Instances over Algorithms: A Different Approach to Business Process Modeling

Stefan Hofer

Instances Over Algorithms: A Different Approach to Business Process Modeling

Stefan Hofer

University of Hamburg, Department of Informatics, Software Engineering Group
and
C1 WPS GmbH

Abstract. Most of today's approaches to business process modeling interpret processes as algorithms. As a result, modelers often try to cover every possible outcome of a process. This algorithmic view increases the amount of information that has to be captured and reviewed. As an alternative, we propose to focus on exemplary process instances instead. We call this concept *Exemplary Modeling* and have built a modeling method around it. In this paper we explain its merits and illustrate its feasibility by reporting on our practical experience.

Key words: business process modeling, requirements engineering, instance modeling, CSCW

1 Introduction

No matter if you optimize business processes, analyze requirements for the development of a new software system or aim to improve IT-business alignment: Business process models are commonly used to capture the necessary information to achieve these goals. Usually, business process modeling can be divided into two major activities:

- Modelers gather and condense information into graphical models.
- Domain experts review and approve the models.

These steps become increasingly difficult with the amount of information that has to be considered. The more information they comprise, the larger and more complex the models become. Also, the total number of models typically increases with the amount of information to capture. In professional environments modelers and domain experts have to cope with this situation. Thus they could draw significant support from concepts that address gathering of extensive information, condensing it into models, and reviewing these models.

In this paper we describe a concept which provides that support – *exemplary modeling*:

- We will describe how modeling of exemplary instances of processes differs from classical approaches. We will point out how the concept helps modelers to gather and condense information and domain experts to review and approve models (see Sect. 2).

- We demonstrate the applicability of exemplary modeling by presenting a modeling method that is based on this concept. The method was developed cooperatively at the University of Hamburg and C1 WPS GmbH (see Sect. 3).
- We reflect on the concept by sharing our experience gained by using the modeling method in commercial projects (see Sect. 4). Drawbacks and limitations will also be discussed.
- A comparison of the concept with related work concludes this article.

2 Exemplary Modeling

2.1 Motivation

Being involved with business process modeling in academic and professional contexts, we noticed how hard it is to get domain experts to participate in modeling. Shipman and McCall provide a possible explanation:

> "The difficulties that users have in formalizing information are not just interface problems. More effort is required of users in part because formal representations require the explicit statement of information that might have been left implicit [...] in a less formal representation. In addition, substantial effort is typically required to learn a formal representation. This requires both talent and interest in formalization that users are unlikely to have." [13, p. 286]

We compared this situation with a similar problem domain – requirements engineering, and particularly user participation during software development. Our software engineering process is heavily influenced by the *Tools and Materials Approach* [16]. This approach fosters the use of prototypes and pilot system to incorporate user feedback into the development process. Prototypes are very tangible and allow for the discussion of concrete workflows and boundary conditions. Thus, they provide a proper basis for discussion about the system.

In our opinion, most approaches to business process modeling lack such a foundation because models are usually rather abstract and algorithmic descriptions of processes. Typically they are produced by decomposition. Processes are broken down into subprocesses until the level of activities is reached. Graphically, they are depicted as a sequence of boxes and arrows – much like diagrams used in *Structured Analyses and Design Technique* [11] or similar engineering techniques. Two prominent examples of this kind of business modeling are the *Business Process Modeling Notation* (*BPMN*) and *Event-driven Process Chains* (*EPC*) (for both see [15]). In fact, EPCs were originally conceived as a modeling technique for software engineers, not for business process modelers and domain experts. The underlying mindset aims at describing precisely how a process is executed. Thus, activities and decisions are modeled in an "algorithmic" way; a way suitable for automating processes with software. However, the behavior of interactive systems such as typical business applications cannot be reduced to algorithms as Wegner explains in [14]. Jacobson et al. criticize that algorithmic modeling techniques are used for business processes:

> "All these techniques come from the computer world. It is as though we learned to think in a way that works for computer systems, and we realized we could apply the same way of thinking to describe an organization [...]. We believe information systems should be described so they are easy for people to understand, with abstractions that people can comprehend. We think it is bizarre to apply the way of thinking that governs computer systems to business process."[6, p. 36]

A main feature of algorithmic modeling techniques is that case switches (called *gateways* in BPMN [2]) like "exclusive or" are an important element of the models. Each switch increases the number of paths from start point to end point. Therefore, a model displays all these possible paths – a graphical representation of the process's transitive hull of states and state transitions. But from a domain expert's view, different paths may represent very different outcomes. That means that their graphical proximity in the model does not necessarily correspond to their relationship in the real world. In a similar way, case switches do not express that some paths are more important or more likely than others.

Additionally, the use of case switches is suggestive of depicting *complete* models, meaning that every possible path is covered. This blends in with the underlying logic of algorithmic modeling techniques: The execution of an incomplete algorithm would result in an error or an undefined state. In our experience, this ambition for completeness leads either to an increase in the number of models or the use of more case switches. In fact, completeness is unachievable for interactive business applications: "Incompleteness is a necessary price for modeling independent domains of discourse whose semantic properties are richer than the syntactic notation by which they are modeled [...]." [14, p. 98]

In conclusion, the common practice of modeling business processes with methods that have an "algorithmic" nature does provide plenty of room for improvement.

2.2 Exemplary Modeling as a Different Approach to Modeling

Our approach to business process modeling is based on the idea that concrete instances of a process are easier to understand than abstract descriptions of all possible outcomes. These exemplary instances of a process can be modeled without case switches – every model represents one path from process start to end. As a consequence, modelers and domain experts do not have to deal with the complete set of possible states that a process may have during its execution. Consider this example:

A sales process for an insurance takes different paths depending on the client being already customer of the insurance company or not. In an usual process model, this would be modeled as a "exclusive or" switch, splitting the process model into two paths[1] (see Fig. 1). With our approach, we would create one model for each significant path through the model (so, two for this example – see Fig. 2).

[1] Omitting the possibly negative outcome of the second switch for the sake of brevity.

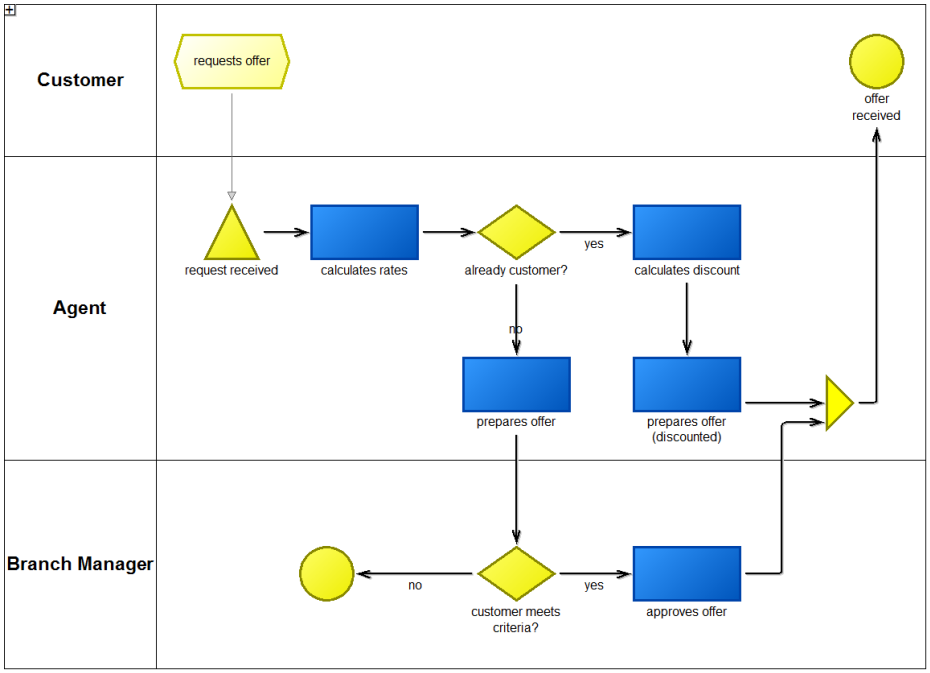


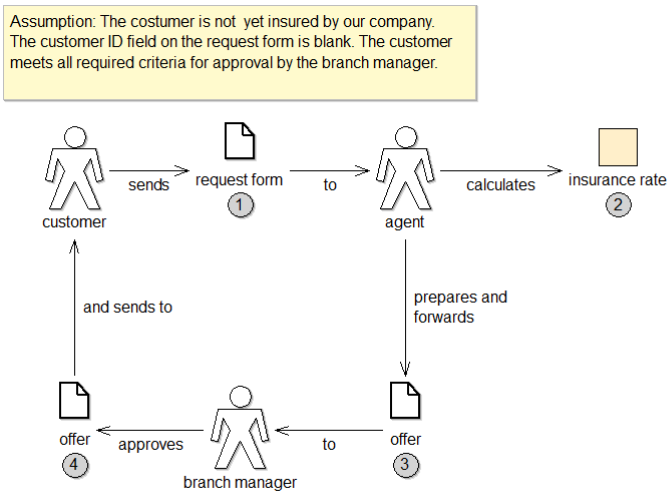
Fig. 1: An insurance sales process modeled with BPMN.

Every exemplary process instance is considered a *scenario*. In [4], John M. Carroll describes scenarios as follows:

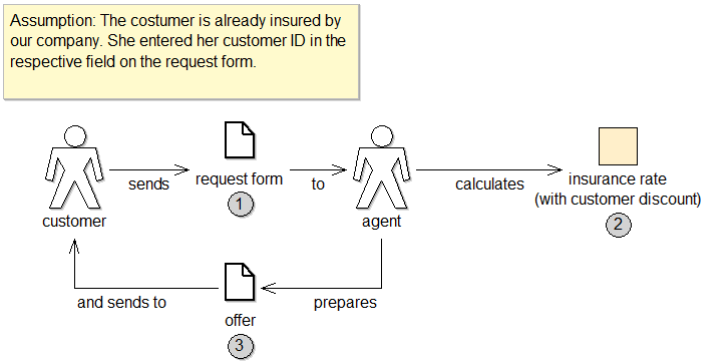
- "Scenarios are stories – stories about people and their activities." [4, p.46]
- Scenarios "mention or presuppose a setting", meaning they are set in a specific context. [4, p.47]
- "Scenarios include agents or actors [...] each typically with goals or objectives."[4, p.47]
- "Scenarios have a plot; they include sequences of actions and events" [4, p.47]

In contrary to other modeling approaches, a scenario-oriented approach does not describe actions in an algorithmic way. Instead, it focuses on the actors (human or IT systems) that are involved and order their activities in an exemplary sequence. That way, activities and their IT support can be discussed without consideration of an algorithm that could automate these activities.

Obviously, modeling every unique path of a process may take would increase the number of models dramatically. However, in our experience it is usually sufficient to look at some characteristic paths that a process may take. This is why the number of models for each process can be limited by focusing on the most important scenarios. Smaller variations and optional steps can easily be annotated in textual form. Reconsider the previous example:



(a) New Customer.



(b) Existing Customer.

Fig. 2: Exemplary insurance sales process models.

Let us assume that a new customer may be treated like an existing customer if his or her spouse is already an existing customer. The only difference is that some additional fields have to be filled out on the application form. Instead of creating additional models for this variation, a textual annotation is sufficient to incorporate it into the insurance sales process for existing customers. Figure 3 depicts this variation of Fig. 2b.

3 The Exemplary Business Process Modeling Method

Exemplary modeling is merely a concept. To turn it into a modeling method for professional use, one needs:

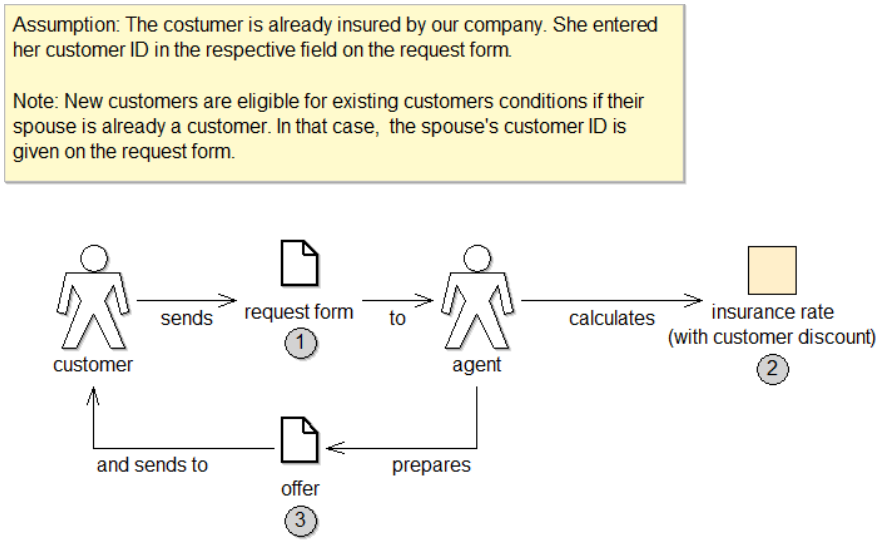


Fig. 3: Small variations are covered by annotation rather than separate models.

– a modeling notation,
– a modeling process, and
– a modeling tool.

This section describes how these were built around the idea of exemplary modeling. The result was a tool-supported method called *exemplary Business Process Modeling (eBPM)* (see [3]).

3.1 Cooperation Scenarios

The eBPM notation consists of several types of diagrams with *cooperation scenarios* being the most important one. It becomes obvious that cooperation scenarios implement the concept of exemplary modeling when looking at some of the graphical elements that can be used in this diagram type:

– *Actors* are people or IT systems that perform actions. They may interact with each other by exchanging Information (see *Work Objects*).
– *Work Objects* can take the form of documents, data, tools and so on.
– *Relationships* express interaction of actors, creation or editing of things, use of IT systems and so forth.

Consequentially, there are no case switches available. It should also be noted that modelers can choose whether they use actors to represent a concrete actor (e.g. “Mr. Smith”) or a role (e.g. “agent”).

eBPM’s cooperation scenarios were derived from a diagram type called *cooperation pictures*, proposed by Krabbel et al. in [8]. Cooperation pictures were

created as a requirements engineering method to capture cooperation. Thus both diagrams have a strong focus on depicting cooperation. For example, they both use pictograms to create a more tangible view on cooperation.

Compared to cooperation pictures, cooperation scenarios describe a business processes as a sequence of steps. However, in contrast to other methods like BPMN this sequence is not depicted as a control flow. Instead, an order is established by numbering the steps (see Fig. 3). A more detailed look on cooperation scenarios shows that every step has to be carried out by an human or a software system. To do that, an actor performs an activity which is modeled as relation between actor and a work object. Activities are depicted as arrows and annotated with verbs. Thus, every step can easily be transcribed in textual form in a subject-predicate-object pattern. As a result, cooperation scenarios can be "read" like stories – just like suggested by Carroll (cf. Sect. 2.2).

Although the notation is not the focus of this paper, we find it noteworthy that its tangibility complements exemplary modeling. As stated above, work objects are depicted as icons that match the real world objects they represent: A document icon symbolizes a document, a telephone icon symbolizes communication via telephone, ...

The cooperation scenario and the other diagram types of eBPM are embedded in a meta model. Other important diagrams include:

- *Model of Terms* for collecting and structuring work objects.
- *Model of Roles* for collecting and structuring roles that actors can assume.
- *IT-Landscape* for collecting and structuring IT systems.

Since these diagram types are not necessarily required to comprehend the idea of exemplary modeling, we will not explain them any further in this article. Additional information can be found in [3].

3.2 Modeling Process

Typically, eBPM models are developed in workshops. Participants include modelers, domain experts and, if needed, technically oriented experts. During the workshop, the model is visible for all participants (e.g. projected on a screen). Since only concrete cases are modeled, the resulting models can easily be validated and agreed upon by the participants. Thus, the workshop covers both of the modeling steps described in Sect. 1: Information gathering and condensing as well as reviewing.

To ensure the models are scenario-based, the modeler has to guide the modeling process. She does so by discussing the boundary conditions of the exemplary processes and includes the description into the graphical model. The examples in Fig. 2 each contain such a description in a yellow textbox. Since every modeled action is performed by an actor, the modeling process advances by asking questions like:

- Who performs this action?
- What things does the actor use for that step?

3.3 Tool Support

Although eBPM may be used with a flip chart or generic drawing tools like Microsoft's Visio, tool support was implemented using the BOC Group's *Ado* modeling platform. A specific software tool offers various advantages:

- References between models and/or model elements allow for a hypertext-like navigation.
- The enumerated actions of a model can be visualized one step at a time. This "walkthrough" feature helps to read large models by telling a "visual story".
- Models can be evaluated by certain criteria – for example, which IT systems are used by which actors.

A free version (without any restriction to non-commercial use) is available via the University of Vienna's *Open Model Initiative* website (see [9])[2].

4 Reflection

In this section, we reflect upon the use of the eBPM method in general and upon the concept of exemplary modeling in particular. To do so, we begin with an overview of the fields where eBPM was used successfully. Then, we compare our experience with the initial claim that exemplary modeling supports modelers and domain experts to perform their modeling activities (see Sect. 1).

4.1 Field of Application

eBPM has been in use for several years now in so different domains as banking, insurance, federal government, logistics, and health care. It has mainly been adopted to support requirements engineering rather than for "classical" business process modeling. The following list presents some fields in which the method has been used so far:

Software Development. Current business processes and their IT support were modeled using eBPM. From this starting point, the models for the to-be processes were derived. These models had a strong focus on how the new systems should support the user's activities.

Evaluation of *Commercial Off-the-shelf* (*COTS*) Software. Organizations that wanted to buy COTS software used eBPM to describe how IT support for their business processes looked like ideally. These models were used in the acquisition process. The vendors of possibly fitting COTS software had to describe how the modeled activities would look like if their product was used. This way, it became evident for the organization how well each possible solution fitted and how they compared against each other.

[2] So far, the software is available in German language only.

Bringing Software Into Service. Complex software solutions are often introduced into organizations incrementally. eBPM was used to model how each step affects the business processes. Also, the use of interim solutions (like adapters between systems or organizational workarounds) were planned with help of eBPM models. Thus, the transition to the new software system was smoothened.

Quality Assurance. Software migration projects require a quality assurance process that helps ensuring the success of the changes. eBPM was used to describe core business processes that were of crucial importance to the organizations. Then, to-be models of the processes were derived to describe the planned outcome of the migration. In a third step, these descriptions were turned into process-oriented test cases.

Organizational Changes. Organizations that wanted to change their structure used eBPM to describe their current work places. After analyzing the models for improvement or reorganization, eBPM was used to outline how the to-be work places should look like and how IT could support them.

4.2 Experience

Comprehension. In our experience, eBPM models are easy to understand, regardless of one's background or domain. An indication of that ease of use is that we do not explain the eBPM notation to the participants of modeling workshops. These workshops can be successful even if the modeler is the only one of the participants who has experience with the method. However, we usually have to explain the exemplary character of the models. Especially participants who used other modeling approaches before are used to distinguishing cases.

Concreteness of Scenarios. Although a cooperation picture depicts an instance of a process, modelers have some freedom in choosing its level of concreteness. As described in Sect. 3.1, actors may represent individuals or roles. We recommend the latter as often several representatives of a role participate in a modeling workshop. Modeling roles as actors allows all representatives to identify with the actor. Simultaneously, it helps them to abstract from little details that differ from individual to individual.

Variations. The power of scenario based modeling becomes evident when participants are constrained to separate important from less important variations of business processes, thus facilitating information gathering as well as reviewing of models. However, the pragmatic approach of handling small, but not negligible variations as textual annotations comes with a pitfall. It may lead to lots of descriptive text accompanied by a weak model instead of a rather rigorous model with some additional textual comments.

Responsibilities of Modelers. Naturally, the concept of exemplary modeling and the freedom of choice that eBPM provides call for a skilled modeler. We learned from training prospective eBPM modelers that the usefulness of models increases dramatically with experience. That experience is of particular importance when conducting a modeling workshop (see Sect. 3.2). For example, the modeler has to guide the process of finding a sufficient amount of scenarios for a given purpose.

Feedback Loop. Notation, modeling process and tool support allow for "live modeling". Breitling et al. conclude: " The modeler immediately translates the contributions of the participants into eBPM models that are visible to everyone. This procedure shortens the feedback loop significantly." [3, p. 189] This effect is amplified by the "walkthrough" feature provided by the modeling tool (see Sect. 3.3).

We noticed that approving of models is supported especially well by exemplary modeling. Domain experts can easily dismiss incorrect models because of their concreteness. It is also interesting to see that domain experts identify with the actors in the model. They recognize their work objects and the actors they interact with. This too helps them to evaluate process models.

Notation. Both the symbols of the modeling language as well as the layout have a profound impact on model comprehension [12]. Hence, it is difficult to distinguish to what extent the applicability of eBPM can be attributed to its notation, to the concept of exemplary modeling, and to the skills of the modeler.

Limitations. Processes that mainly consist of complex state transitions proved to be difficult to capture solely with exemplary models. Again, we found it useful to focus on a small number of typical chains of transitions so that one could gain an idea of the general purpose of the process. But in addition we used state transition diagrams to clarify the lifecycle of work objects. This combination was valuable when dealing with processes that have a legal dimension like for example dunning processes of insurance companies.

It should be noted that all this experience was gained in industries that rely on both people and IT systems to execute their business processes (see Sect. 4.1 for fields of application). eBPM was neither developed as a language for process automation nor as modeling method for embedded systems or similar areas.

In summary, we noticed that exemplary modeling does indeed support modelers and domain experts in their respective modeling activities.

5 Related Work

5.1 Instance Modeling

Since exemplary modeling deals with instances of business processes a discussion of *instance modeling* seems appropriate. Within the area of information

modeling, there are several approaches that propose modeling instances as an alternative or an extension to some sort of a "class" model or schema. For example, Agarwal et al. suggest modeling schemas that describe the building blocks and relations that can be used to define business processes (see [1]). Although the resulting process models are instances of the process schema in terms of object-orientation, they are not scenarios. Thus, they are not instances in terms of exemplary modeling.

A different instance modeling approach proposed by Parsons and Wand in [10] aims for a semantically richer database design by making classification of data instances optional (i.e. allowing decoupling from a relational database schema). Their design consists of two layers of models, one for instances and one for classes. In a somewhat comparable way, cooperation pictures (containing "instances") may refer to models of terms and models of roles (see Sect. 3.1). The latter model types can be used to define structured collections of work objects and actor's roles, respectively.

5.2 Use Cases

Finally, we want to address a well known modeling approach that is also scenario based: *Use Cases*. Since there are several variants of use cases (see for example [7]) we will limit this discussion to Cockburn's prominent version as described in [5].

Apparently, use cases are text-based whereas eBPM is a graphical modeling method. Although there are so-called *Use Case Diagrams* in UML, Cockburn makes clear that they are "not a notation for capturing use cases"[5, p. 128]. Use cases and eBPM both center around actors and use scenarios to describe business processes. Yet, there are differences in focus. In [5], Cockburn distinguishes three levels of use cases:

- *User Goals* correspond to elementary business processes.
- "*Summary-level* goals involve multiple user goals. [...] They show the context in which the user goals operate. They show life-cycle sequencing of related goals. They provide a table of contents for the lower-level use cases [...]." [5, p. 64]
- "*Subfunction-level* goals are those required to carry out user goals." [5, p. 66]

We do not make this distinction as exemplary modeling works on all three levels. However, eBPM is not just based on exemplary modeling but also on modeling cooperation. Thus, we usually use eBPM to capture a level between user goals and summary-level goals. In summary, eBPM has a strong focus on how several actors work together to achieve a goal.

Still, the similarities between the two approaches may indicate that eBPM is merely a graphical addition to use cases. However, this paper only covers one of several of eBPM's model types, neglecting valuable modeling capabilities that are not related to use cases (see Sect. 3.1).

All in all, it should not be surprising that we have incorporated eBPM in use cases several times. Cooperation pictures are well suited as a use case's "main

success scenario" [5, p. 28]. Extensions and variations of use cases blend well with the way we deal with variations in exemplary modeling (see Sect. 2).

6 Summary

In summary, we think that an algorithmic view on process modeling adds to the problem of how to handle extensive information:

- The quest for completeness increases the amount of information that needs to be gathered.
- Subsequently, more information needs to be condensed into models. This leads to either more complex models and/or an increased number of models, organized as hierarchies of processes and sub-processes.
- Due to their abstract nature und their extensiveness, these models are usually hard to review and approve.

In this paper, we proposed a different approach to process modeling. This approach is based on the idea that it is easier to capture and evaluate exemplary instances of processes rather than an abstract description of all their possible outcomes. We explained how scenarios provide a proper basis for exemplary modeling. By introducing our Exemplary Business Process Modeling Method (eBPM) we showed that the concept of exemplary modeling is suitable for a professional modeling technique. This was illustrated by presenting several fields of application that eBPM has proven successful in. Using exemplary modeling in these fields of applications allowed us to gain experience and reflect on the concept. We observed, that exemplary modeling (as implemented by eBPM) indeed helped to avert the disadvantages we experienced with other modeling techniques. Finally, we compared our approach to Use Cases and pointed out how these two approaches can be combined.

References

1. Agarwal, R., Bruno, G., Torchiano, M.: Instance Modelling – Beyond Object-Oriented Modelling. In: Proceedings of the 3rd International Conference on Information Technology (2000)
2. Object Management Group: BPMN Graphical Elements. http://www.bpmn.org/Samples/Elements/Gateways.htm (last visited 2011-06-30)
3. Breitling, H., Kornstädt, A., Sauer, J.: Design Rationale in Exemplary Business Process Modeling. In: Dutoit, A., McCall, R., Mistrik, I., Paech, B. (eds.) Rationale Management in Software Engineering. pp. 191–208. Springer, Heidelberg (2006)
4. Carroll, J.: Making Use. Scenario-Based Design of Human-Computer Interaction. MIT Press (2000)
5. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley Longman, Amsterdam (2000)
6. Jacobson, I., Ericsson, M., Jacobson, A.: The Object Advantage: Business Process Reengineering with Object Technology. Addison-Wesley Longman, Amsterdam (1995)

7. Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley (1992)
8. Krabbel, A., Wetzel, I., Ratuski, S.: Participation of Heterogeneous User Groups: Providing an Integrated Hospital Information System. In: Proceedings of the Participatory Design Conference PDC 96. pp. 241–249 (1996)
9. Open Models Initiative Website: http://www.openmodels.at/web/bpm (last visited 2011-06-30)
10. Parsons, J., Wand, Y.: Emancipating Instances from the Tyranny of Classes in Information Modeling. In: ACM Transactions on Database Systems, Vol. 25, No. 2, pp. 228–268 (2000)
11. Ross, D. T., Schoman, K. E.: Structured Analysis for Requirements Definition. In: IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, pp. 6–15 (1977)
12. Schrepfer, S., Wolf, J., Mendling, J., Reijers, H.: The Impact of Secondary Notation on Process Model Understanding. In: Lecture Notes in Business Information Processing, Vol. 39, pp. 161–175 (2009)
13. Shipman, F. M., McCall, R. J.: Supporting Knowledge-Base Evolution With Incremental Formalization. In: Proceedings of the ACM Human Factors in Computing Systems Conference 1994. pp. 285–291(1994)
14. Wegner, P.: Why Interaction is More Powerful Than Algorithms. In: Communications of the ACM, Vol. 40, No. 5, pp. 80–91 (1997)
15. Weske, M.: Business Process Management. Springer, Berlin (2007)
16. Züllighoven, H.: Object-Oriented Construction Handbook. dpunkt.verlag, Heidelberg (2005)