9 Design Rationale in Exemplary Business Process Modeling

H. Breitling, A. Kornstädt, J. Sauer

Abstract: Exemplary Business Process Modeling (EBPM) is an efficient approach to object-oriented software application design. With the help of EBPM, a substantial amount of information about business processes and work practice in the application domain can be gathered and connected to the design and usage model of the software system under scrutiny. Although EBPM was not originally conceived for this purpose, the experience which we share in this article suggests that EBPM should be a standard method for building a basis of knowledge from which design rationale can be gathered.

Keywords: design rationale in software engineering; exemplary business process modeling; cooperation scenario; scenario-based design; object orientation

9.1 Overview of Exemplary Business Process Modeling

The Exemplary Business Process Modeling (EBPM) approach is a scenario-based method that encompasses models and methodologies for the analysis and design of business processes and the software that supports them. It can be employed to model present as well as future processes.

EBPM was developed cooperatively at C1 WPS and Hamburg University in the context of the Tools and Materials approach and its application-oriented document types [19] – significantly influenced by Züllighoven. It was hedged when it occurred to us that neither texts nor standard UML diagrams alone are sufficient to discuss questions of work routine and software systems with users. Building on Krabbel's and Wetzel's Cooperation diagrams we added element after element and finally tool support on the basis of BOC's Adonis modeling tool.

The main benefit of EBPM is the comprehensibility of its models and its suitability for both software designers and domain experts. Therefore, it can be used in workshops and modeling sessions with participants from technically-oriented and domain-oriented groups. Its focus is on modeling exemplary, concrete scenarios without case differentiation.

Being exemplary instead of exhaustive and capturing knowledge about current and future business processes instead of decisions, EBPM is not a design rationale methodology. Instead, EBPM is a valuable auxiliary methodology on which methodologies that capture design rationale can

foot. We have employed EBPM for several years to capture current and future processes in banking, insurance and logistics projects, often in the context of migrating between software systems. Depending on the cardinality of process groups under scrutiny, well over 100 EBPM models were furnished per project.

9.2 The EBPM Paradigm

We capture design rationale in order to provide a sound basis for deciding about how to design new or how to evolve existing software systems. To this end, we document decisions during the software development process together with problems, influencing variables, alternatives (also the discarded ones), arguments, and discussions. It has been shown that this information is extremely valuable when it comes to explain the system to new members of the development team [9].

A couple of methodologies for capturing design rationale have been developed (see [4][Chap. 1 in this book] for an introduction). In these interpretations of design rationale management, participants aim at recording *every* decision including its underlying decision base (*every* option (discarded or not), *every* influencing variable, and *every* discussion). These exhaustive forms of rationale management require substantial – sometimes prohibitive – investments of resources (see Sect. 1.5.1 for a discussion of the capture problem).

Experience from our projects suggests that in many cases it is not viable to manage design rationale information to that extent – especially when it comes to recording decisions which concern design alternatives that have been discarded and do not become a part of the shipped software product. Instead we follow suggestions brought forward by Dutoit and Paech [5] to closely integrate requirements engineering and capturing design rationale. Therefore, we aim at striking a balance between the need for – potentially quite costly – design rationale information and limiting ourselves to the information that we find to be essential when it comes to make design decisions.

But what rationale information is essential? We found that as in requirements engineering, the users' work context is the ultimate source for justifying the system's design. It is concepts and processes from that source that are responsible for the vast majority of requests for adding new or changing existing features. As EBPM was devised for capturing the users' work context complete with processes, work objects, and underlying concepts, it is ideally suited to form the basis for design rationale

extraction, especially in the context of customized software development. It is these aspects of EBPM that we focus on in this contribution.

Regarding the categorization given in [4][Chap. 1 in this book], EBPM is a prescriptive approach in the sense that its models' basic structure is given by a meta model that has to be used and that guides the thinking of designers. EBPM aims to be less intrusive by granting the designers some flexibility in adopting this meta model for their specific needs.

9.2.1 EBPM as Scenario-Based Method

The term scenario was established in informatics at the latest at the beginning of the nineties. In [1] (pp. 46–47), Carroll describes them as follows:

- Scenarios are stories – about persons and their actions
- Scenarios are set in a specific context
- Scenarios contain agents or actors that have goals that they follow
- Scenarios have a plot; they consist of a sequence of actions and events
- The scenario's plot is supposed to be supported at least in part with the help of a software application

In EBPM, scenarios are represented graphically with

- Icons for actors, business objects and other artifacts
- Arrows for actions and
- Memo sheets for context information

See Fig. 9.1 for an example.

Scenarios are modeled in three layers with different model types: C*ooperation Scenarios* focus on the interaction between several actors, W*orkplace Scenarios* focus on the actions that an individual actor carries out at his workplace alone and *IT Interaction Scenarios* describe the interaction of an individual actor with a single application system or a group of related systems. In every scenario type, the flow of action is clearly discernible.

Artifacts from these models can be associated with a *Model of Terms* that relates business terms with another and can give explanations for them in the form of a glossary. Different roles of actors can be described in a *model of roles*.

If a substantial number of cooperation scenarios needs to be modeled, *business use case diagrams* are added to provide a graphical overview.

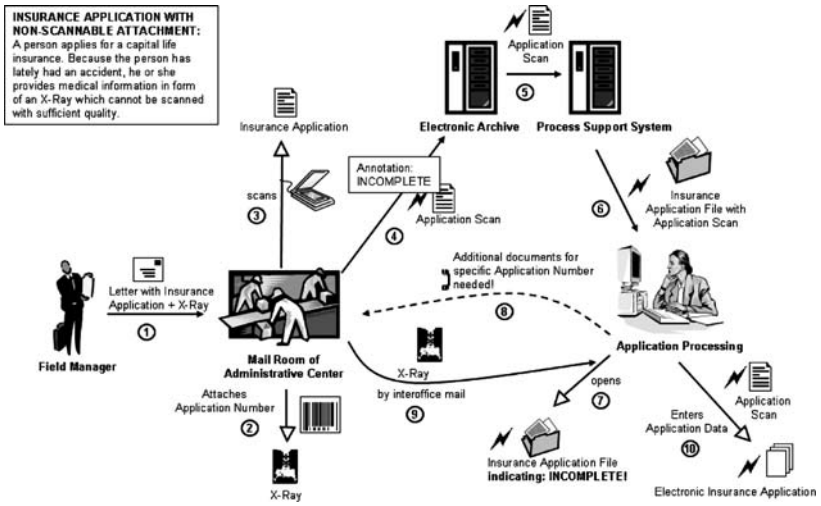

Fig. 9.1. Example cooperation scenario

A business process is typically modeled with EBPM in two to four related scenarios. As already said, scenarios depict one relevant, exemplary sequence of events without case differentiation. This greatly improves the comprehensibility of the models. If there are important variants, then these are depicted in scenarios of their own. Minor variants can be annotated in textual form with notes that are added to certain steps of scenarios.

9.2.2 Modeling Workshops

Sometimes we develop EBPM models with only one or two participants in direct interviews. Most often though, we hold modeling workshops with many participants from technically oriented and domain oriented groups.

The processes can be treated from different angles in these workshops. It has been constantly shown in our projects that the participants from different group can equally contribute their point of view and their knowledge. We have found that EBPM models are comprehensible for participants with different backgrounds and serve well as a common basis and design platform. This is mainly because the models capture domain specific actors and their action together with the related artifacts in a simple, graspable notation.

The models are created during the workshops. We often use a setting with a moderator who is in charge of the discussion and a modeler at a laptop connected to a video projector. The modeler immediately translates the

contributions of the participants into EBPM models that are visible to everyone. This procedure shortens the feedback loop significantly.

9.2.3 Tool Support for EBPM

Even though the EBPM can be used with a flip chart, pen, and paper, software tool support is essential for its efficient use. A standard tool for graphical modeling (like MS Visio) can be used, but a specific software tool offers several advantages like stepwise visualization of scenarios, queries over a large quantity of models and navigation between models in a hypertext style. Such a tool enables the efficient, direct creation of models during modeling sessions.

We are using a tool that offers many more possibilities, e.g., the attachment of arbitrary files to all model elements or the automatic numbering of steps. This facilitates the modelers' work a lot and offers expanded possibilities. All models are filed in a centralized repository to promote coordination in the design team.

9.3 EBPM Models

In this section, we will present the main models of EBPM and their interconnections.

9.3.1 Cooperation Scenario

The starting point of EBPM is the Cooperation Scenario. Figure 9.2 shows the basic underlying meta model. A visual language is used to represent a specific cooperative work scenario. A fundamental part of the scenario is its background story, a short text that explicitly describes the story's context and states the domain-related assumptions made for the model. It needs to be reasonably strict in order to motivate the specific scenario and exclude alternative courses of action.

The story that is being told is divided into consecutive, numbered steps. Every step is performed by an actor. In a Cooperation Scenario step, an actor can basically do two things: either cooperate with another actor – by just communicating, or else transferring a work object to her or him for further processing – or inspect and/or modify a work object on her or his own. Each action type is visually represented by a different type of arrow.

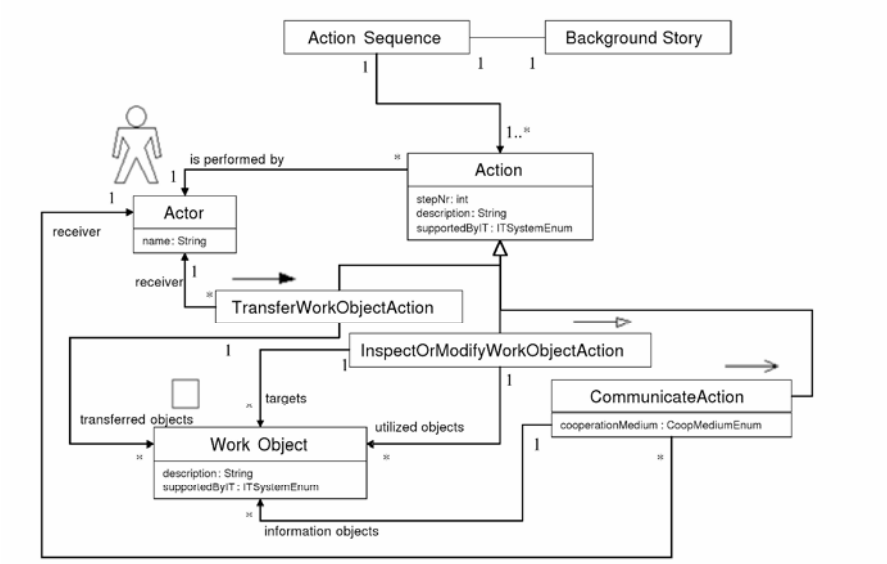

Fig. 9.2. Basic meta model of the Cooperation Scenario. For clarity, the corresponding EBPM symbols are located next to each box

Work objects are first level model elements of their own. A lightning bolt symbol indicates that the work object is not physically present but exits in an IT system only. The steps of the Cooperation Scenario are to be read in sequence according to the numbers on the action arrows. Thus, the Cooperation scenario is a story in pictures, resembling storyboards or comic strips. This is illustrated in Fig. 9.3 that starts showing step 1 only and then incrementally adds steps.

The steps of the Cooperation Scenario correspond to elementary sentences in natural language. For example, step 2 in Fig 9.3 reads as: “The Mail Room (team) uses a bar code sticker to attach an application number to the X-Ray”. The actor performing the step takes the role of the subject, the action itself is represented as a verb, the involved other actor and the work objects become objects in the grammatical sense.

Also note that this closeness to natural language sentences requires that work objects are duplicated for every step. This means that there is not only one instance of a work object but it appears as many times as it is used in an action. For example, the scanned application form occurs several times in Figs. 9.1 and 9.3.

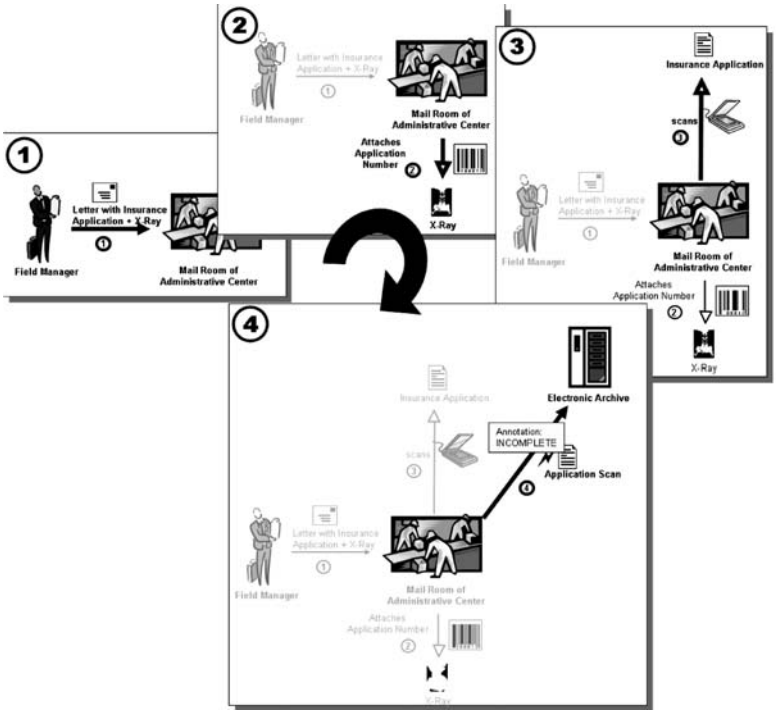

Fig. 9.3. The Cooperation Scenario as a story in pictures

9.3.2 Workplace Scenario

When a cooperation scenario becomes crowded with too many actions, they become less readable. Most of the time, not all actions have to deal with cooperation but take place at one individual actor's work place instead. In order to unclutter the cooperation scenarios, actions that are confined to one workplace only, can be "folded" into a Workplace Scenario that acts as a sub-model of the main Cooperation Scenario (see Fig. 9.4 for an example). A Cooperation Scenario references each of its Workplace Scenarios as one single step of the entire sequence. The Workplace Scenarios for a specific actor are visualized as numbered items over a desk symbol that is shown next to the actor. This way, Workplace Scenarios can be "stepped over" when examining the big picture of the Cooperation Scenario (see Fig. 9.4).

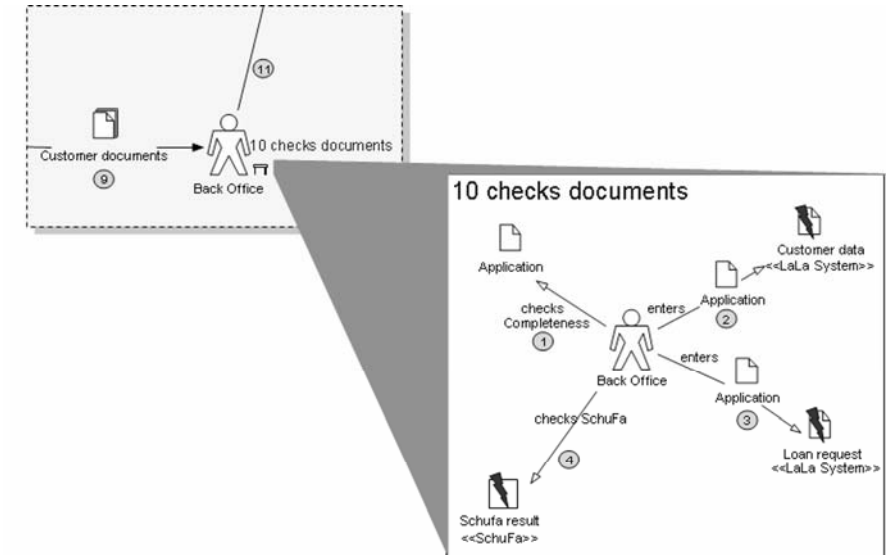

Fig. 9.4. Expanding a workplace step to a Workplace Scenario. Lightning bolts indicated electronic documents. The presence of a workplace scenario is indicated by the desk symbol next to the actor in the *top left* figure

9.3.3 Model of Terms

The Model of Terms contains the relevant terms for the domain concepts of one or more Cooperation as well as Workplace Scenarios and relates them to each other. The work objects in the scenario models are instances of the concepts in the Model of Terms.

The elements of the Model of Terms can be related via the "is-a" association, "is-part-of" association or a weak, untyped association type whose instances can be augmented with free text. Elements in the Model of Terms can be stereotyped as containers such as folders. Those are connected with other objects using the "contains"-association. Furthermore, there are IT-inspired stereotypes Tool and Service (see details given later).

Although the Model of Terms of EBPM can be a starting point for IT system design, it is important to emphasize that it cannot naively be mapped to a UML Class Model or to an ER model. It is strictly domain-oriented and does not define classes, class operations or the cardinality of relations. Consequently, transformation from the Model of Terms to UML's Class Model cannot be automated.

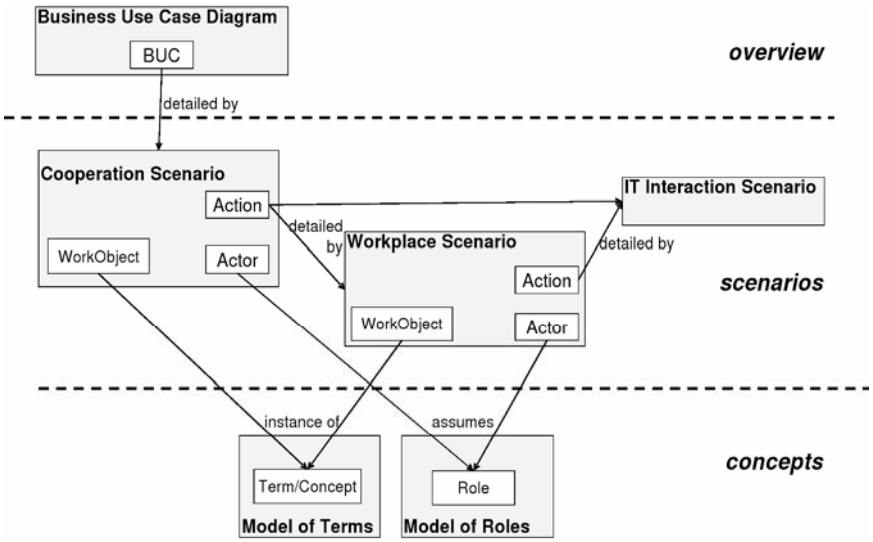

Fig. 9.5. Metamodel overview

9.3.4 Other Diagram Types

While the three diagram types discussed so far are invariably used when modeling business processes, diagrams of the following three types are only used when the specific need arises. We will not describe meta models for these types here, but show how they relate to others in Fig. 9.5.

- Models of Roles can be used in complex organizations to indicate which roles certain actors can take. It liberates modelers from using individual actors only.
- IT Interaction Scenarios depict one actor interacting with one or more IT systems. These scenarios resemble UML's sequence diagrams except that the communication shown is between the actor and the system, not inside a system
- Business Use Case Diagrams provide an overview of several Cooperation Scenarios. The Business Use Case bubbles in the diagram reference the Cooperation Scenario models

9.4 Capturing Design Rationale

In this section, we will elucidate how an application context captured with the various kinds of EBPM models can help software professionals to make rational design decisions about: the business object class model, the

relational database model, tool features, candidates for services, the front end technology, the communication infrastructure and the usage model aspects that are related to the work processes in the application context.

At this point it has to be stressed that in all of the following cases, references to attributes, methods, and class hierarchies only pertain to those features that directly correspond to elements in the application domain. No software system can be constructed by referring to the application domain alone. Most classes and class hierarchies will therefore have to have technology-based attributes and classes such as classes for persistence and graphic display. However, these technological aspects only come into play in design stages which are outside the domain of EBPM.

We find that the problems of maintaining design models and linking them to EBPMs are not worth the effort. Instead, we use EBPMs as a mere light-weight approach for gathering requirements as a basis for extracting design rationale. Based on these models, we usually build prototypes of low complexity (PowerPoint mocks, executable GUI mocks, or prototypes with a GUI and very limited functionality) to discuss our understanding of the application domain and our solution approach with users. Based on these feedback cycles, we either modify EBPMs or refine our prototypes.

9.4.1 Business Object Class Model

The most straightforward way of deriving a design decision from an EBPM model is to take the *Models of Terms* as input for a class model of the application's business objects. These models comprise a wealth of information that corresponds to features of class models:

- Each work object in a *Model of Terms* is a good candidate for a *business object class*. This is in accordance with the underlying principle of object-oriented design – namely that objects in the system correspond to objects in the application domain. This principle of responsibility driven design (see [18]) follows the original purpose of the first object-oriented programming language "Simula" [3], namely simulation. Depending on the scope of a project, a certain number of work objects might not be considered parts of the application.
- A *business class's methods* can be derived from a work object's textual description as well as from the way it is referenced in Cooperation and Workplace Scenarios.
- For design purposes, it is highly useful to use the links in the opposite direction, namely to go from a *Model of Terms* to *Cooperation* or *Workplace Scenarios*, i.e., to find out in what way a work object is used in which scenarios.

- In conjunction with information taken from annotations, the way a work object is used – and thus its methods – can be taken from its usage in *Cooperation* and *Workplace Scenarios*.
- A *business class's attributes* can be derived from its methods and descriptive text plus from the aggregations indicated in *Models of Terms*. While aggregation information can be inferred directly just by looking at the aggregates relationship in a *Model of Terms*, finding attributes cannot be done visually but requires an examination of the methods (see above) and from the descriptive text: if there are methods that augment a work object in some way and others that query its status, then is obvious that there needs to be an attribute that holds that specific piece of information. Information about additional attributes or their type might be obtained from a format description stored in the annotations, e.g., whether an integer or a String is more appropriate to represent document IDs.
- A *business class's position in a class hierarchy* can be taken from its position in *is-a* relations in *Models of Terms*.

9.4.2 Relational Database Model

By exploiting the same features as in the section "Business object class model," elements of relational database model can be derived from EBPMs. Whereas strictly object-oriented features such as methods and class hierarchies cannot be transferred from EBPMs to a database model in a meaningful way, the other features can be derived analogous to those in Sect. 9.4.1:

- Each work object in a *Model of Terms* is a good candidate for an *entity* because these are the things that users work with. Even if it becomes obvious at later design stages that some work objects do not warrant introducing separate entities, using work objects as starting points for entities is a good idea.
- A *business class's attributes* can be derived from its methods and descriptive text plus from the aggregations indicated in *Models of Terms* (see above).
- While a *business class's position in a class hierarchy* cannot be directly transferred into a relational database model, it is a good indicator that the work objects are closely connected: they represent similar concepts on different levels of concreteness and feature at least some common attributes. This is an indicator that these work objects might be mapped to a single entity.

9.4.3 Tool Features

On a general level, every application can be regarded as a tool that allows users to accomplish a certain number of tasks with the help of an IT system. It is in accordance with this view to classify a calculator, a word processor or an ERP as tools. Nevertheless, in the context of modeling business processes, considering a whole ERP as a tool would be too coarse-grained. We therefore limit our definition too those tools which (1) typically have a main and sometimes some subwindows/dialogs and (2) which serve to view/manipulate business objects.

Using EBPM, the use of tools can be modeled explicitly or implicitly. Explicit modeling means that a tool appears as a specialized work object in *Cooperation Scenarios*, *Workplace Scenarios,* and *Models of Terms*. In this case a tool's features can be found by taking advantage of the link mechanism described in "Business object class model: Each link from a *Model of Terms* to *Cooperation* and *Workplace Scenarios* is followed and the accumulated ways of access are a tool's feature list. Implicit modeling means that a tool is not present yet but that there is high degree of similar access to a number of work objects. These similar ways of accessing a work object are indicative of potential tools.

Regardless of the way of modeling tools – which is usually a mixture of explicit and implicit modeling – other tool features can be gleaned from the EBPMs:

- *Tool modes* can be found be examining how tool use differs from user type to user type or from scenario to scenario. For example, some users might just use basic functionality while specialist users call upon similar functionality but require a higher degree of detail or additional kinds of information.
- A *tool's versatility* can be derived by looking at how many work object types are handled with a specific tool. Usage of just a single work object suggests a highly specialized tool while usage on many different work object types hints at a quite generic tool such as a browser or spreadsheet-like tool.

9.4.4 Candidates for Services

Identifying candidates for services requires that tools have already been found. On that basis, tools that make use of the same work object type in similar ways in several *Cooperation* and *Workplace Scenarios* indicate that the tools might be implemented by using the same service that provide access to business objects.

9.4.5 Workspace Types and Front End Technology

In general, (1) the number of tools employed, (2) their modes, and (3) their usage frequency provide important information about the workplace a user requires in a certain scenario. By looking at all scenarios that a user participates in, one can determine the degree of complexity, flexibility, and efficiency he or she requires. Based on those parameters, an informed choice of the appropriate front end technology, infrastructure, and usability-related criteria can be made. This naturally extends to the choice of general communication infrastructure including hardware: for example, employment of barcode scanners in a work process would receive attention in an EBPM Scenario.

However – as mentioned at the start of this section – we do not augment EBPMs to include explicit and detailed rationale information, e.g., "the front end technology for novice bank customers is an HTML 4.2 compliant web browser supporting JavaScript because most such customers are not willing to install new software on their home computers, do not have the skills to properly reconfigure their firewall, and access their bank account information sufficiently infrequently that they do not require more complex features of a graphical user interface." Instead, we rather make sure that the EBPMs contain every domain-oriented detail that the users deem important and base our decision thereon.

9.4.6 Difference Annotations

When designing a software system, the designers not only determine the behavior of an application but shape the future work processes. In most cases, these are bound to achieve the same effect (or a superset thereof) as the present ones, only faster and cheaper or with a higher quality.

Because of this, EBPM is often used to model the relevant existing processes of the application domain before beginning to design. These models show the actors doing their work as they do in the present, using today's artifacts and IT systems ("as-is" models). Based on these models, the future processes are designed with their respective future software support ("to-be" models). This helps to ensure that the old processes and the services they provide are not broken in transit to the future design.

For this purpose, the EBPM method offers an additional feature called *Difference Annotations*. Difference Annotations link specific actions in Cooperation Scenarios to actions in other Cooperation scenarios and are commented textually. In this way, new or different actions in to-be models

can point to their present counterparts and describe relevant differences between present and future.

Difference Annotations can furthermore be used to compare several future scenarios. The creation of alternative EBPM scenario models is justified if they can support decisions on important design issues. For example, those scenarios can highlight different ways of distributing tasks in an organization or demonstrate the variety of possible usage models and their impact on the work process. A Difference Table as shown in Fig. 9.6 can be generated from the Difference annotations attached to a scenario.

action	referenced action	referenced scenario	comment
the Front Office is automatically informed that the contract is ready	the Back Office calls the Front Office to inform them that the contract is ready	application - minimal support	replacing a phone call by automation will speed things up, although detaching Front and Back Office a bit.

Fig. 9.6. One row of a Difference Table (schematic)

Difference Annotations are a simple and effective technique when examining the scenarios, whether to analyze the gap between current and future processes or to compare alternative scenarios. Sometimes we attach them while deriving a new scenario from a basic one. When analyzing retrospectively, we take printed versions of the models and compare them. First, we mark the differences with pen and paper, then, after evaluation, we attach them to the models in electronic form. This cannot be automated because it is all about finding the significant steps and interpreting them. When merging alternative scenarios, we recommend documenting the Difference Table together with the derived decisions in a protocol that is then attached to the "surviving" scenario.

9.4.7 Quantification

A second basis for deciding between different alternative scenarios is the quantification of EBPM scenarios. This is done very straightforward: the domain experts give rough estimates for how long actions take. Transportation and/or wait time is used when annotating transfers of work objects. Processing time is used when annotating inspections and modifications of work objects.

9.5 Relations to Other Approaches

There are other approaches in Requirements and Software Engineering as well as in economics that deal with the same issues as EBPM or have some overlap with it regarding the models and techniques that they use

EBPM's Cooperation Scenario is based on Krabbel's and Wetzel' Cooperation Pictures (cf. [10]). EBPM adds the strictly scenario-based approach and the comprehensive meta model. Another method that is remarkable for its visual representation and appropriateness for group work is PICTIVE (described in [11, 12]), which is rooted in Participatory Design. Contrary to EBPM, it does not provide a straightforward of deriving an object model.

Approaches sharing the scenario-based nature of EBPM come in a various formats, for example Jacobson's Use Cases (cf. [7]) and Rubin's and Goldberg's Object Behavior Analysis (described in [15]). These are more focused on the dialogue between user and software system than EBPM and less on the cooperative work process. They lack a visual representation for their scenarios and are therefore less suited for workshops with groups. Use Cases are superior to EBPM regarding variations and case differentiation when written as main success scenarios with extensions (cf. [2]).

Examples for more formal diagram techniques for processes are UML's Activity Diagram (see [8]) and the Event-Driven Process Chains of Scheer (see [16]). In contrast to EBPM, these can be used for a more formal specification of a process. On the other hand, they are not easily understandable for people without education in math or IT and therefore not well-suited for communication with users and domain experts. They are less object-oriented than EBPM because they do not focus on the domain objects and their usage (although there have been attempts to tackle this problem, see for example [17]). Furthermore, they are unable to "tell stories" like EBPM which can be applied even to cooperative work that consist to great extent of situated actions.

Concluding this section, we want to allude to approaches that deal with design rationale in a way that is potentially compatible to EBPM in the sense that only minor modifications would be necessary to fit them in. One of these approaches is Claims Analysis (see [1]) which augments scenarios that demonstrate specific design alternatives with claims that state expected advantages and disadvantages of those design decisions. Another one is Contribution Structures which adds to requirements explicit information about the persons contributing to them (cf. [6]). Yet another one is the Inquiry Cycle (see [13], [14]), a conceptual framework that recommends to incrementally refine scenarios and requirements documentation

and attach information to them about the related discussions happening in the incremental process.

9.6 Conclusion

Exemplary Business Process Modeling (EBPM) is a lightweight yet highly useful basis for design rationale management. The three main factors that make EBPM so advantageous are:

1. *Its smooth integration with requirements engineering as suggested by Dutoit and Paech (see [5])***.** As EBPM already provides all necessary means, there is no need to duplicate the relevant information in a design rationale management system.
2. *Its focus on just the most relevant scenarios***.** Thus, a maximum of the daily work routine can be captured with optimal effort.
3. *Its focus on positive in formation.* While decisions against a certain alternative can still be derived by looking at the complement of the positive information, there is no need to keep, manage and update information about discarded alternatives.

EBPM has many qualities that can be traced to other approaches:

- It is scenario-based (like the Use Case approach and OBA)
- It models business processes (like UML Activity diagrams or EPCs)
- It analyzes cooperative work (like Cooperation Pictures)
- It is suitable for collaborative workshops (like PICTIVE)

EBPM is unique in its combination of those features. Its power lies in the immediate understandability of its models: all models are rendered graphically and are based on a simple meta model; the story-like structure of the scenarios makes it easy for domain and software experts alike to discuss domain-related matters in workshops. Models are usually very stable at the end of the first workshop. Based on these models, a substantial amount of design rationale can be derived (see Fig. 9.7).

In addition to the design-rationale-related advantages, the analysis of EBPMs can be employed to obtain rationale information for roll-out/migration planning (based on which application parts are used how intensively and on difference tables).

Rationale for …	Provided by analysis of …
business object classes/ ER entities	term or concept in *Model of Terms*
business object class methods	• Usage of work objects in *Scenarios* • glossary from *Model of Terms*
business object class attributes/ER attributes	• aggregations from Model of Terms • glossary from Model of Terms
business object class hierarchy/OR mapping	Hierarchy of Terms in *Model of Terms*
tools	• usage of work objects in *Scenarios* (explicitly modeled tools only) • glossary from *Model of Terms* (explicitly modeled tools only) • similar access to work objects in Scenarios • (implicit) • recurring sequences of work in Scenarios
tool modes	differences in usage for different Scenarios
tool versatility	number of work objects accessed (explicitly modeled tools only)
workplace types	• roles in *Model of Roles* • "work objects used from workplace
services	How different tools provide similar methods on work objects
endorsement of design decision	• difference tables for "as-is" and "to-be" models • difference tables for several "to-be" models • quantification information

Fig. 9.7. Rationale information provided by EBPM

References

[1] Caroll JM (2000) Making Use: Scenario-Based Design of Human–Computer Interaction. Cambridge: MIT

[2] Cockburn A (2000) Writing Effective Use Cases. Reading, MA: Addison-Wesley

[3] Dahl O-J, Nygaard K (1967) SIMULA - A language for Programming and Description of Discrete Event Systems, Oslo 3, Norway, Norwegian Computing Center, Forskningveien 1B, 5th ed.

[4] Dutoit AH, McCall R, Mistrík I, Paech B (2006) Rationale management in software engineering. Heidelberg Berlin New York: Springer

[5] Dutoit AH, Paech B (2000) Supporting evolution: Using rationale in use case driven software development. In: 6th Internaltional Workshop on Requirements Engineering for Software Quality (REFSQ'2000), Interlaken Switzerland, June

[6] Gotel O, Finkelstein A (1995) Contribution structures. In: Proceedings of the Second IEEE International Symposium on Requirements Engineering (RE '95), IEEE Computer Society, York UK, March 27–29, pp. 100–107

[7] Jacobson I, Christerson M, Jonsson P, Overgaard G (1992) Object-Oriented Software Engineering: A Use Case Driven Approach. Reading, MA: Addison-Wesley
[8] Jacobson I, Ericsson M, Jacobson A (1995) The Object Advantage: Business Process Reengineering with Object Technology. Reading, MA: Addison-Wesley
[9] Karsenty L (1996) An Empirical Evaluation of Design Rationale Documents. In: Proc. of the SIGCHI conference on Human factors in computing systems, Vancouver, British Columbia, Canada, pp. 150–156
[10] Krabbel AM, Ratuski S, Wetzel I (1996) Requirements Analysis of Joint Task in Hospitals. In: B. Dahlbom et al. (ed.): IRIS 19 "The Future", Proceedings of the 19th Information Systems Research Seminar in Scandinavia, August 1996 at Lökeberg, Sweden. Gothenburg Studies in Informatics, Report 8, pp. 733–749
[11] Muller MJ (1991) PICTIVE – An exploration in participatory design. In Reaching through Technology: CHI'91 Conference Proceedings, pp. 225–231
[12] Muller M, Tudor L, Wildman D, White E, Root R, Dayton T, Carr R, Diekmann B, Dykstra-Erickson E (1995) Bifocal tools for scenarios and representations in participatory activities with users. In: J.M. Carroll (ed.): Scenario-Based Design. NY: Wiley
[13] Potts C, Bruns G (1988) Recording the reasons for design decisions. In: Proceedings of the 10th International Conference on Software Engineering. Los Alamitos, CA: IEEE Computer Society
[14] Potts C, Takahashi K, Anton A (1994) Inquiry-based scenario analysis of system requirements. IEEE Softw., 11(2):21–32, March
[15] Rubin KS, Goldberg A (1992) Object behavior analysis. Commun. ACM 35(9), 48–62
[16] Scheer A-W (2001) ARIS – Modellierungsmethoden, Metamodelle, Anwendungen.4. Auflage. Berlin Heideleberg New York: Springer-Verlag, 2001
[17] Scheer A-W, Nüttgens M, Zimmermann V (1997) Objektorientierte Ereignisgesteuerte Prozeßkette (oEPC) Methode und Anwendung. Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 141, Saarbrücken URL: http://www.iwi.uni-sb.de/public/iwi-hefte.
[18] Wirfs-Brock R, Wilkerson B (1990) Designing Object-Oriented Software. Englewood Cliffs, NJ: Prentice Hall.
[19] Züllighoven H (2003) Object-Oriented Construction Handbook. San Fransisco: Morgan Kaufmann