A Role-Based Capability Modeling Approach for Adaptive Information Systems

Hendrik Schön, Jelena Zdravkovic, Janis Stirna, Susanne Strahringer

A Role-based Capability Modeling Approach for Adaptive Information Systems

Hendrik Schön[1], Jelena Zdravkovic[2], Janis Stirna[2], and Susanne Strahringer[1]

[1] Business Informatics, esp. IS in Trade and Industry, TU Dresden, Dresden, Germany
{hendrik.schoen,susanne.strahringer}@tu-dresden.de
[2] Department of Computer and System Sciences, Stockholm University, Stockholm, Sweden
{jelenaz,js}@dsv.su.se

Abstract. Most modeling approaches lack in their ability to cover a full-fledged view of a software system's business requirements, goals, and capabilities and to specify aspects of flexibility and variability. The modeling language Capability Driven Development (CDD) allows modeling capabilities and their relation to the execution context. However, its context-dependency lacks the possibility to define dynamic structural information that may be part of the context: persons, their roles, and the impact of objects that are involved in a particular execution occurrence. To solve this issue, we extended the CDD method with the BROS modeling approach, a role-based structural modeling language that allows the definition of context-dependent and dynamic structure of an information system. In this paper, we propose the integrated combination of the two modeling approaches by extending the CDD meta-model with necessary concepts from BROS. This combination allows for technical development of the information system (BROS) by starting with capability modeling using CDD. We demonstrate the combined meta-model in an example based on a real-world use case. With it, we show the benefits of modeling detailed business requirements regarding context comprising environment- and object-related information.

Keywords: Capability Modeling, Roles, Context, Business Requirements.

1 Introduction

Organizations need a rapid response to changes in the business environment in terms of new legislation, changes in customer and supplier behavior, new and often adverse events. Such change cannot always be foreseen at the time of information system (IS) development and hence the current approach that is based on implementing change by redesigning and redeploying applications is no longer sufficient. A strand of approaches aims at continuous development and tightening the gap between development and operations [1]. This is, however, not suitable for developing and customizing enterprise applications that need to respond to change both on the business and IS level. That is, a congruent approach that supports responsiveness to changes in the application context and facilitates the responses to transcend from the business to the information system is needed. Sandkuhl and Stirna [2] contributed to making IS more

flexible with respect to the adaptation to context. The concept of capability was used for this purpose because it unifies the business aspects traditionally used in areas such as enterprise modeling like goals and processes with execution context [3]. Furthermore, it connects context with the specification of algorithms for adjusting the IS once the context changes. The stance of CDD is that any information that influences the IS is to be modeled as context.

BROS (Business Role-Object Specification) [4] is a structural modeling language for design time specification of business objects concerning a domain model as well as specific business logic. Role-fulfilling objects cover the static specification part regarding the separation of concerns, whereas the dynamic specification part of the business logic is expressed via events. The final BROS model serves as a blueprint for development and can be implemented in role-based modeling languages. BROS supports the specification of system-internal variability that is induced by, e.g., the change of role fulfillment. In case of human or organizational roles, changes of this kind often require adaptations in terms of business process variants because the same roles can be fulfilled by several actors each of which having a different skill profile. This aspect has not been elaborated in the CDD approach. *Therefore, the objective of this paper is to explore the integration of the CDD and BROS for the purpose of supporting role-based capability modeling and IS design.* Among the motivators for the CDD [5] are the following goals, to which the proposed integration of the two approaches is set to contribute:

- *To allocate resources to process execution tasks and to provision human resources to process execution.* The integrated proposal addresses this goal by explicitly modeling skill profiles of actors and skill requirements of roles, which allows specifying the actor-role fulfillment by using the concept of *scene* in BROS.
- *To customize services according to context.* The integrated proposal allows designing and monitoring changes in the context caused by actor-role fulfillment.
- *To monitor process execution.* The CDD approach supports model-driven generation of a monitoring application, Capability Navigation Application (CNA) for overseeing context elements, KPIs, and triggering capability adjustments. The proposal allows integration of actor-role fulfillment and skill monitoring in the CNA.

The remainder of the paper is structured as follows. Section 2 covers the related background of our research. Section 3 provides a conceptual overview of the suggested role-based capability approach. Further, in section 4, the abstract and theoretical part of our research is demonstrated via the introduction of the extended meta-model, a core part of the paper. Section 5 demonstrates our new approach by applying it to a real-world use case, a lecture management scenario in higher education, followed by the conclusion in Section 6 with summary and outlook.

2 Background

The enterprise modeling discipline endeavors to support businesses by means of IS, which imposes supporting some low-volatile business processes and concepts, but

lately even more is required – coping with dynamically changing business environments requiring adaptations of IS at execution time. In this regard, adaptability is seen as an architectural property, enabling a system to efficiently adjust to different or evolving operational or usage circumstances [6,7].

To achieve adaptability, organizations should be able to, by the support of modeling, master different variations of their businesses, such as user preferences, environmental variations, changes on partners' sides, legislations, and other [8]. This study also investigates the area of dynamic adaptions of IS and finds that there is a plethora of capability modeling approaches that depict adaptability elements in different ways. Many of the existing approaches address capability delivery by means of, for example, services, business processes, or actions. Nevertheless, the current state in capability design does not offer a transition to tasks associated with IS development. The CDD approach (section 2.1) relies on enterprise models for designing IS based business capabilities with inbuilt support for adaptation to changing contexts at the execution time [2]. Amongst Enterprise Architecture frameworks and languages, including TOGAF, Archimate, DODAF, NAF and MODAF, the NAF framework [9] is the closest to CDD in its ability to define local conditions in design, but it does not have a method for capability adjustments at run-time. Also, the work of Rodriguez et al. [10] is related to CDD and includes context-dependency as well. However, this approach focuses more on reliability modeling and transformation with replicas at design time. The specifications of the other frameworks provide methods neither for the use of capability at runtime nor for adjustments [3]. However, its methodology and the underlying architecture for designing variability for the purpose of adaptation lack the support for dynamic roles of the entities being involved in the implementation of the capabilities, such as subjects (persons, organizations) and objects. The BROS language (section 2.2) uses business scenarios as a fitting complement to support the specification of system variability induced by the change of role fulfillment.

2.1 Capability-Driven Development (CDD)

The foundation for CDD is provided by the conceptual Capability Meta-Model (CMM). CMM was developed on the basis of industrial requirements and related research on capabilities. In brief, it consists of the three main parts of the meta-model:

- Enterprise model for representing organizational designs with Goals, KPIs, Processes (with concretizations as Process Variants) and Resources;
- Context model for representing for which context a Capability is designed (represented by Context Set) and Context Situation at runtime that is monitored and according to which the deployed solutions are adjusted; and
- 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 Process Variants and Resources are needed to support a Context Set.

The meta-model in Fig. 1 is a simplified version of CMM showing the key components of CDD; the full version with complete element definitions is available in [11].

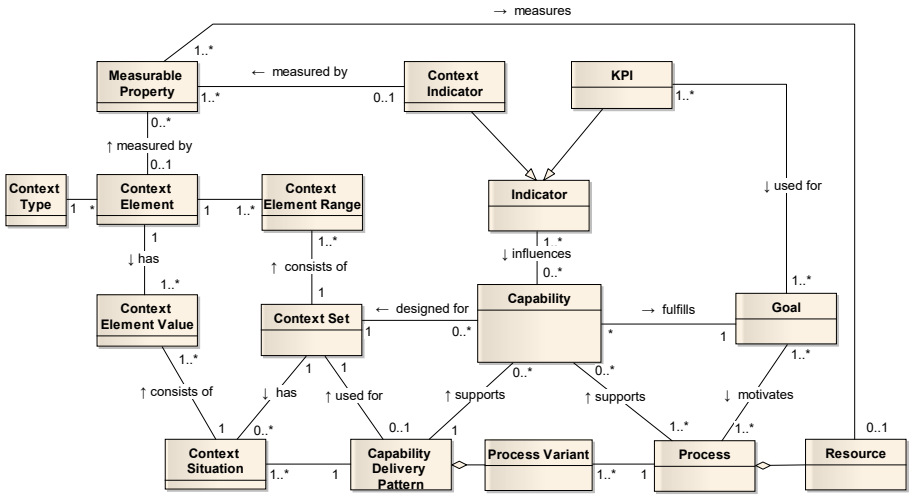

Fig. 1. A conceptual meta-model supporting Capability Driven Development

Table 1. Concepts of the core CDD 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 sets 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. It can be calculated from several Measurable Properties.
Goal	Goal is a desired state of affairs that needs to be attained. Goals can be refined into sub-goals. Goals should typically be expressed in measurable terms such as KPIs.
Process	Process is series of actions that are performed in order to achieve particular result. A Process supports Goals and has input and produces output in terms of information and/or material. A process is perceived to consume resources.
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 using the same input and delivers the same outcome as the Process in a different way.

The CDD methodology combines three interconnected cycles of working – design, delivery, and feedback. Design starts with configuring existing or creating new enterprise goals and processes combined with captured business contexts and eliciting required capabilities. This is followed by delivery of the capability requiring composition and integration of existing technologies and applications, such as ERP systems. During the execution of the application, the changes of context are monitored, and runtime adjustment algorithms are used to calculate if the context's changes require another capability pattern. Feedback is achieved by monitoring defined KPIs, which enable capability refinement and pattern updating.

2.2 Business Role-Object Specification (BROS)

Roles and the related concepts were investigated in various research areas during the last decades (e.g., theories [12], modeling languages [13], programming languages [14,15], runtime environments [16], or enterprise modeling [17,18]). Roles extend the established object-oriented paradigm by the ability to represent an object in different contexts and by changing its behavior and characteristics accordingly. Roles qre described in terms of (a) behavioral, (b) relational, and (c) context-dependent properties [12, 13]. This serves a more accurate description of the domain's entities with their context-dependent structure and behavior. BROS uses this advantage of roles to model software based on required business needs.

The BROS modeling language [4] was originally developed for an easy adaptation of (structural) reference models [19], it can also be used for creating role-based software in general. It utilizes the role-paradigm to specify mainly structural models. Via roles, however, BROS (in contrast to traditional modeling languages such as UML) is able to include the behavior-aware specifications in structural models non-invasively.

BROS does not focus on process modeling itself; it explicitly includes events induced in the respective background processes, nevertheless. Via events, temporality, and role-based context-dependent behavior, BROS allows for behavioral modeling constructs within a mainly structural modeling language. Thus, BROS benefits from CDD due to its ability to define the complex business constraints (i.e., when to choose a scene) as a background source of these events.

The main concepts of BROS are objects, roles, scenes, and events (Fig. 2). Objects are selected from an underlying structural domain model and are the target of any use case or enterprise-specific adaptation done by using the remaining concepts. BROS utilizes roles as specific representations of objects in certain scenes (the role's context). The enterprise-specific processes are the main drivers of the adaptation and serve two kinds of information: (a) the scene as an encapsulation context of a use case or task, and (b) the events as certain points in time affecting the roles. The details of the language, as well as an example, are described in [4], based on the research of

CROM [13]. For the purpose of this research, the BROS concepts were introduced in the CDD meta-model with the knowledge implied by the BROS meta-model.

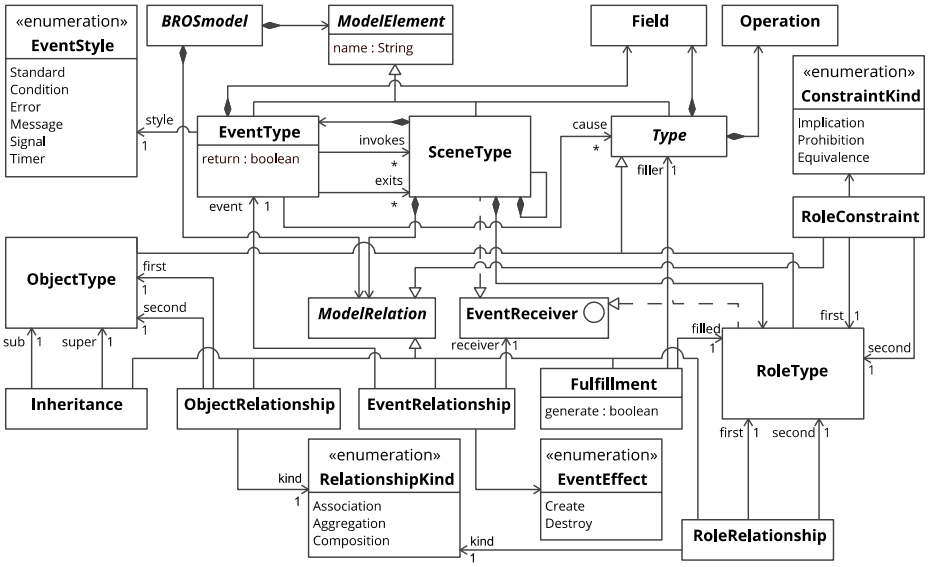

Fig. 2. The basic BROS meta-model [4]

3 Conceptual Approach

According to the motivation for this research, we strive for a framework that extends the CDD approach with the role-based paradigm provided by the BROS approach. Although both approaches are settled on different levels and phases of the software development stack (see Fig. 3), the role concepts introduced by BROS are suitable to be used for fine-grained capability design. CDD and BROS have been developed independently of each other. Nevertheless, they share a common concept of "dedicated context": the process variant in CDD and the scene in BROS. Both are representatives of a special, single task or execution, dependent on the chosen environment.

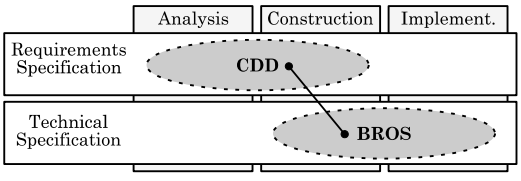

Fig. 3. The connection between both areas

This task may change during runtime since one or more requirements from the given environment is dropped. In CDD, a process variant is derived from a general process description (e.g., giving a lecture). The concrete process variant is then chosen by

variation points based on the environment requirements. Thus, CDD focuses on the conceptual view of the requirements, capabilities, and goals of the respective IS. In contrast, BROS utilizes scenes to describe the behavior of roles for certain tasks or executable procedures. The scene defines the context-dependent boundary of a role's validity (e.g., the role "Teacher" is only valid in the context of the scene "Giving Lecture") in combination with a start and an end as specific points in time.

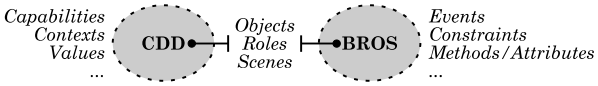

Fig. 4. The set of common concepts

BROS is intended to be an extension to CDD, hence, we integrated its concepts (see Fig. 4) into the already existing CDD methodology. We state that, with the BROS concepts, it is possible, to describe the capabilities of an enterprise with regard to performers that are able to play certain roles (or not). CDD, as presented in [2], is able to model the capability dependence on static environment information (e.g., resource utilization, calendar time or the weather condition), while including BROS enables the modeling of capabilities that depend on the participating performers (that is, actors and objects with abilities), illustrated in Fig. 5.

Technically, our proposal is realized as a meta-model extension to the CDD meta-model. Extending the meta-model also allows maintaining the adaptation and decision mechanisms of CDD. Thus, we strived for a non-invasive adaptation to implement the BROS features for two reasons: (a) to use CDD as a new source of business knowledge usable in BROS, and (b) to include the structural modeling concepts (scenes, roles, and objects) into CDD to provide a more powerful modeling approach.

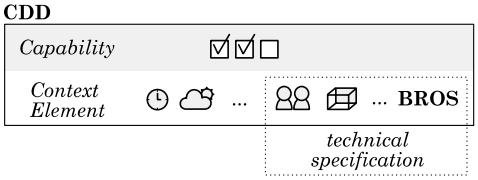

Fig. 5. Adding BROS to CDD

4 Meta-model Extension

This section introduces the full (extended) meta-model, describes the respective model elements as well as their relationships and purposes.

To achieve the envisioned integration of CDD and BROS and keep the existing CDD method components and method extensions intact, the meta-model has been extended "non-invasively" (c.f. [2] for more information about CDD methods components). The CDD-BROS integration is intended as a method extension. For this purpose, we extended the complete CDD meta-model with a set of new meta-model elements that override or extend already defined elements. As a result, the CDD meta-

model ensures that the new extension is compatible with the CDD environment and other CDD extensions. The meta-model depicting CDD with the BROS extension is shown in Fig. 6. The set of BROS elements contains the newly developed elements. While *ProcessVariant* and *ProcessVariantVariationPoint* are overridden (i.e., marked as abstract) and not usable together with the BROS extension, the *ContextElement* and *MeasurableProperty* are extended and can be used simultaneously with the related BROS elements. Apart from the inherited elements, several new elements are used to model the BROS part of the combined approach.

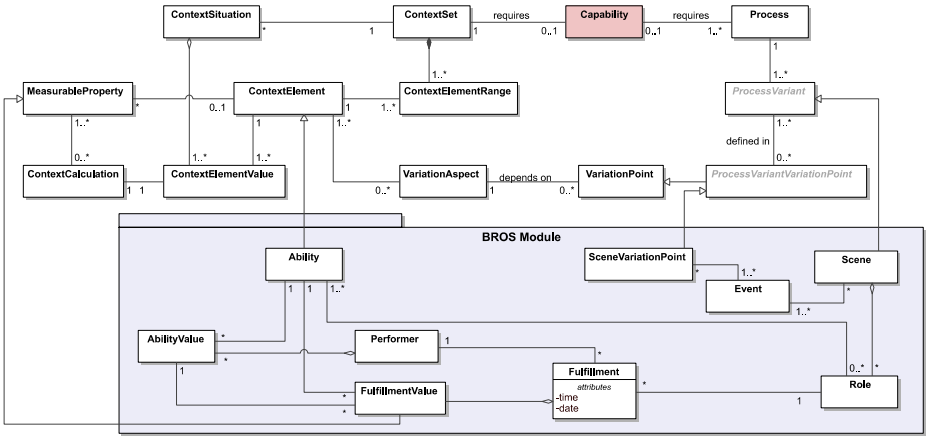

Fig. 6. The small CDD meta-model with the BROS extension

The newly introduced BROS elements are responsible for dedicated context decisions based on provided skills by entities. For that reason, we use the semantics of roles, objects, and scenes established in BROS.

A *Performer* is an entity (type) of the real world that is allowed to take over a *Role* in a *Scene* via *Fulfillment* to provide certain *Abilities*. For example, the performer "professor" is able to fulfill the role "teacher" in the scene "giving lecture" to achieve the ability (and responsibility) of "teaching". This is also possible for non-human objects like, e.g., the entity "room" that may fulfill the role "lecture hall." However, the performer and its roles depend on the use case that has to be modeled. With the meta-model extension, it is stated that the scene (a BROS concept) replaces the *ProcessVariant*. Thus, the process defined by CDD now uses scenes to describe different flows dependent on the chosen context. Scenes contain roles that are determined for a specific temporal execution (e.g., the "giving lecture" scene requires the role "teacher"). The variation mechanisms in CDD then do not point to process variants but specific scenes with roles. The roles are fulfilled by performers (real-world entities). The fulfillment between a performer and a role is annotated with the fulfillment value that quantifies the ability of the performer to fulfill a specific role. E.g., various instances of a performer "professor" fulfill the "lecturer" role in the scene "giving lecture" with their own specific *FulfillmentValue* in term of a skill profile. This value is inherited from *MeasurableProperty*, i.e., the fulfillment value is given by, e.g., a database in

the ERP system that lists the employees with their skill profiles and may be time-dependent. There must be at least one role (provided by a scene) that is responsible for providing the necessary abilities. However, at runtime, the concrete *AbilityValue* for the abilities is derived via *ContextCalculation* from the fulfillment value, i.e., dependent on the entity that takes over the specific role. Since the ability is inherited from *ContextElement*, the *ContextElementRange* (from CDD) is assigned to the abilities to limit the possible value range for the context. At runtime, those are concrete value boundaries. If the range is violated (due to not fitting ability values) the adaptation part of CDD uses the *SceneVariationPoint* to define another scene that is able to be used for the new context. However, this paper does not focus on the adaptation part, which is defined in the full CDD meta-model. In the proposed meta-model, we use *Events* since the BROS scene definition includes, inter alia, a start and end via an event. Thus, we use an event as an interface from the scene variation point towards the scene. This allows the start of multiple scenes with triggering a single start event.

The new meta-model elements are listed in Table 2. The M1 level is used for capability design, e.g., specifying that a lecturing capability is based on performers such as professors and roles such as teachers, students, and course assistants. The M0 level of a capability model materializes once the lecturing capability is executed and runtime-specific professors, e.g., "John" and "Alice", perform specific roles for specific scenes. An M1 instantiation example for the new elements in this meta-model is given in the next section. Due to the non-invasive changes to the original meta-model, all mechanisms of CDD, like the capability adjustment algorithms and calculations of KPIs and context, are still operational.

Table 2. New meta-model elements within the BROS method component

Concept	Description	M1 Example	M0 Example
Performer	A real-world entity on type-level that is able to do something	Person, Room, Computer	Alice, INF003
Role	A context-specific behavior that may be adopted by a performer	Attendant, Teacher, Lecture Hall	Alice's Teacher role, INF003's Lecture Hall role
Scene	A contextual boundary that denotes a temporal execution	Giving Lecture, Checking Exams	Lecture ID 5
Fulfillment	The process of a performer playing a role	Employee-Teacher-Fulfillment	Alice playing the Teacher
Fulfillment Value	A runtime value that is related to the profile of a role.	--	Profile of room INF003 when fulfilling Lecture Hall
Event	A type of point in time when something may happen	Start, End, Interruption	9am at 24. Dec 2019, Incoming Call ID 42

Scene Variation Point	A mechanism that decides the triggering of events to start a certain scene	--	--
Ability	An action related to the possibilities of a performer	Heating up, Having capacity, Teaching	Teaching ability of Alice
Ability Value	A runtime value that denotes the quantity of a certain ability	--	42, 1337, yes

5 Use Case – Provisioning of Subjects in Higher Education

This section demonstrates the proposed CDD and BROS integration with an example case of the teaching environment at a large university in Sweden.

5.1 Use Case Description

The provisioning of the subjects in Higher Education requires substantial planning and effort. That includes organizing the lecturers' team, scheduling, admission of students, and publishing course materials. Once a course starts, the major activities are teaching sessions, exercises, supervision, and examinations.

Requirements Engineering is a standard subject offered at both the undergraduate and graduate levels to about 250 students in total. The course is given in Swedish at the undergraduate level, and in English at the graduate level. The team of teachers includes several roles: lectures and Q&A seminars are given by professors; exercises are supervised by teaching assistants, PhD students, and professors; tool tutoring and supervision is done in the computer labs and led by teaching assistants, PhD students, and research assistants. The course material includes lectures, tutorials, reading material, and media. It is published on the Moodle online education portal. The platform is managed by the whole teaching team according to the assigned roles and responsibilities. During the course execution, the portal is also used for managing communication among the students and the teachers, management of quizzes, grading of exercises, as well as other activities. Since we investigate a Swedish university, there is the possibility of a sudden and severe snowfall in colder seasons. Hence, the local traffic information system and the weather forecast are analyzed for possible general delays. If severe delays throughout the city are to be expected or are occurring, the course events might be cancelled, rescheduled, repeated, and/or switched to online delivery.

Concerning course scheduling, each classroom has a limit for the maximum number of persons. Because the classrooms are a resource constraint, they need to be booked well in advance. If the number of students exceeds the size of the classroom, it is possible to stream a lecture to another classroom in real-time. This, however, poses additional tasks related to the management of the teaching process. It is not easy to re-schedule the rooms in cases when more students than estimated register for the

course (the deadline is the day when the course starts), as well as when additional tutoring (and thereby rooms) becomes needed. The final exam is classroom-based and as such requires a sufficient number of places and invigilators for each of the examination rooms, which requires engaging both the teachers as well as additional staff.

5.2 Meta-model Instance Design

According the defined schema of model layers by Object Management Group [20, 21], the meta-model is on the M2 layer. An instance of this layer is the M1 layer, which is a model that uses the M2 defined concepts to specify the targeted "real model." An instance of M1 is on the layer M0, which represents "real items" like "Alice" as a person or "INF003" as a room. However, since the CDD and BROS extensions are on M2, we need to define the capability design on the M1 layer before considering runtime items. However, not all M2 concepts need to be instantiated on the M1 level because concepts that denote runtime concepts, like values of context elements or ability values, belong to M0 (i.e., fulfillment, fulfillment value, ability value, and event). For these values, we model the M1 pendants as a type that needs to be expressed at runtime. Thus, as elaborating the M0 is not our primary goal, we do not go into detail of their runtime assignments.

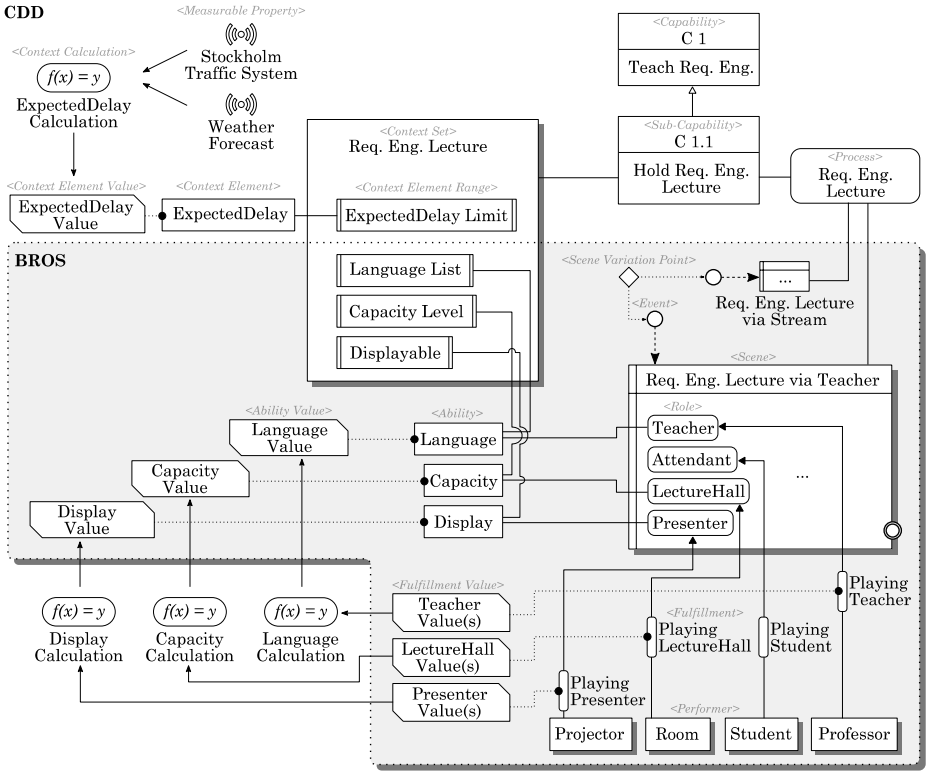

Fig. 7. An example instance of the meta-model for the use case

Fig. 7 shows the M1 instance of the M2 meta-model, including the CDD part (white background) and the new BROS concepts (gray background). We derived this instance example from the use case as only one of many different possibilities. The respective M0 types from the meta-model are annotated to the M1 model elements.

The use case requirements are encoded in the context set named "Req. Eng. Lecture", which serves the overall CDD capability of "Teach Req. Eng." The ranges (specified on M0 with concrete range boundaries later on) set the parameters of deciding between different scenes. For this use case, there are several of such ranges as parameters: a specific schedule delay, a required teaching language, a device that is able to display the slides, and so forth. If something happens, e.g., there is too little capacity available in the room for the lecture, the adaptation part of CDD (not shown in the meta-model) adapts towards this new situation with triggering changes. By using BROS, the capacity of a lecture room is modeled in Fig. 7 as an ability of a role that is fulfilled by a performer, i.e., any room (e.g., "INF003") that plays the role of the lecture hall at runtime so that its capacity is used for the lecture's capacity.

If at runtime this capacity is outside the range set in the capability design, the adaptation part has two options:

1. It uses a different performer (that is, a new room with higher capacity) that fulfills the role in the same scene so that the scene does not change; or
2. If there is no other performer available, the scene is switched to another scene (e.g., a scene that streams the lecture to various locations) that meets the range requirements with possibly other roles (e.g., a stream receiving device).

If neither of the two possibilities can be applied, then an error occurs since there is no available solution to the new context. The real-world entities, the performers, have to fulfill the roles in a scene, i.e., the CDD environment is able to perform calculations deriving the ability value out of their profile since the real room "INF003" at runtime does not know which abilities one wants to derive (e.g., its capacity or its ability to be ready for exams). The CDD environment delivers the actual runtime value for the ability (e.g., "15" for capacity) that gets checked against the range boundaries, which can be Boolean, lists, formal expressions or simple number ranges (e.g., "1 to 20"). The context set may also contain ranges that are not dependent on entities but on environmental states. In Fig. 7, we modeled the traffic situation and the weather forecast as measurable properties, so that the calculation results in the value of expected delay. This is checked against the range in the context set to decide whether it is possible to hold a lecture or whether one should start streaming (or skip the lecture). This expected delay is an environment-based state and independent from any concrete performers and roles (for demonstration purposes on how to model BROS-independent context elements). Thus, when designing capabilities, one has to decide between environment- or entity-based context elements and their ranges. Regarding the modeling complexity, we only designed a simple CDD-BROS model for one capability with limited scene-based variability. There are plenty of options to extend this design, e.g., multiple abilities or performers per role, performers that fulfill a set of roles in certain scenes, involving different IT supporting tools for teaching and other variants.

5.3 Use Case Discussion

With the modeled use case stated in Fig. 7, we argue that the modeling of entity-based context elements receives more attention in the capability design. Previously, every context element was handled as external. Thus, the modeling of capabilities is enhanced due to additional modeling constructs:

- The construct of scene allows the definition of concrete variations of an executable, providing a set of necessary roles;
- Roles (encapsulated in scenes) enable modeling the necessary entity-based context elements (i.e., abilities of performers);
- Performers are the main constructs to define the concrete entities that are responsible for fulfilling a context element range (indirectly via roles).

This trinity of the role-based BROS paradigm (scene, roles, and performers) is the tool for switching between contexts and related situations at runtime. When encountering an unmet range condition, the new possibility, to switch between fulfillments instead of switching to a whole new scene, is an important improvement. As such, the same context scene may be continued with only changing the performers, who are fulfilling the needed roles and their abilities for the scene.

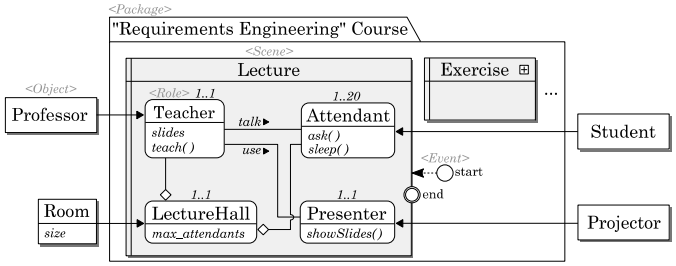

Fig. 8. An example BROS model with objects, roles, events, and a scene

The combination of CDD and BROS allows modeling on multiple levels, i.e., subsequent stages along the model-driven development lifecycle (as shown in Fig. 5). With the usage of the BROS concepts in CDD, the transition towards implementation is simplified. BROS, as intended to be on the technical level, is tightly connected to the underlying software development. Assuming the fact that the CDD modeling would define several scenes, roles, and performers for its contexts, these can also be used for a related BROS model. Fig. 8 shows a possible BROS model for software construction that is derived from the CDD model in Fig. 7 with the same scene, its roles, and the related performers. This interrelation of the different abstraction levels while developing IS via CDD and BROS benefits model-driven development.

6 Conclusion

We have shown two existing approaches, CDD and BROS, which were developed independently and for different purposes, that have be combined to complement each other. This was done by integrating the needed BROS elements into the CDD meta-model as an extension. The combined approach was demonstrated with a use case and allowed for a more realistic and fine-grained modeling of an enterprise's capabilities.

The introduction of entity-based abilities, related to roles, grasp the nature of capability responsibility and liability. Thus the CDD-BROS integration contributes to an enterprise modeling approach that can be used for early business-focused modeling as well as for later specification of technical details in a seamless manner. The different facets of adaptation and variation are covered through the combined approach encompassing adaptations at runtime due to resource allocation via performers. This, in general, supports role-based capability modeling and IS design as stated in Section 1.

Comparing this overall contribution to Enterprise Architecture approaches, we conclude that the suggested combination supports modeling on the level of detail that is needed for seamless IS development even encompassing runtime aspects like accounting for performers. However, in comparison to MDA-like approaches, which are (by nature) aligned to seamless integration along the development lifecycle, our suggestion is stronger when it comes to early capability driven modeling and context-dependent adaptations. One limitation of our current work is that we cannot ensure that other methodological enhancements in the enterprise architecture, enterprise modeling, or MDA domain may have achieved comparable goals. Also, the CDD-BROS-integration still needs to be fully implemented and supported with tools. This will require that the BROS extension to the CDD meta-model is implemented as a CDD method component. In the realm of a full-fledged integration, examples at the level of complexity of our use case could then be used to demonstrate the full potential of using CDD with BROS – not only at the modeling level but also within the accordingly developed IS.

Acknowledgements

This work is partially funded by the German Research Foundation (DFG) within the Research Training Group „Role-based Software Infrastructures for continuous-context-sensitive Systems" (GRK 1907).

References

1. Hüttermann, M.: DevOps for Developers. Apress. 10.1007/978-1-4302-4570-4 (2012).
2. Sandkuhl, K., Stirna, J.: Capability Management in Digital Enterprises. Springer International Publishing, Cham (2018).
3. 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. pp. 1–20 (2017).

4. Schön, H., Strahringer, S., Furrer, F.J., Kühn, T.: Business Role-Object Specification: A Language for Behavior-aware Structural Modeling of Business Objects. In: Proceedings of the 14th International Conference on Wirtschaftsinformatik. Siegen, Germany (2019).
5. Bērziša, S., Bravos, G., González Cardona, T., Czubayko, U., España, S., Grabis, J., Henkel, M., Jokste, L., Kampars, J., Koç, H., Kuhr, J.-C., Llorca, C., Loucopoulos, P., Juanes Pascual, R., Sandkuhl, K., Simic, H., Stirna, J., Zdravkovic, J.: Deliverable 1.4: Requirements specification for CDD, CaaS–capability as a service for digital enterprises. Riga Technical University (2013).
6. Morin, B., Barais, O., Jezequel, J.-M., Fleurey, F., Solberg, A.: Models@Run.time to Support Dynamic Adaptation. Computer 42, pp. 44–51 (2009).
7. Engel, A., Browning, T. R., Reich, Y.: Designing Products for Adaptability: Insights from Four Industrial Cases. In: Decis. Sci. 48(5), pp. 875–917 (2017)
8. Koutsopoulos G., Henkel M., Stirna J.: Dynamic Adaptation of Capabilities: Exploring Meta-model Diversity. In: Enterprise, Business-Process and Information Systems Modeling. pp. 181–195. Springer, Rome, Italy (2019)
9. North Atlantic Treaty Organization: NATO Architecture Framework v4. North Atlantic Treaty Organization (NATO) (2019).
10. Rodrigues, G.N., Roberts, G., Emmerich, W.: Reliability Support for the Model Driven Architecture. In: de Lemos, R., Gacek, C., and Romanovsky, A. (eds.) Architecting Dependable Systems II. pp. 79–98. Springer Berlin Heidelberg (2004).
11. Grabis, J., Henkel, M., Kampars, J., Koç, H., Sandkuhl, K., Stirna, J., Valverde, F., Zdravkovic, J.: Deliverable 5.3: The Final Version of Capability Driven Development Methodology (2016).
12. Steimann, F.: On the representation of roles in object-oriented and conceptual modelling. Data Knowl. Eng. 35, 83–106 (2000).
13. Kühn, T., Leuthäuser, M., Götz, S.: A Metamodel Family for Role-Based Modeling and Programming Languages. In: Combemale, B., Pearce, D.J., Barais, O., and Vinju, J.J. (eds.) Software Language Engineering. pp. 141–160. Springer (2014).
14. Herrmann, S.: Programming with roles in ObjectTeams/Java. In: Proceedings of the 2005 AAAI Fall Symposium (2005).
15. Leuthäuser, M.: A Pure Embedding of Roles, http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-227624 (2017).
16. Taing, N., Springer, T., Cardozo, N., Schill, A.: A Dynamic Instance Binding Mechanism Supporting Run-time Variability of Role-based Software Systems. In: Companion Proceedings of the 15th Intern. Conf. on Modularity. pp. 137–142. ACM (2016).
17. Almeida, J.P.A., Guizzardi, G., Santos, P.S.Jr.: Applying and extending a semantic foundation for role-related concepts in enterprise modelling. In: Proceedings of the 12th IEEE Intern. Enterprise Distributed Object Computing Conf., EDOC. pp. 31–40. IEEE (2009).
18. Frank, U.: Delegation: An important concept for the appropriate design of object models. J. Object Oriented Program. 13, pp. 13–17 (2000).
19. Schön, H.: Role-based Adaptation of Domain Reference Models: Suggestion of a Novel Approach. In: Drews, P., Funk, B., Niemeyer, P., and Xie, L. (eds.) Tagungsband Multikonferenz Wirtschaftsinformatik 2018. pp. 1447–1453. Leuphana (2018).
20. Object Management Group: Meta Object Facility (MOF) Core Specification v2.5.1. Object Management Group (2016).
21. Atkinson, C., Kuhne, T.: Model-driven development: a metamodeling foundation. IEEE Softw. 20, 36–41 (2003).