

Event-based activity modeling

Lars Bækgaard

Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7E
DK-9220 Aalborg Øst
Denmark
Email: larsb@cs.auc.dk

Abstract

We present and discuss a modeling approach that supports event-based modeling of information and activity in information systems. Interacting human actors and IT-actors may carry out such activity. We use events to create meaningful relations between information structures and the related activities inside and outside an IT-system. We use event-activity diagrams to model activity. Such diagrams support the modeling of activity flow, object flow, shared events, triggering events, and interrupting events.

Keywords: Information systems, human activity systems, event modeling, activity modeling, interaction.

1 Introduction

The primary purpose of this paper is to present and discuss a modeling approach that supports modeling of activity in information systems and their environments. Many information systems operate in ways that are deeply integrated in other organizational activities. In such situations the information systems are part of the organizations rather than merely systems that support the organizations.

Organizations depend on material processes and information processes that deal with movement, manipulation, and consumption of objects and they depend on coordination processes that deal with requests for, agreements about, control of, and evaluation of work processes (Denning & Medina-Mora 1995).

Information systems are responsible for information processes and they may play important roles in material and coordination processes as well. Material processes may be mediated by digitally controlled machinery and many coordination processes are communication processes in which actors express requests, requirements, contracts, and evaluations.

We view information systems as human activity systems (Checkland 1981) in which a combination of human actors and IT-actors (IT-systems) carry out information processing activities (Rose et al 2003). We use the term actor as a reference to both human actors and IT-actors even though we recognize that there are important differences between human beings and IT-actors (Dreyfus & Dreyfus 1986, Weizenbaum 1984).

Our modeling approach is event-based. Events are occurrences without duration (Jackson 1983). We use events to structure the activities that are carried out by actors. We use events to create meaningful relations between human actors and IT-actors and between information systems and their environments in terms of shared events.

Events may be observed and they may have consequences (Bækgaard 2001). When an event occurs information about the event may be registered and activity may be initiated. When a book is borrowed in a library the following may happen. Information about the book and the borrower may be registered and an activity that monitors timely returning of the book may be initiated. When a business receives an order the following may happen. Information about customer, product etc. may be registered, one or more employees may carry out packing and shipping activities, and an ERP-system may carry out invoicing activities.

Human actors and IT-actors interact when they carry out activities. Existing modeling languages focus on interaction either in terms of shared objects or in terms of shared events. Actors share objects when they create, manipulate, and exchange material objects or information objects. Data flow diagrams (De Marco 1978), action diagrams (Goldkuhl 1996), and activity diagrams (Rumbaugh et al 1999) support modeling of object sharing. Shared events have multiple participants. Therefore, they can be used to coordinate the activities of two or more interacting actors. Entity/object life cycles can be modeled in terms of shared events (Jackson 1983, Mathiassen et al 2000). Lindgreen (1990) uses events to trigger activities but he does not model events as atomic processes that can be shared by participants. We are not aware of languages that combine modeling of object sharing with modeling of shared events.

In Section 2 we discuss events and their roles in information systems. In Section 3 and Section 4 we present a modeling approach that supports event-based modeling of activity and information. We use examples to present and discuss our modeling approach and its capabilities and to demonstrate that events can be used as a foundation for combined modeling of interaction as object sharing *and* participation in shared events. In Section 5 we discuss our modeling approach and its relations to other modeling approaches.

2 Events

An event is an occurrence that is assumed to occur at a point in time (Jackson 1983). Events do not exist per se—even the smallest process takes time. When we view a phenomenon as an event we assume that a further subdivision of the phenomenon is irrelevant and that it makes sense to neglect its duration. Events occur within processes like chemical processes, biological processes, technical processes, or social processes.

We can view phenomena like “Customer files an order”, “Borrower borrows a book”, “Website user clicks”, “Heart beats”, or “Elevator starts moving” as events. Each of these phenomena has a durable occurrence but in certain contexts it makes sense to assume that they occur at points in time.

Events can have participants and properties. Participants can be human actors, IT-actors, or things like machines or books. An event is shared if it has two or more participants. One of the participants is active in the sense that this participant requests the event. The other participants are passive in the sense that they react to requests from the active participant. A property can be a value like “Joe” or 7124. Ordering events may have participants like customers and products and they may have properties like dates and times. A borrow event in a library may have a borrower and a book as participants. The borrower is active and requests the borrow event. The book is passive and responds to the request. Passive participants may not be ready to participate in requested events.

Events can be viewed as change objects that have consequences. They can be used to define business workflows (Barros & ter Hofstede 1998). For example, an event like ordering may trigger packing and shipping activities. Yourdon (1989) uses events to trigger processes in data flow diagrams. Jackson (1983) and Mathiassen et al (2000) use events to synchronize a computerized information model with the modeled universe of discourse (UoD). Each time an event occurs in the UoD a similar event is triggered in the computerized information model in order to ensure that the state of the model reflects the state of the UoD. Used in this way events correspond to atomic database transactions (Gray 1981). The occurrence of an event can influence the admissibility of other events. For example, a library book must be borrowed and returned in an alternating sequence.

Events can be viewed as information objects that are observed and described. For example, a data warehouse with multi-dimensional information about customer behavior may be based on information about events like ordering and payment (Bækgaard 1999, Kimball 1996). Information about events can be registered, stored, manipulated, and presented for actors.

When we view events as information objects we can use them to build shared action memories (Goldkuhl & Ågerfalk 1998) that support actors that participate in activities. When we view events as change objects we focus on the activities in which actors participate and in which they may use action memories to improve the activities. This implies that events may play a central role when we want to understand the relations between information and activity. Events are parts of the activities in which actors use information about events.

We use expressions on the form “Name [ParticipantTypes] (PropertyTypes)” to define event signatures. An event signature defines a type of conforming events. The element “ParticipantTypes” defines the types of participants that can participate in events of the defined type. The element “PropertyTypes” defines the types of properties of events of the defined type.

An event expression conforms to a signature expression if it has the same name as the signature and if its participants and properties conform to the participant and property types of the signature. For example, the event expression Borrow [Borrower1, BookItem4] (September 29, 2001) conforms to the signature expression Borrow [Borrower, BookItem] (Date).

In the following sections we present a modeling approach that supports modeling of events as both change objects and information objects. In Section 3 we show how to model events as change objects that occur within activities that are carried out by actors. In Section 4 we show how to model events as information objects that can be observed and about which information can be registered.

3 Activity modeling

Activity modeling is a process in which selected activities of selected actors are analyzed and described.

Conceptual activity models (Checkland 1981) can be used to model an activity as a set of related activities. Such models contain brief descriptions of the involved activities and they define the flow of activity in terms of temporal dependencies between activities. There are no explicit references to manipulation or flow of objects and there is no notion of events in

conceptual activity models. Such models can be useful in initial analysis activities (Avison & Wood-Harper 1990).

Data flow diagrams (De Marco 1978), action diagrams (Goldkuhl 1996), and activity diagrams (Rumbaugh et al 1999) are flow-based in the sense that they can be used to model a set of activities and the exchange of objects between these. Data flow diagrams are restricted to information objects. Action diagrams can be used to model exchange of material objects as well. Activity diagrams can be used to model signals that are sent from an activity modeled in one diagram to an activity modeled in another diagram. Flow-based approaches support the modeling of transformation and flow of objects but they must be extended in order to support event-based modeling.

Statecharts (Rumbaugh et al 1999) can be used to model the sequencing of messages send to and from state-dependent (encapsulated) objects in a way that resembles the sending and receiving of signals in activity diagrams. If encapsulated objects are used to model events the consequences of the events must be modeled in a fragmented manner because they must be distributed to all participating objects (Bækgaard 2002). Originally, this distribution paradigm was developed in order to support the creation of manageable and maintainable software complexes. Our intentions are different. We want to model the activity of actors.

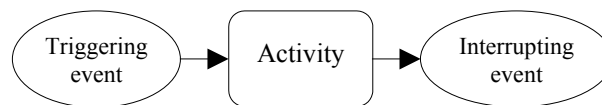


Figure 1 Generic event-activity diagram

We have based our modeling approach on activity diagrams because they offer a fairly rich repertoire of constructs for the modeling of activity flow and object flow. We use event-activity diagrams to model activity in a way that combines two types of interaction between activities – sharing of objects and participation in shared events. Event-activity diagrams are activity diagrams (Rumbaugh et al 1999) extended with events and asynchronous object sharing. An event-activity diagram defines a type of activity and it is composed of triggering events, interrupting events, and an activity (Figure 1). An activity is initiated when a triggering event occurs. The activity runs until it reaches an exit point or until an interrupting event occurs. One or more actors carry out the activity.

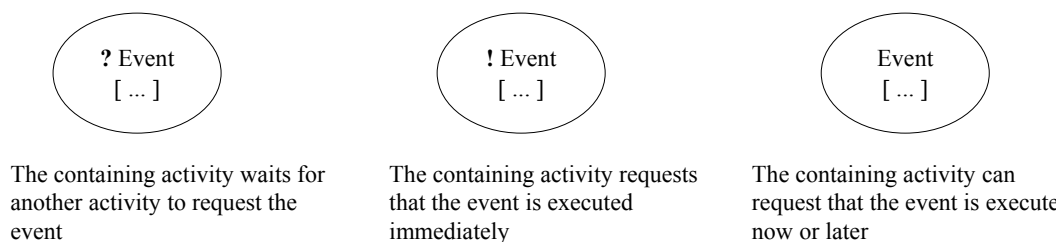


Figure 2 Event participation roles

Event-activity diagrams can contain three different types of event participation roles (Figure 2). If the event name is preceded by a question mark the containing activity is passive relative to the event and it waits for another activity to request the event. If the event name is preceded by an exclamation mark the containing activity is active relative to the event and it requests the event to be executed immediately. Otherwise, the containing activity is active relative to the event and it can request the event to be executed now or later. This implies that we can model passive event participation and two types of active event participation.

The event-activity diagrams in the following figures model activities in a library. A Book Monitor (Figure 3) is triggered by the occurrence of a Borrow event. The solid arrows represent transfer of control. The black dot defines the entry point. The black dot with the ring defines the exit point. After 30 days a Recall event is requested. After another 5 days a Fine event is requested and the activity terminates. A Book Monitor is terminated if an interrupting Return event occurs. If such a Return event occurs within 30 days no Recall event is requested. If an interrupting Return event occurs within 35 days no Fine event is requested. Recall and Fine events are requested immediately. Therefore, a Recall event is requested after exactly 30 days and a Fine event is requested after exactly 35 days.

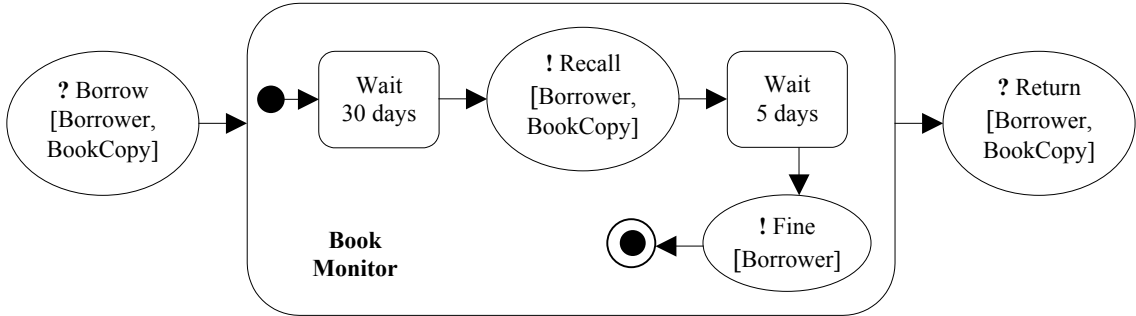


Figure 3 Book Monitor

The Borrower entity that participates in Recall, Fine, and Return events must be identical to the Borrower entity that participates in the triggering Borrow event. The BookCopy entity that participates in Recall and Return events must be identical to the BookCopy entity that participates in the triggering Borrow event. This is indicated by the explicit parameter names. Borrow and Return events are requested by other activities (Figure 5).

Maintain Book Collection (Figure 4) is triggered by the occurrence of an OpenLibrary event and it is interrupted by the occurrence of a CloseLibrary event. Two activities are carried out in parallel as indicated by the horizontal bar. They interact via the object Book List. The dotted arrow that connects the two activities represents this interaction. A dotted line does not represent transfer of control. OpenLibrary and CloseLibrary must be requested by other activities.

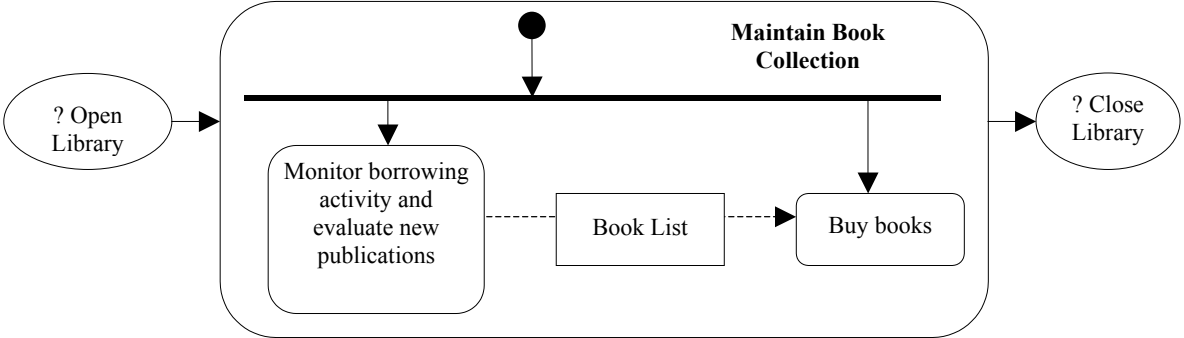


Figure 4 Maintain Book Collection

We use event-activity diagrams to model entity life cycles (Jackson 1983, Mathiassen et al 2000) in terms of temporal rules for the participation of entities in events. The event-activity diagrams in the following figures model entity life cycles.

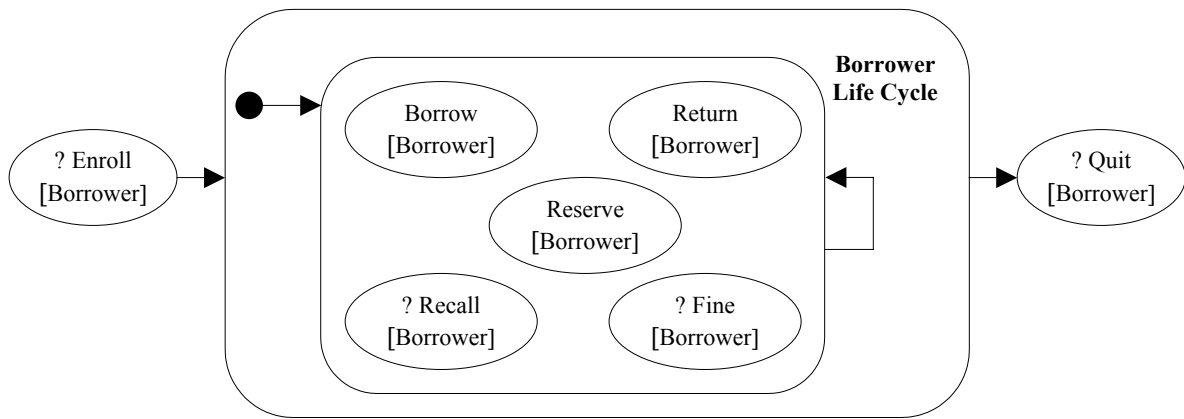


Figure 5 Borrower Life Cycle

A Borrower Life Cycle (Figure 5) is triggered by the occurrence of an Enroll event and it is interrupted by the occurrence of a Quit event. A Borrower Life Cycle is a sequence of Borrow, Return, Reserve, Recall, and Fine events. The Borrower activity chooses one of these events during each of the iterations. The Borrower entity that participates in Borrow, Return, Reserve, Fine and Quit events must be identical to the Borrower entity that participates in the triggering Enroll event. Borrow, Return, and Reserve events are requested at undefined points in time. Enroll, Recall, Fine, and Quit must be requested by other activities.

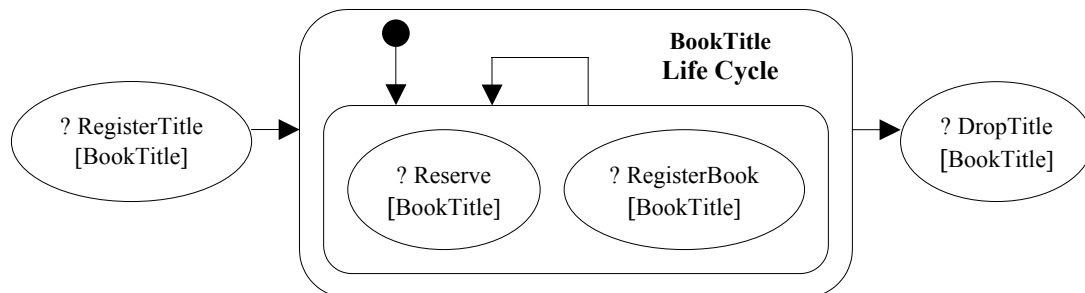


Figure 6 BookTitle Life Cycle

A BookTitle Life Cycle (Figure 6) is triggered by the occurrence of a RegisterTitle event and it is interrupted by the occurrence of a DropTitle event. A BookTitle Life Cycle is a sequence of Reserve and RegisterBook events. The BookTitle entity that participates in Reserve, RegisterBook, and DropTitle events must be identical to the BookTitle entity that participates in the triggering RegisterTitle event. All events must be requested by other activities.

A BookCopy Life Cycle (Figure 7) is triggered by the occurrence of a RegisterCopy event and it is interrupted by the occurrence of a DropCopy event. A BookCopy Life Cycle is composed of two parallel activities. One of these activities is composed of an alternating sequence of Borrow and Return events. The other activity is composed of a sequence of Recall events. The BookCopy entity that participates in Borrow, Return, Recall, and DropCopy events must be identical to the BookCopy entity that participates in the triggering RegisterCopy event. All events must be requested by other activities.

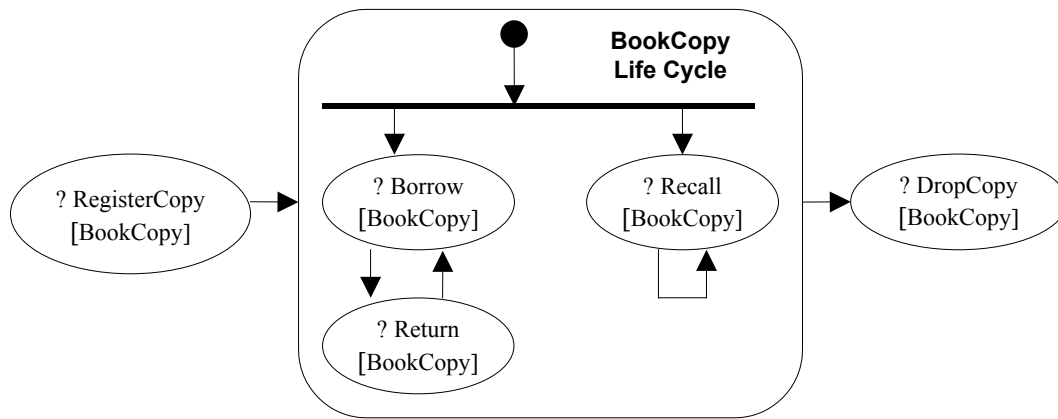


Figure 7 BookCopy Life Cycle

A set of event-activity diagrams defines a set of activity types. Activities are instantiated as a consequence of triggering events and interrupting events may terminate them. Activities interact via shared resources or via shared events.

4 Information modeling

Information modeling is a process in which information about selected phenomena is analyzed and described. The resulting information model can be expressed in terms of, say, entity-relationship diagrams (Batini et al 1982) or class diagrams (Rumbaugh 1999). Such diagrams can be interpreted as definitions of general properties of the modeled information *and* as conceptual definitions of the modeled phenomena.

Actors produce and use information when they participate in activities (Checkland & Holwell 1998). This is one of the reasons why most IT-based information systems contain some sort of database that facilitates storage and retrieval of information. Goldkuhl & Ågerfalk (1998) use the term action memory in order to emphasize that information about historical actions may be prerequisites for new actions.

We use historical information about events and their participants and properties to represent an organization's action memory and other information of relevance to actors. The purpose is to create a shared pool of information that can be used by actors to improve the activities in which they participate.

We can use records (Codd 1970), objects (Mathiassen et al 2000), or entities (Chen 1976, Shipman 1981) to represent structured information. Records have the form $(value_1, \dots, value_n)$. A record is simple if each value is a simple value like an integer or a string. A record is complex if the values can be records, tables, or other types of composite values. Simple records suffer from serious limitations with respect to information modeling because it is hard to define the relations between records and modeled phenomena in a coherent manner (Kent 1979). An object is a simple or complex record that is augmented with an identifier and a set of methods that facilitate access to the record. Objects can be used to model information about events in a rich and relevant manner (Bækgaard 2002).

We use an entity-based modeling approach in order to avoid the information modeling limitations of the record-based paradigm and the event modeling limitations of the object-based paradigm (as argued in the previous section). Entity-based models use graphs rather than records to represent individuals and their mutual relations. Each modeled phenomenon is

represented by a unique entity that is related to values that represent the properties of the phenomenon.

We assume that events and entities have the following characteristics when we observe them and record information about them. We assume that they are unique and can be distinguished from all other phenomena. We assume that we can attribute values to them in order to describe their individual characteristics. We assume that we can relate them to other phenomena. We assume that we can classify them into sets of individuals with common properties.

Events and entities have different temporal semantics. Events occur at time points whereas entities exist within periods of time. These temporal semantics are built into the event-entity-relationship model and its manipulation language (Bækgaard 1999, Bækgaard 2001). Therefore, it is not necessary for us to model the temporal properties of events and entities explicitly.

We use the event-entity-relationship model to express information structures in terms of entity sets, event sets, value sets, and relationship sets. We model phenomena with durable existences as entities. We model phenomena with momentary occurrences as events. We use values to represent properties of modeled phenomena. We use binary relationships to relate pairs of entities, events, and values. Entities may be related to entities, events, and values. Events may be related to entities and values. Subset hierarchies can be used to model concept hierarchies.

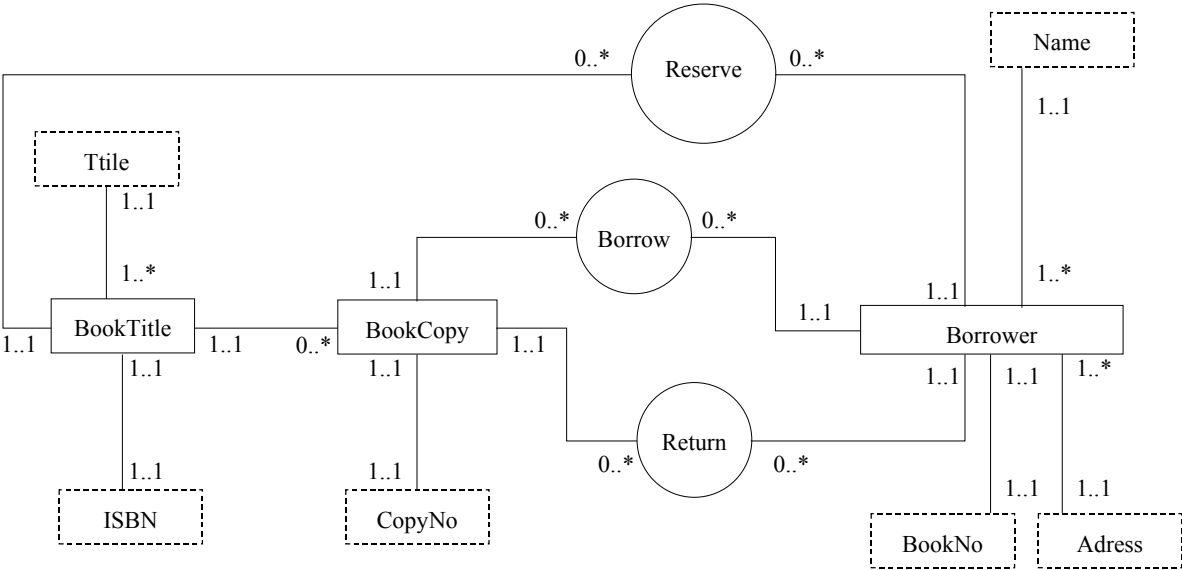


Figure 8 Library information structure

We have modeled a simple information structure from a fictitious library (Figure 8). Solid rectangles represent entity sets. Circles represent event sets. Dotted rectangles represent value sets. Lines represent relationship sets. Each entity in BookTitle represents one book title in the library. Each entity in BookCopy represents one physical copy of a book title in the library. Each entity in Borrower represents one borrower in the library. Each event in Borrow represents one borrow event in the library. Each event in Return represents one return event in the library. Each event in Reserve represents one reserve event in the library. Each entity in

BookTitle must be related to exactly one value in Title and to exactly one value in ISBN. Each value in ISBN must be related to exactly one entity in BookTitle.

A new event is added to Borrow each time a book is borrowed. The new event is related to the participating BookCopy and Borrower entities. Likewise, a new event is added to Borrow each time a book is returned. The new event is related to the participating BookCopy and Borrower entities.

Event-based information structures may be used to represent shared information pools that can be used to support asynchronous interaction between actors. Actors can use information that is stored by other actors.

5 Discussion and future work

We use event-activity diagrams to model activities that may be carried out by a number of human actors and IT-actors. Event-activity diagrams are extensions of activity diagrams (Rumbaugh 1999). First, they contain a notation for events that may be shared by two or more activities. Activities can be active or passive relative to specific events. Second, they contain a notation for object exchange between parallel activities. Third, they contain a notation for parameterized activity types that are instantiated by parameters from triggering events.

We use event-entity-relationship diagrams (Bækgaard 2001) to express event-based information structures that can represent an organization's action memory (Goldkuhl & Ågerfalk 1998) and other relevant information. Event-based information structures combine the characteristics of multi-dimensional modeling of historical information in data warehousing environments (Bækgaard 1999, Kimball 1996) with conventional entity-relationship modeling (Batini et al 1982).

We use events to relate information structures to activities. In the library example from the previous sections we have modeled Borrow events as entity-like phenomena about which information can be registered. Each Borrow event can be related to the participating Borrower and BookCopy entities (Figure 8). We have modeled Borrow events as triggering events for Monitor Book activities (Figure 4). We have modeled Borrow events as parts of the activities in which Borrower entities participate (Figure 5). We have modeled Borrow events as parts of the activities in which BookCopy entities participate (Figure 7). When a Borrow event occurs information about is registered, a Book Monitor activity is initiated, and a Borrower entity and a BookCopy entity participate in the event.

We model interaction in terms of shared objects and in terms of shared events. We model object sharing in terms of exchange of objects between activities and in terms of access to event-based information pools. We have modeled two parallel activities that exchange information (Figure 4). Likewise, two sequential activities can interchange objects when control is transferred from one activity to the other. Activities can interact via shared resources. We have modeled a library information structure that can be used as a shared information resource (Figure 8). We model shared events that define synchronization points for two or more independent activities. We have modeled Borrow events as events that are shared by active Borrower entities and passive BookCopy entities (Figure 5 and Figure 7).

A variety of techniques have been proposed for the modeling of activity but as far as we know no existing modeling languages combine and integrate the modeling of information, activity flow, object flow, and event sharing. A data flow diagram defines a set of algorithmic

processes that interact in terms of data flows (De Marco 1978). Event-activity diagrams can be used to model interaction in terms of data flows and in terms of shared events. Action diagrams define the relations between information and materials and the actions that change these. Action diagrams support the specification of the actors that take action (Goldkuhl 1996). Like Action diagrams our notation supports the modeling of flow of both information objects and material objects. Unlike action diagrams our event-activity diagrams supports the modeling of shared events. Currently, event-activity diagrams contain no notation for the actors that carry out the activities.

Multiview is a systems development method that combines conceptual activity modeling with data flow diagramming (Avison & Wood-Harper 1990). Based on a set of SSM-inspired conceptual activity models (Checkland 1981) a set of new IT-functions are identified. For each function the triggering event and the involved information are identified. Based on this analysis data flow diagrams and entity-relationship diagrams (Chen 1976) are constructed. Multiview's transition from SSM-analysis to the analysis of functions and entities is somewhat unclear due to a gap between conceptual activity models and IT-functions. We believe that our modeling approach can improve Multiview if it is used instead of data flow diagrams and entity-relationship diagrams. First, we model events independently of decisions about IT-functions. Second, we relate events to activities that process information *and* to activities that do not process information. Third, we use events to create meaningful relations between information structures and the related activities inside and outside an IT-system.

Our modeling approach implies a perspective on information systems as sets of parallel activities that interact in terms of object exchange, shared resource pools, and shared events. Activities are triggered by events and they may be interrupted by events. Future versions of our approach will include notation for the human actors and IT-actors that carry out the activities.

Event-activity diagrams are based on a somewhat simplistic assumption about shared events and their roles in human activity. The implicit execution scheme is as follows. An active activity requests an event and each passive activity decides to participate or to refuse to participate. If a passive active is able to participate it has to participate. This simple negotiation scheme is sufficient if the passive activity is the life cycle of, say, a library book. Other types of events may require a more complex negotiation scheme. It is likely that the language action perspective (Winograd & Flores 1986) can be used to improve the communicative aspects of our modeling approach. Likewise, other theoretical frameworks like activity theory (Engeström 1991) may be used to improve our approach.

Acknowledgements

We want to thank the anonymous referees for their comments and suggestions.

References

- Avison D.E., Wood-Harper A.T. (1990), Multiview, Blackwell Scientific Publications.
- Barros A.P., ter Hofstede A.H.M. (1998), "Towards the Construction of Workflow-Suitable Conceptual Modelling Techniques," Information Systems Journal.
- Batini C., Ceri S., Navathe S.B. (1992), Conceptual Database Design. An Entity-relationship Approach, Benjamin/Cummings.
- Bækgaard L. (1999), Event-Entity-Relationship Modeling in Data Warehouse Environments. DOLAP'99. International Workshop on Data Warehousing and OLAP, Kansas City, USA.

- Bækgaard L. (2001), "Event Modeling," In Rossi, M., Siau K. (eds.), *Information Modeling in the New Millennium*, Idea Group Publishing.
- Bækgaard L. (2002), "Event Modeling in UML," *Unified Modeling Language and Unified Process (part of IRMA '02 International Conference)*. Seattle, Washington.
- Checkland P. (1981), *Systems Thinking, Systems Practice*, Wiley.
- Checkland P., Holwell S. (1998), *Information, Systems and Information Systems*, Wiley.
- Chen P.P. (1976), "The Entity-Relationship Model - Towards a Unified View of Data," *ACM Transactions on Database Systems*.
- Codd E.F. (1970), "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*.
- Denning P.J., Medina-Mora, R. (1995), "Completing the Loops," *Interfaces*.
- De Marco T. (1978), *Structured Analysis and System Specification*, Yourdon.
- Dreyfus H.L., Dreyfus S.E. (1986), *Mind Over Machine*, The Free Press, New York.
- Engeström Y. (1991), "Developmental Work Research: Reconstructing Expertise Through Expansive Learning," in Nurminen et al. (eds.), *Human Jobs and Computer Interfaces Conference*, University of Tampere.
- Goldkuhl G. (1996), "Generic Business Frameworks and Action Modeling," *International Workshop on Communication Modeling*, Tilburg, The Netherlands.
- Goldkuhl G., Ågerfalk P.J. (1998), "Action within Information Systems: Outline of a Requirements Engineering Method," *4th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ '98)*, Pisa, Italy.
- Gray, J. (1981). "The Transaction Concept. Virtues and Limitations," VLDB'81. 7th International Conference on Very Large Databases, Cannes, France.
- Jackson M. (1983), *System Development*, Prentice Hall International.
- Jacobson I., Booch G., Rumbaugh J. (1999), *The Unified Development Process*, Addison-Wesley.
- Kent W. (1979), "Limitations of Record-Based Information Models," *ACM Transactions on Database Systems*.
- Kimball, R. (1996), *The Data Warehouse Toolkit*, Wiley.
- Kruchten P. (2000), *The Rational Unified Process - An Introduction*, Addison-Wesley.
- Lind M. (2003), "The Diversity of Work Practices – Challenging the Existing Notion of Business Process Type", *Proceedings of ALOIS 2003 - Action in Language, Organisations, and Information Systems*, Linköping, Sweden.
- Lindgreen P. (1990), *Systems Analysis (in Danish)*, Jurist- og Økonomforbundets Forlag.
- Mathiassen L., Munk-Madsen A., Nielsen P.A., Stage J. (2000), *Object-Oriented Analysis & Design*, Marko.
- Rose J., Jones M., Truex D. (2003), "The Problem of Agency: How Humans Act, How Machines Act", *Proceedings of ALOIS 2003 - Action in Language, Organisations, and Information Systems*, Linköping, Sweden.
- Shipman D. (1981), "The Functional Model and the Data Language DAPLEX. *ACM Transactions on Database Systems*.
- Rumbaugh J., Jacobson I., Booch G. (1999), *The Unified Modeling Language Reference Manual*, Addison-Wesley.
- Yourdon E. (1989), *Modern Structured Analysis*, Prentice-Hall.
- Weizenbaum J. (1984), *Computer Power and Human Reason*, Penguin Books.
- Winograd T., Flores F. (1986), *Understanding Computers and Cognition*, Addison-Wesley.