Towards an Integration of Business Process Modeling and Object-Oriented Software Development

Peter Loos, Peter Fettke

Chemnitz Univeristy of Technology, Chemnitz, Germany

{loos|peter.fettke}@isym.tu-chemnitz.de

1. Introduction

The successful development and implementation of business information systems requires an integrated approach which includes the seamless design of both the business processes and the information systems supporting the business processes. Therefore, several frameworks and modeling methods have been developed for an integrated modeling of the entire enterprise with respect to both organizational and information systems aspects. Due to the architecture of most existing business information systems, these approaches were usually based on traditional software development paradigms rather than on object-orientation. On the other hand, object-oriented modeling methods used to cover only aspects which are close to implementation, but not the business processes. Currently, however, these two worlds are moving closer together because there are several benefits using business process models during object-oriented software development [5, 15]. This paper describes an approach for integrating business process and object-oriented modeling methods. With this approach, it is possible to model the relevant aspects of a company's business processes and its object-oriented information systems without the need for switching between different modeling paradigms or for translating between different modeling languages.

Paragraph 2 describes the notation of EPCs, a widely used method for business process modeling. The concepts which are not yet covered by the UML and which are useful from the view of business process modeling are discussed in paragraph three. Paragraph 4 describes the usability of business process modeling from the view of object-oriented software development. The approach to integrate business process and object-oriented modeling methods is presented in paragraph five.

2. Business Process Modeling with Event-Driven Process Chains

A common business process modeling language is the event-driven process chain (EPC). The EPC provides comprehensive means for modeling several aspects of a business process [7]. Several providers of standard software packages (SAP among them) have used the EPC for documenting the business processes supported by their software solutions. The main elements of an event-driven process chain (EPCs) are functions and events. Functions are triggered by events, and functions produce events. The control flow of a business process is therefore described by a sequence of alternating events and functions [12]. Alternative or parallel paths can be modeled with logical operators, such as AND (all paths in parallel, simultaneous function processing possible), XOR (only one alternative path), OR (one or several paths) and SEQ (all paths in arbitrary order, but no functions simultaneously) or more complex expressions. These operators can be used for splitting and joining the control flow. An example of an EPC is shown in Fig. 1, where the functions are represented by soft rectangles and the events by hexagons.

Although the control flow is the most important aspect for describing a business process, there are other kinds of information which can be relevant. Such kinds of information include people and organizational units responsible for carrying out a certain function, as well as data flows. Therefore, the control flow of the EPC can be extended with data and organizational elements. For each function it is possible to show which organizational role carrying out this function. These roles, as well as other organizational entities, such as teams, departments, or divisions, and their relationships can be defined in an organizational chart of the company. The data flow is shown with in- and output connections between the EPC's functions and data objects. In a traditional approach, these objects can be defined in a data model (e.g. applying the entity relationship approach), while in an object-oriented approach, objects from a class diagram should be used (as it is explained later on). EPCs can be hierarchically structured across any number of levels by assigning more detailed EPCs to every function within an EPC. As a decomposition such a detailed EPC denotes the process which is carried out when the respective function is triggered. In addition to functions, events, data objects, and organizational elements, a wide range of information can be included into an EPC in order to describe specific aspects more clearly. Such types of information include the use of different kinds of information carriers and communication media, (e.g. paper documents, floppy disks, telephone, etc.), information systems used for supporting functions, human know-how and qualifications, in- and output material of a function (e.g. in the production), required equipment or machinery (resources).

One of the main advantages of the EPC is that it is both powerful and easily understandable for end-users. EPCs are often used for capturing and discussing business processes with people who have never been trained in any kind of modeling technique, e. g. with workers on the shop floor. Although EPCs can be understood even by short-time trained personnel, the same models can be refined and used for the requirements definition of an information system. This is one of the reasons that both many end-user companies and many software vendors are using EPCs for business process modeling.

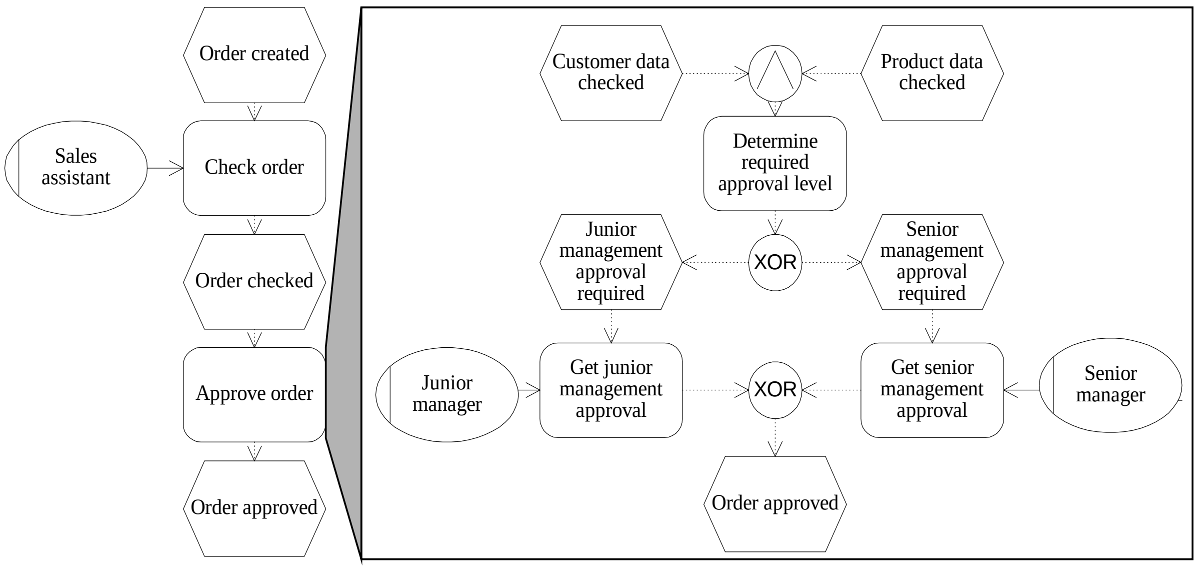

Fig. 1: An example of an event-driven process chain

3. Lacks of Object-oriented Software Development from the View of Business Processes Modeling

The developers of the Unified Modeling Language (UML) have recognized the need for modeling methods which cover the end-users’ views and which allow modeling processes [1]. Therefore diagrams like the use case diagram and the activity diagram have found their way into UML [10], as well as some specific stereotypes for business modeling [10]. However, the UML diagrams still do not cover all aspects required for business process modeling. It is thus necessary to combine the UML with powerful state-of-the-art business process modeling languages [9, 13].

When discussing the lacks of object-oriented modeling from a business modeling view it is necessary to clarify the purpose of the modeling activity in respect to the **scope of the universe of discourse** [8]. The smallest granularity of a process description is object internal. Such processes might be handled completely by one object operation or by several operations of one specific object. The next step of granularity ranges beyond a single object, but stays inside the object system that implements the application software. The largest granularity occurs when business processes range beyond the scope of an object system. Some parts of business processes may be carried out by other object systems, but most of these activities are done manually without any or only with little system support. This classification according to granularity gives an indication for applying UML behavioral diagrams for modeling business processes.

Use case diagrams cover some organizational aspects, especially the interaction with organizational units external to the object system. However, use case diagrams do not offer mechanisms for depicting the process flow, they particularly do not provide means for defining the sequence of activities and conditions for the activation of process activities. The internal flow of use cases has to be described in textual form.

Sequence diagrams and collaboration diagrams offer some aspects of business process flow, but they focus on the interaction between objects and not on the depiction of an overall control flow. Furthermore, these diagrams are close to implementation design. They are useful for the modeling of technical details. While it is possible for end users to get familiar with more straightforward business modeling methods, the detailed description of message exchange in sequence diagrams and collaboration diagrams is not suitable for non-software experts. Finally, the diagrams do not offer a real support for organizational purposes.

Statechart diagrams are usually assigned to one object. This scope is too narrow for most business processes unless it can be assured during designing that a single object class matches exactly the respective business process. For business processes with a higher granularity it is not feasible to define all conceivable states (in the sense of state machines), a business organization can take on. Besides that, no organizational aspects are modeled in statechart diagrams.

Activity diagrams can be applied as business modeling method [3, 11]. They offer components like activities, relationships for control flow, and logical connectors for control flow variations. The elements can be grouped in swimlanes in order to assign organizational units. Compared with EPCs, there are some problems connected with the use of activity diagrams for business process modeling: Not all logical operators for splitting and joining the control flow can be modeled with own depictions in a straightforward way as provided in EPC. More elaborated ways for this have to be employed. The organizational responsibility for activities can be expressed by placing the activities in swimlanes. However, swimlanes are not sufficient for modeling advanced and precise organizational relationships like *is responsible for, provides support for, must be informed about result of,* and *must approve of.* Although there is no inherent reason, UML does not consider activity diagrams for modeling object external flows. There is no support for modeling additional information like it is available in the EPC, e.g. information carriers or required materials and other resources [2]. The definition of activity diagrams as state machines is problematic for applying activity diagrams according to the UML definition for business process modeling, since not all business functions in a company can be regarded as object internal action states.

4. Benefits of Business Process Modeling from the View of Object-oriented Software Development

Business process models can be used for several activities during the object-oriented software development lifecycle. These benefits are now discussed against the background of the unified software development process [6] (refer Fig. 2).

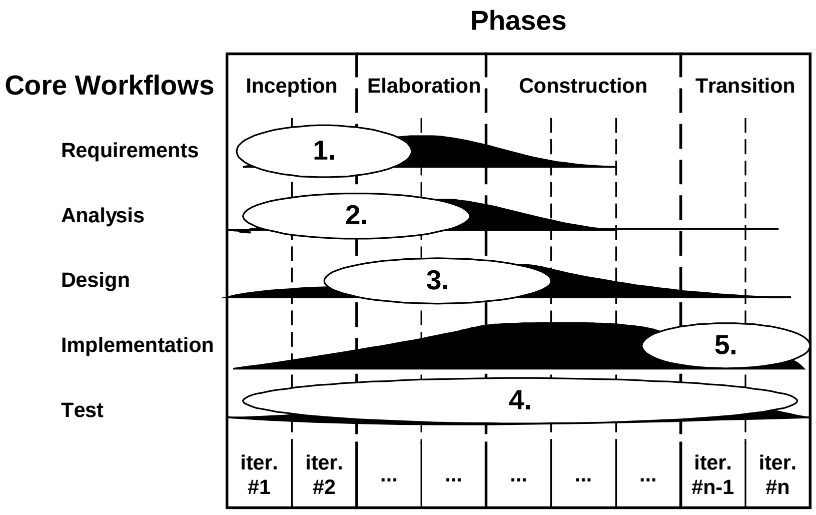

Fig. 2: Overview of applications of business process models in the USDP

Firstly, EPCs can be used for gathering requirements in the inception phase. Therefore it is possible to capture the flow of events in the business domain. Further it is possible to describe in detail organizational aspects of the business information systems. While modeling use cases, EPCs can be used as a starting point for identifying the actors of the system. If the use cases of the system get clearer, EPCs can be applied to describe precisely the workflow of an use case. Secondly, the developed business process model is valuable for creating an outline of the class model while doing analysis. Thus, it is possible to identify relevant classes based on an EPC such like process or entity classes. Thirdly, during the design workflow, EPCs can be used to identify software components by analyzing business areas which need related business data and functions. Fourthly, test activities can benefit of a precise business process model by identifying business test cases which have to be passed before the developed application can be ascertained as correct. Fifthly, EPCs can be used as a blueprint for system deployment. Furthermore, if the developed application is not a standard software package, in this phase the specific aspects of the enterprise which will employ the new information system has to be modeled.

5. Integration of Business Process and Object-Oriented Modeling

The above paragraphs show that there is a need for applying business process modeling concepts in object-oriented software development. This paragraph introduces integrated concepts from EPCs and UML. A more comprehensive discussion can be found in [8]. Each of the UML diagram types contains some aspects which are also relevant for business process modeling. However, since each diagram type has a certain focus, it can be useful to work with several or all of these diagrams in order to make clear different aspects of the business processes and the underlying information system. There can be two kinds of relationships between the EPC and the process-related UML diagrams: The first kind of relationship is relevant if both diagrams are used together, each for describing a different aspect. Such a relationship can include the use of references to the same kinds of elements, and the detailed description of a model element by a diagram of another type. The second kind of relationship is the definition of translation rules from one notation to the other.

The most important connection of the EPC is the one to the **class diagram**, since class diagrams are the central part of the UML and the foundation for the actual implementation. By connecting EPCs and class diagrams it is possible to define which kinds of objects are required, created, or changed by a function, and what operations and attributes are needed. There are four beneficial kinds of reference relationships: On a high level, in- and output-connections between functions and packages can be defined. An input-connection, denoted by an arrow from a package to a function defines that the function requires information stored with objects of the package's classes. On a more detailed level, functions are connected directly with classes rather than packages, in order to express that a function uses or modifies objects of a certain class. The functions from the EPC can be further detailed. On this level, it can be useful to define the operations and attributes to be used. A connection between an operation and a function depicts an operation call during the execution of the function. It is also possible to define in- and output-connections between functions and attributes. Of course, in the actual implemented OO-system it is not possible to access attributes directly, but only via appropriate operations. However, for the development of business processes and the main class structures, it can be rather convenient just to link an attribute to a function, so that it is not necessary to model standard *get*- and *set*-operations explicitly.

UML **statechart diagrams** can be used for modeling an object's lifecycle by defining its relevant states and possible state transitions. Such a statechart diagram is usually not drawn for every class, but only for those classes which require a detailed analysis of their states and transitions, e.g. in rather complicated cases. An EPC can contain in- and output-connections between functions and classes, in order to describe that a function uses or modifies objects of these classes. However, it may not be sufficient for the function to have any object of a specified class, but it may be also important that the object is in a certain state. The information about objects' states can also be modeled in an EPC by drawing in- and output-connections between functions and object states rather than between functions and classes. Each EPC function which transfers the object from one state to another carries out one of the transitions defined in the statechart diagram. EPC and statechart diagram must be consistent, i.e. a function cannot transfer an object from one state into another, if this is not possible according to the transitions defined in the statechart diagram. It should be noted that there is some redundancy between the use of states and events in an EPC. In many cases, an event denotes that an object has changed to a new state.

A **use case diagram** specifies interactions with a software system. It does not need to represent a business-related functionality. There exist three kinds of connections between an use case and an EPC. Firstly, a use case diagram can specify a function of an EPC. For carrying out an EPC function, such as writing an offer, there may be several use cases, e.g. check status, check product catalogue, edit text and print text. Secondly, it is also possible to define the underlying sequence of activities of a use case which are often described verbally only [4, 14] in a more precisely way. For these cases it is useful to assign an EPC to a use case in order to provide a model of the underlying process rather than just a verbal description. Thirdly, an actor in a use case diagram can be interpreted as an organizational unit in the EPC. In all cases, it is possible to use the same roles (respectively actors) in both diagrams. These roles can be defined in the organizational chart.

Sequence and collaboration diagrams are closer to implementation than common applications of EPCs. While EPCs may contain rather loosely defined business functions, and they include only those object types and functions that are relevant from a business perspective, sequence and collaboration diagrams describe the detailed message exchange in a much more formalized way. It is usually not reasonable to translate EPCs into sequence/collaboration diagrams directly although there is some redundant information in both diagrams. It is therefore the responsibility of the modeler to ensure that the message exchanges depicted in a sequence or collaboration diagram are consistent with the business processes to be supported. Mainly, EPCs and sequence/collaboration diagrams are connected via the use of the same elements, i.e. object types and operations.

Since **activity diagrams** cover no additional information comparing to EPCs it is usually not advisable to use both diagram types together to cover the same part of the universe of discourse (although this would be possible, and diagrams of both types could contain references to the same elements). However, it is possible to translate activity diagrams into EPCs and vice versa. Of course, some types of information in an EPC (such as the differences between position and division, material flow or information carriers) can get lost in such a translation, since EPC can cover a wider range of information than activity diagrams.

6. Conclusions

On the one hand, it has been shown that object modeling methods, namely the UML, in their current definition is not sufficient for business modeling. On the other hand, business process modeling methods, namely the EPC, supports object-oriented software development. So it has been argued that the methods of business process and object modeling should be integrated. Such an approach was presented in paragraph 5. It is expected that the further development of the UML, especially of the activity diagrams, will include more and more concepts required for business modeling. Further on, it is recommended that a diagram for organizational structure modeling should be introduced. The OMG (the Object Management Group, which is responsible for the UML standardization) has recognized UML’s gap on business process modeling and has already formed a responsible working group. As long as the business process modeling capability lacks, the presented EPC-UML combination provides a pragmatic approach and an appropriated way for an integrated modeling of business processes and object-oriented information systems.

References

[1] Ambler, S. W.: What's Missing from the UML? Techniques that can help model effective business applications. in: Object Magazine, 7 (1997) 8.

[2] Berkem, B.: Traceability Management from Business Processes to Use Cases with UML – A Proposal for Extensions to the UML’s Activity Diagram Through Goal-Oriented Objects. in: Journal of Object-Oriented Programming. September 1999, S. 29-34, 64.

[3] Fowler, M.; Scott, K.: UML Distilled – A Brief Guide to the Standard Object Modeling Language. 2nd ed., Reading, MA 1999.

[4] Jacobson, I.; Christerson, M.; Jonsson, P.; Övergaard, G.: Object-Oriented Software Engineering - A Use Case Driven Approach, Wokingham et al. 1993.

[5] Jacobson, I.; Ericsson, M.; Jacobson, A.: The Object Advantage, Business Process Reengineering with Object Technology. Wokingham et al. 1994.

[6] Jacobson, I.; Booch, G.; Rumbaugh, J.: The Unified Software Development Process. Reading, MA et al. 1998.

[7] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)". Publication of the Institut für Wirtschaftsinformatik, Paper 89, Saarbrücken 1992 (http://www.iwi.uni-sb.de/public/iwi-hefte/heft089.zip).

[8] Loos, P; Allweyer, T.: Process Orientation and Object-Orientation - An Approach for Integrating UML and Event-Driven Process Chains (EPC). Publication of the Institut für Wirtschaftsinformatik, Paper 144, Saarbrüken, March 1998. (http://www.tu-chemnitz.de/wirtschaft/wi2/home/loos/iwih144.pdf)

[9] Nüttgens, M.; Feld, T.; Zimmermann, V.: Business Process Modeling with EPC and UML: Transformation or Integration?, in: Schader, M.; Korthaus, A. (ed.), The Unified Modeling Language - Technical Aspects and Applications, Physica, Heidelberg 1998, pp. 250-261.

[10] OMG (ed.): Unified Modeling Language Specification, Version 1.3, 1999.

[11] Paech, B.: On the Role of Activity Diagrams in UML, in: P.-A. Muller, J. Bezivin, (eds.), Proceedings of the Workshop <<UML>>’98, Beyond the Notation (Mulhouse, June 3-4, 1998), pp 245-250.2.

[12] Scheer, A.-W.: Business Process Engineering. Reference Models for Industrial Enterprises, 2nd ed., New York et al. 1994.

[13] Scheer, A.-W. ; Nüttgens, M.; Zimmermann, V.: Objektorientierte Ereignisgesteuerte Prozeßkette (oEPK) - Methode und Anwendung. Publication of the Institut für Wirtschaftsinformatik, Heft 141, Saarbrücken 1997.

[14] Schneider, G.; Winters, J. P.: Applying Use Cases: A Practical Guide. Reading, MA, 1998.

[15] Taylor, D. A.: Business Engineering with Object Technology. New York et al. 1995.