations are by no means exhaustive and more work on collecting such experiences from other ME projects should be devoted.

References

1. EU FP7 CaaS Project. Capability as a Service for digital enterprises, proj. no. 611351 http://caas-project.eu/
2. Zdravkovic J., Stirna J., Grabis J., A Comparative Analysis of Using the Capability Notion for Congruent Business and Information Systems Engineering, Complex Systems Informatics and Modeling Quarterly, CSIMQ, no. 10, pp. 1–20, Available: https://doi.org/10.7250/csimq.2017-10.01 (2017)
3. OPENGROUP TOGAF - enterprise architecture methodology, version 9.1, http://www.opengroup.org/togaf/ (2012)
4. UK Ministry of Defence (2013) NATO Architecture Framework v4.0 Documentation, http://nafdocs.org/modem
5. Bērziša, S. et al. (2014), Deliverable D1.4: Requirements specification for CDD, FP7 proj. 611351 CaaS, http://caas-project.eu/deliverables/
6. Bērziša, S. et al. (2015) Deliverable 5.2: The Initial Version of Capability Driven Development Methodology, FP7 proj. 611351 CaaS, DOI: 10.13140/RG.2.1.2399.4965
7. Sandkuhl K., Stirna J., Persson A., Wißotzki M. (2014): Enterprise Modeling – Tackling Business Challenges with the 4EM Method. Springer, ISBN 978-3-662-43724-7S.
8. Hevner A.R., March S.T., Park J., Ram S. (2004) Design Science in Information Systems Research. MIS Quarterly 28(1): 75-105
9. Brinkkemper S. (1996) Method engineering: engineering of information systems development methods and tools. Inform. Software Tech. 38(4): 275–280
10. Smolander K., OPRR: A Model for Modelling Systems Development Methods, in Next Generation CASE tools, K.Lyytinen and V.-P.Tahvanainen (Eds.), IOS Press, 1991
11. Bergsten P., Bubenko J., Dahl R., Gustafsson M.R., Johansson L.A., RAMATIC - a CASE shell for implementation of specific CASE tools, SISU, Stockholm, 1989
12. Marttiin P., Harmsen F., Rossi M., A Functional Framework for Evaluating Method Engineering Environments: the case of Maestro II/Decamerone and MetaEdit+, University of Jyväskylä, 1996
13. Kelly S. (1997) Towards a Comprehensive MetaCASE and CAME Environment: Conceptual, Architectural, Functional and Usability Advances in MetaEdit+, PhD thesis, University of Jyväskylä, Finland
14. Brinkkemper S., Saeki M., Harmsen F. (1999) Meta-modelling based assembly techniques for situational method engineering. Inform. Syst. 24(3):209–228
15. Henderson-Sellers, B., Ralyté J., Ågerfalk P. J., Rossi M. (2014) Situational Method Engineering. Springer 2014, ISBN 978-3-642-41466-4, pp. 1-274
16. Ralyté J., Backlund P., Kühn H., Jeusfeld M.A. (2006) Method Chunks for Interoperability. ER 2006, Springer LNCS 4215, pp. 339-353
17. Goldkuhl G., Lind M., Seigerroth U. (1998) Method integration: the need for a learning perspective, IEEE Proceedings Software, Vol 145 (4), pp 113-11
18. OMG (2015) OMG Meta Object Facility (MOF) Core Specification, Ver. 2.5
19. Grabis, J., M. Henkel, J. Kampars, H. Koç, K. Sandkuhl, D. Stamer, J. Stirna, F. Valverde, J. Zdravkovic D5.3 The final version of Capability driven development methodology, FP7 proj. 611351 CaaS, DOI: 10.13140/RG.2.2.35862.34889
20. Zdravkovic, J., J. Stirna, J. Grabis, (2017) A Comparative Analysis of Using the Capability Notion for Congruent Business- and Information Systems Engineering, CSIMQ, no. 10, pp. 1–20, Available: https://doi.org/10.7250/csimq.2017-10.01
21. Stirna, J., Grabis, J., Henkel, M., Zdravkovic, J. (2012) Capability Driven Development - an Approach to Support Evolving Organizations. In: PoEM 2012, Springer, pp.117-131
22. Grabis, J., Kampars, J. (2016) Design of capability delivery adjustments, CAiSE 2016 Workshops, Springer LNBIP 249, pp. 52-62