8-6-2011

A Socio-Technical Approach to Interaction Modeling

Lars Bækgaard
Aarhus University, larsb@hih.au.dk

A Socio-Technical Approach to Interaction Modeling

Lars Bækgaard
Business and Social Sciences, Aarhus University, Herning, Denmark
larsb@hih.au.dk

Abstract

The purpose of the paper is to present and discuss a set of interaction primitives that can be used to model the dynamics of socio-technical activity systems, including information systems, in a way that emphasizes structural aspects of the interaction that occurs in such systems. The primitives are based on a unifying, conceptual definition of the disparate interaction types. The primitives can be combined and thus used to represent mediated interaction. We present a set of visualizations that can be used to define multiple related interactions and we present and discuss a set of examples that indicate that interaction primitives can be useful modeling tools that can supplement conventional flow-oriented modeling of business processes.

Keywords

Information systems. Interaction. Scenarios. Modeling. Socio-technical systems.

Introduction

Interaction is a widely used concept that occurs in many different areas of information systems and information systems development. Information systems can be viewed as activity systems (Checkland and Holwell 1998) or work systems (Alter 2006) in which human beings interact with other human beings and with technology and objects. DEMO (Dietz 2006) and BAT (Goldkuhl 1996; Goldkuhl and Lind 2004) are business models that view business activity in terms of interaction between business parties.

We present and discuss an approach to interaction modeling that is based on a set of interaction primitives and a corresponding visualization technique that supports modeling of complex networks of interactions. We are not aware of any existing modeling approach that supports the variety of interactions that is covered by our primitives.

The underlying rationale is that interaction is a fundamental characteristic of information systems and that interaction modeling should play a substantial role in information systems analysis and design. Each individual interaction in an information system can be viewed as a dynamic relation between and actor and one or more elements in the system. Interaction is a source of internal and external change. Exchange of representations between two elements in an information system is a source of internal change. Exchange of representations between an information system and its environment is a source of change in the relations between an information system and its environment.

Within the area of HCI interaction is understood as interaction between human beings and computers (Rogers, Sharp et al. 2002). Use cases represent systems that offers services to actors (Cockburn 2001). A use case specification defines interaction between a system and one or more actors. It does not define interaction between the actors unless their interaction is mediated by the system. Neither human-computer interaction nor use cases can capture direct interaction between two human actors.

Many modeling languages support activity modeling. However, each of them support a limited form of interaction modeling. UML interaction diagrams represent interaction by messages by means of which objects control objects (Rumbaugh, Jacobson et al. 1999). Data flow diagrams represent interaction by data flows that enable two activities to interchange representations (De Marco 1978). Activity diagrams (Rumbaugh, Jacobson et al. 1999), EPC diagrams (Dehnert 2002; Lübke, Lüecke et al. 2006), and BPMN diagrams (White 2004) can be used to represent two different types of interaction: Exchange of representations between activities and transfer of control between activities.

Clearly, interaction is an important concept that is treated in a restricted and somewhat ad hoc manner. We propose a set of interaction primitives that cover all the types of interactions that are inherent in the above-mentioned approaches. We supplement the primitives with a visualization technique that makes it possible to model situations that are characterized by multiple related interactions.

Our research method can be characterized as design science (March and Smith 1995; Hevner, March et al. 2004). We have designed our interaction primitives and the corresponding visualization technique. The science aspect is represented by a case study that we use as a basis for an evaluation of our modeling approach.

Botanizing Modeling Languages

In this section we discuss three modeling languages with respect to their use of interaction primitives. We describe the basic idea of each language and the types of interaction primitives it supports. This botanizing gives us four basic kinds of interaction primitives that are useful when we design information systems.

A data flow diagram can represent interaction in terms of flow of representations between participants (De Marco 1978). A participant can be an external source/consumer of representations, a representation store, or an activity that manipulate representations. The partial data flow diagram in Figure 1 represents a situation where a representation *r* is flowing from the activity a_1 to the activity a_2.

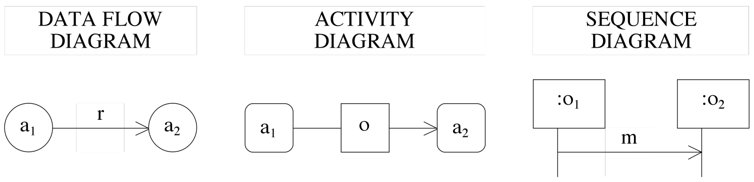

Figure 1 Interaction in modeling languages

A UML activity diagram can represent interaction in terms of flow of objects between activities and in terms of one activity controlling another activity (Rumbaugh et al. 1999). The partial activity diagram in Figure 1 represents a situation where an object, *o*, is flowing from the activity a_1 to the activity a_2.

A UML sequence diagram can represent interaction in terms of messages that are passed among objects (Rumbaugh et al. 1999). The partial sequence diagram in Figure 1 represents a situation where an object, o_1, sends a message, *m*, to an object o_2. A message is a request that activates an action in the receiving object. Messages can contain parameters.

Sensing is a type of interaction where a participant senses aspects of someone or something. For example, a customer may listen to a radio in a store. The data flow diagram in Figure 1 can be interpreted as a sensing where the activity a_2 senses the representation *r* if *r* is loosely coupled to its medium. For example, this situation occurs when a copy of a digital file is transferred via a network. After the transfer both a_1 and a_2 has access to *r*. The activity diagram in Figure 1 can be interpreted as a sensing where the activity a_2 senses the object *o*. The sequence diagram in Figure 1 can be interpreted as a sensing where the object o_2 senses parameters that are passed via the message *m*.

Moving is a type of interaction where something or someone is moved from a source to a destination. For example, an employee may move items from a storage room to a shelf in a store. The data flow diagram in Figure 1 can be interpreted as a moving where the representation *r* is moved from the activity a_1 to the activity a_2 if *r* is physically bound to its medium. For example, this situation occurs when *r* is physically bound to a piece of paper or a digital medium. The activity diagram in Figure 1 can be interpreted as a moving where the object *o* is moved from the activity a_1 to the activity a_2. The activities in an activity diagram may be explicitly located in terms of named swim lanes. The sequence diagram in Figure 1 should not be interpreted as a moving. The reason is that the parameters that are passed via the message *m* are per definition not coupled to a medium.

Controlling is a type of interaction where one participant controls the behavior of another participant. For example, an accountant may control a piece of accounting software in order to get certain computations done. The activity diagram in Figure 1 can be interpreted as a controlling where the activity a_1 terminates itself and initiates a_2 when the object *o* flows from a_1 to a_2. The sequence diagram in Figure 1 can be interpreted as a controlling where the object o_1 controls the actions of the object o_2 by sending a message that initiates a certain set of actions in o_2.

Modifying is a type of interaction where one participant modifies something. For example, a programmer may modify a piece of source code in order to add new functionality. The sequence diagram in Figure 1 can be interpreted as a controlling where the object o_1 controls the actions of the object o_2 by sending a message *m* that initiates a certain action in o_2. Modifying can also be indicated in activity diagrams since the exchanged objects may have visible states that change as they passed from activity to activity.

This walk-through has given us four types of interactions that are scattered in a number of modeling methods: *sensing, moving, modifying* and *controlling*. In the following we present of model of these types and ways of visualizing them.

Interaction Primitives

In this section we define an ontology-based model of a set of interaction primitives. The model is not a modeling language, but rather a source of various visualizations and notational forms suited for a variety of purposes.

Our notion of interaction is based on a basic set of uni-directional interaction primitives. We view interaction as a dynamic relation between two elements in an activity system. This implies, for example, that we view the (modifying) actions of an actor that modifies an object as interactive actions. And we view the (observing) actions of an actor that observes an object as interactive actions. The primitives can be combined to represent bi-directional interactions.

We do not claim that our four interaction primitives constitute a complete set of primitives that cover all imaginable forms of interaction. They were collected from existing methodologies and new technologies may appear that necessitates new primitives. For example, pervasive computing emphasizes space and movements in space in a way we have not seen in existing methods (Bardram and Bossen 2005).

SENSE is a primitive that represents a situation where an Experiencer senses aspects of a Source. Its actants are Experiencer, Phenomenon, and Source. Its effect depends upon the Phenomenon sensed: reading a book increases the Experiencer's ability to tell about it, whereas reading a warning sign decreases his desire to progress further.

SENSE has two variants. The general variant SENSE represents a situation where the Experiencer is different from the Source. Example: A person may listen to music from a radio. FEEL is a variant that represents a situation where the Experiencer is identical to the Source. Example: A person may sense aspects of his own emotional state.

SENSE has the following format: <*Experiencer*> <*Senses*> <*Phenomenon*> <*Source*>.

Examples:

- *He hears music from the radio*
- *He smells the odor of the fish*
- *He reads a copy of the file*

MOVE is a primitive that represents a situation where an Agent moves an Object from a Source to a Destination. The actants are Agent, Object, Source and Destination. The effect is to increase the Object's ability to participate in actions whose Source equals the Destination of the move action. Example: Flying from Copenhagen to Stockholm enables a person to fly from Stockholm to Madrid.

MOVE has four variants. The general variant (MOVE) represents a situation where the Agent is different from both the Source, the Object, and the Destination. Example: A customer can transport products from a store to his home. GIVE represents a situation where the Agent is identical to the Source. The Agent/Source gives an object to a Destination. Example: A customer can give an order to an employee. TAKE represents a situation where the Agent is identical to the Destination. The Agent/Destination takes an Object from the Source. Example: A customer can take a product from a shelf. WALK represents situations where the Agent is identical to the Object. The Agent/Object moves itself from a Source to a Destination. Example: An employee can walk from an office to a department store.

MOVE has the following format: <*Agent*> <*Moves*> <*Object*> <*Source*> <*Destination*>.

Examples:

- *The customer throws the product from the shelf into the basket*
- *The pump pumps the water from the heater to the cooler*
- *The train carries the passenger from Aarhus to Copenhagen*

MODIFY is a primitive that represents a situation where an Agent modifies an Object. The actants are Agent and Object. Examples: Persons can change the properties of things, persons can combine things into new things, persons can divide things into sets of things, employees can modify raw materials into products, programmers can modify software, journalists can write articles, IT systems can use their actuators to modify objects. The effect solely depends upon the kind of modification. Assembling a kit to a chair enables the assemblage to participate as Destination in the action of sitting down, whereas cooking potatoes enables them to participate as the Object of eating.

MODIFY has the following format: <*Agent*> <*Event*> <*Object*>.

Examples:

- *The cook makes a pizza*
- *The software agent changes the user profile*
- *The database module deletes the row from the database*

CONTROL is a primitive that represents a situation where an Agent uses requests to control an Experiencer. The actants are Agent and Experiencer. Examples: A department manager asks an employee to undertake a certain task, a business intelligence system asks for certain representations in a database, actors can initiate activity, actors can redirect a flow of activities, actors can suspend and terminate activity.

Requests can be communicative or materiel depending on the qualities of the Agent and the Experiencer. Both the Agent and the Experiencer must be human beings in order for a request to be communicative. When one person request something from another person the request may have the form of linguistic expression like "please, give me the butter". A person that controls a user interface may express a request in linguistic terms like "select all employees from New York". However, the request is material because the user interface is bound to react to the request in a material manner.

CONTROL has the following format: *<Agent> <Event> <Experiencer>*.

Examples:

- *The boss requests a report from an employee*
- *The user turns on the system*
- *The reading method calls the constructor method*
- *The user interface terminates the simulation*

An IT system can be controlled by a set of requests that the IT system can respond to. Such a request set may include requests like trigger, pause, resume, terminate, and other requests. The request set of a controlled IT system defines an action space for the controlling actor. The action space depends on the request set and the IT system's responses to each request. The precise account of the human flexibility is outside the present paper and belongs to text linguistics and conversation analysis.

Modeling Primitives

In the following we introduce a set of graphical visualizations for our four interaction primitives. The primitives constitute a model that can be visualized in several ways, depending upon the purpose. The golden rule is that the difficult and problematic parts of the representation are represented graphically whereas the unproblematic parts are represented by text or left out.

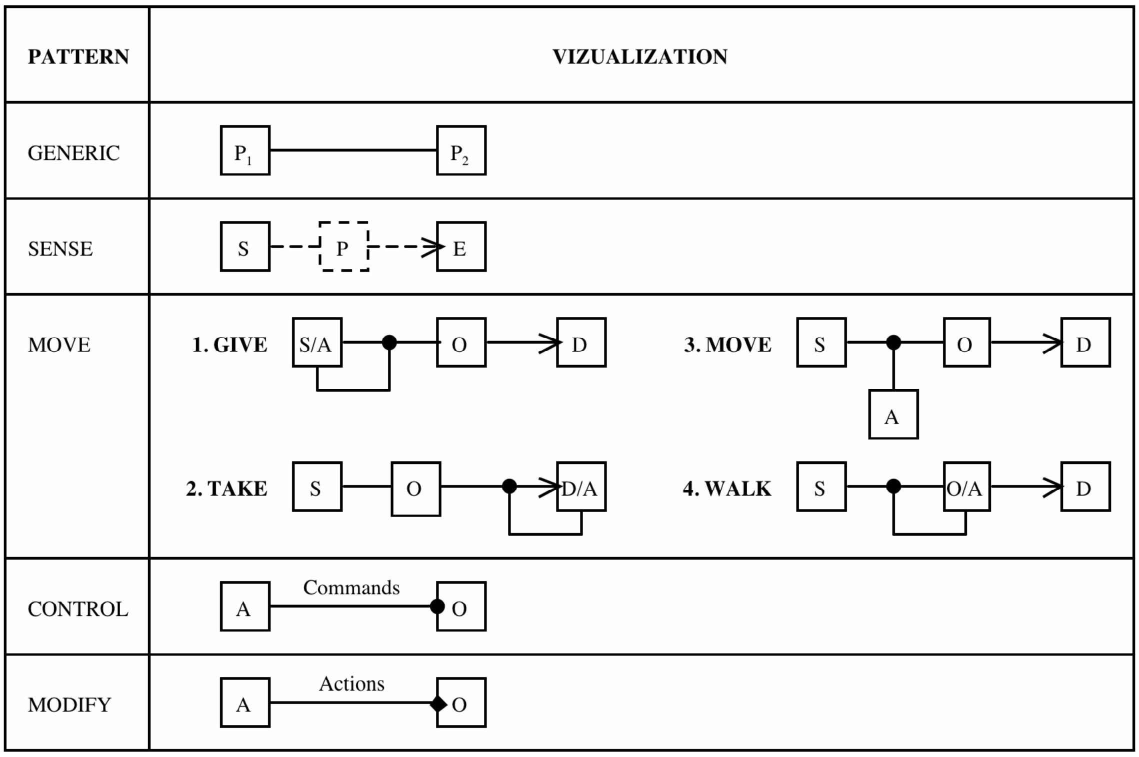

Figure 2 Visualized interaction primitives

An important choice is the level of detail. Components of word meanings can be the atomic building blocks, as in Schank's conceptual dependency diagrams (Dunlop 1990). Whole word meaning can be building blocks as in Sowa's conceptual graphs (Sowa 2000). Sentences can be building blocks when we want to represent chains of events and illustrate how events influence one another (Fillmore 1968; Fillmore 1977; Dik 1989; Nurcan, Etien et al. 2005; Andersen 2006). We have created our notation to support modeling of complex networks of interactions.

The present proposal distinguishes between participant representations and role representations. Distincitve arrow types are used to symbolize the interaction types. In Figure 2 we have shown our notation. Boxes symbolize participants and arrows symbolize interactions and the roles played in interactions. Participants can be things or events for the reasons explained in Section 2, and things include representations.

GENERIC is an unspecified interaction primitive that shows two interacting participants (P_1, P_2) without any assumptions about the type of interaction and the associated roles. GENERIC can be used in situations where the specific characteristics of an interaction are not yet clear.

SENSE is a pattern where an Experiencer (E) senses a Phenomenon (P) that is assumed to be a sensible characteristic of a Source (S). The convention is that the Phenomenon is described as it is or should be experienced by the Experiencer, not by the Source. It is thus a receiver-oriented conception of communication. This is indicated by the fact that the arrow runs from the Phenomenon to the Experiencer.

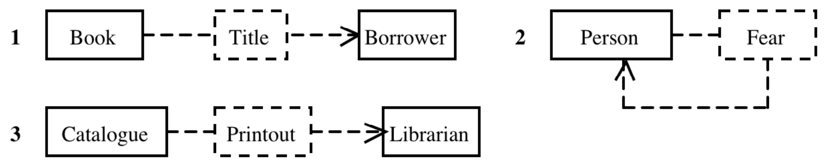

Figure 3 The SENSE primitive - Examples

Figure 3 contains three examples of SENSE. In example 1 a borrower senses (reads) the title of a book. The title is the phenomenon and the book is the Source. In example 2 a person senses his own anger. This example is a special form of SENSE where the experiencer is identical to the source. We call this SENSE variant FEEL. In example 3 a librarian senses (reads) a printed copy of a computerized catalogue.

CONTROL is a primitive where an Agent (A) influences the actions of an Object (O) by giving requests to the Object. The arrow runs from the Agent to the Object.

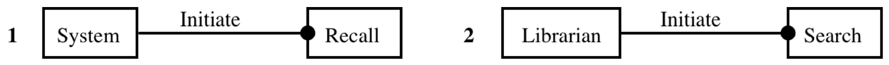

Figure 4 The CONTROL primitive - Examples

Figure 4 contains two examples of CONTROL. In example 1 a system initiates a recall process. In example 2 a librarian initiates a search process.

MOVE is a primitive where an Agent (A) moves an Object (O) from a Source (S) to a Destination (D). The arrow runs from the Source to the Destination. The box on the arrow represents the moved Object, for example a thing or a representation.

As illustrated in Figure 2 MOVE has four sub primitives each of which has a distinct Actor that performs the move action. In each sub primitive this Actor is related to the move by a an arrow. 1. The Source performs the move action. 2. The Destination performs the move action. 3. An external Agent performs the move action. 4. The moved Object performs the move action.

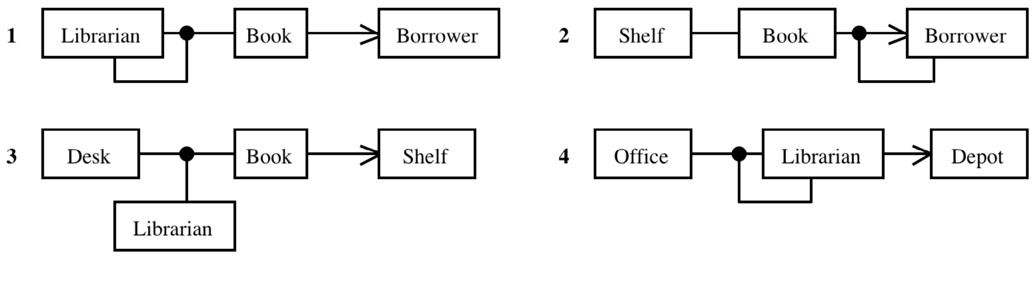

Figure 5 The MOVE primitive - Examples

Figure 5 contains four examples of MOVE. In example 1 a librarian (Source) gives a book to a borrower. In example 2 a borrower (Destination) takes a book from a shelf. In example 3 a librarian (external Agent) carries a book from a desk to a shelf. In example 4 a librarian (Object) goes down into a depot.

MODIFY is a primitive where one Agent (A) performs actions whose effects are that an Object (O) is changed. The Agent acts in a way that changes the Object. The arrow runs from Agent to Object.

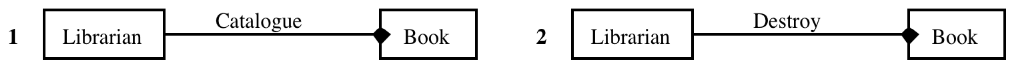

Figure 6 The MODIFY primitive – Examples

Figure 6 contains two examples of MODIFY. In example 1 a librarian catalogues a book. In example 2 a librarian destroys a book.

Case Study - Library Information Services

We performed a case study at a Danish public library in which an information search service offered to library users was changed (Bækgaard, Jørgensen et al. 2007). The case study was carried out as a part of a change project at the library. The purpose of the project was to identify and implement possible improvements of the library's mediation of library user's search for relevant information.

The service is based on communicative interaction between a library user and a librarian that engage in a dialogue about the library user's information needs, potential search terms, and the relevance of search results. The library user expresses needs for information and the librarian uses his understanding of these needs to search for information resources via library databases and Internet-based search engines.

The librarian interacts with the search systems in a way that is visible for the library user. The library user can see the librarian's queries and the corresponding search results. There is an important element of cognitive activities where the user and the librarian tries to understand the problem at hand and where they consider possibilities and reflect upon formulations and search results.

The librarian adds information resources (URLs etc.) that are relevant for the library user to a resource object that contains the selected resources. For example, if an Internet search engine returns one or more relevant URLs these are added to a digital text document using cut-and-paste, they are written on a piece of paper, or the screen shots on which they appear are printed. This implies that a resource object is comprised of unrelated digital text documents and pieces of paper.

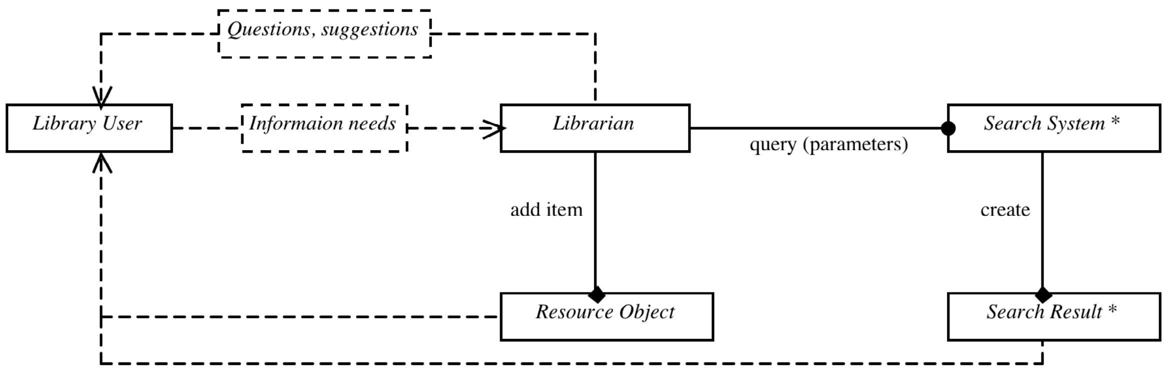

Figure 7 – Service 1 (current information search)

The diagram in Figure 7 represents the current service in terms of a service scenario. The element Library User represents a library user. The element Librarian represents a librarian. The element Search System represents a search system. The star means that more than one specific search system may play this role.

Three types of interaction primitives are used in the diagram in Figure 7. Two SENSE primitives represent the dialogue between the library user and the librarian. The library user expresses information needs and the librarian formulates questions and suggestions. SENSE is used because nothing is exchanged between the two parties. CONTROL is used to represent the interaction between the librarian and the search systems. The librarian controls the actions of the search systems by means of commands in terms of queries. MODIFY is used to represent the interaction between the librarian and the resource object and the interaction between the search systems and the search results. The motivation for this is that the librarian modifies the resource object by adding items to it. And the search systems modify the search results by creating these. SENSE is used to indicate that the library user can see the resource object and the search results.

In the current version of the service the resource object is an un-integrated collection of digital text documents, paper notes, and screen dumps. This is a major problem with the current service execution. The librarian has to create the resource object by means of cut-and-paste, handwriting, and printing operations. Consequently, the librarian creates the semantic integration of the search systems and the resource object. The IT systems do not support the integration in any way. Also, the resource object itself is heterogeneous and it is not internally integrated because it is composed of hand-written notes, printed screen-shots, and digital text documents. It is very difficult to reuse past search results and share the information represented by these among librarians and library users. The purpose of the innovation project was to create an IT tool to support the maintenance of resource objects.

The future service (Figure 8) is based on the same interaction between library user and a librarian as the current service. The library user expresses information needs and the librarian asks questions and suggests interpretations. The librarian uses search systems to search for resources that are relevant for the library user. The librarian and the library user share access to search results and to the resource object in which relevant resources are recorded.

A piece of software, a resource manager, supports maintenance of the resource object. When the librarian and the library user identify a relevant information resource (for example a URL) the librarian can use the resource manager to add the resource to the resource object.

Rather than using cut-and-paste or paper-and-pencil to maintain the resource object the librarian marks the relevant part of a search result and tells the resource manager to add the selection to the resource object that is now structured and fully digitalized.

Apart from simplifying the recording of relevant resources the Resource Manager integrates the otherwise non-integrated search systems. The service has two effects. It produces a resource object with information resources and it records this object in a resource database that can itself be accessed as a search system.

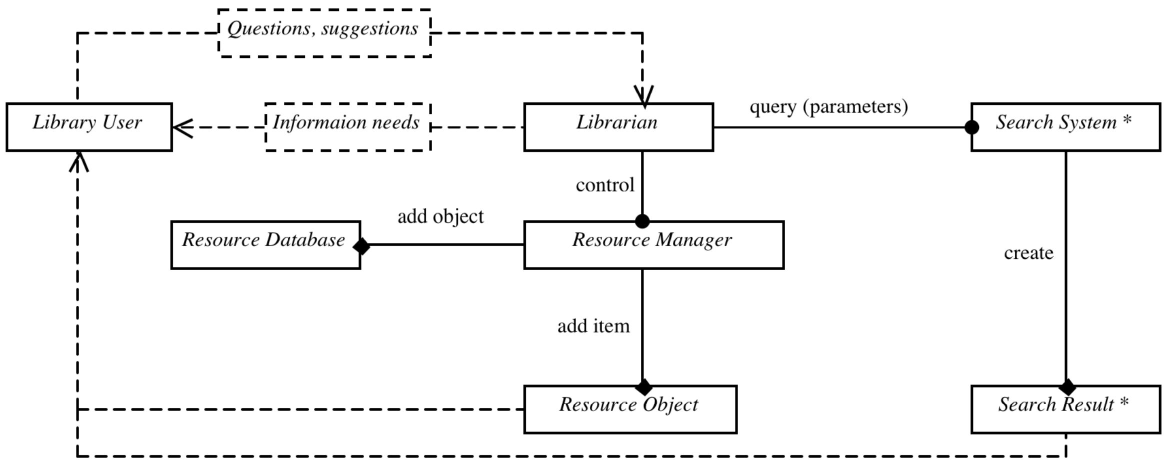

Figure 8 – Service 2 (future service)

The librarian uses the resource manager to select and modify resources from search results. This resource object is a structured and integrated digital document that contains structured resource items. The resource manager stores the final version of the resource object in a resource database that can itself be used as a search system. Consequently, the librarian can use the resource database as a memory of past service executions that may be searched and reused in future executions of the service.

Apart from simplifying the recording of relevant resources the resource manager integrates the otherwise non-integrated search systems. The service has two effects. It produces a resource object with information resources and it records this object in a resource database that can itself be accessed as a search system.

It is possible to give library users remote access to the resource manager and thereby to offer a version of the service that is executed solely by IT systems. This implies that a library user interacts directly with search systems and with the resource manager. Consequently, the library users must perform the cognitive activities that are performed by a librarian in the previously discussed versions of the service.

The diagram in Figure 9 represents the future self-service in terms of service scenario where the library user interacts directly with a search system and with the resource manager without the mediating help of the librarian.

From the librarian's point of view the service frees him or her from serving all library users. This may give the librarian more time to other work activities. A potential disadvantage is that the resource database may be "polluted" by resource objects of low quality. From the library user's point of the view the service can be executed any time without the library user having to be present at the library. A potential disadvantage is that the library user cannot benefit from the knowledge and experience of librarians.

The diagrams in the previous figures illustrate the use of interaction primitives to represent interaction patterns in socio-technical systems. The structure of resulting diagrams resemble the structure of UML collaboration diagrams in which a set of objects interact by means of messages (Rumbaugh et al. 1999). It would be possible to use a collaboration diagram to represent the interactions in the library. This presumes, however, that software objects represent all participants and that messages represent all interactions. The differences between the involved types of interactions would have to be indicated in other ways that are not supported directly by collaboration diagrams.

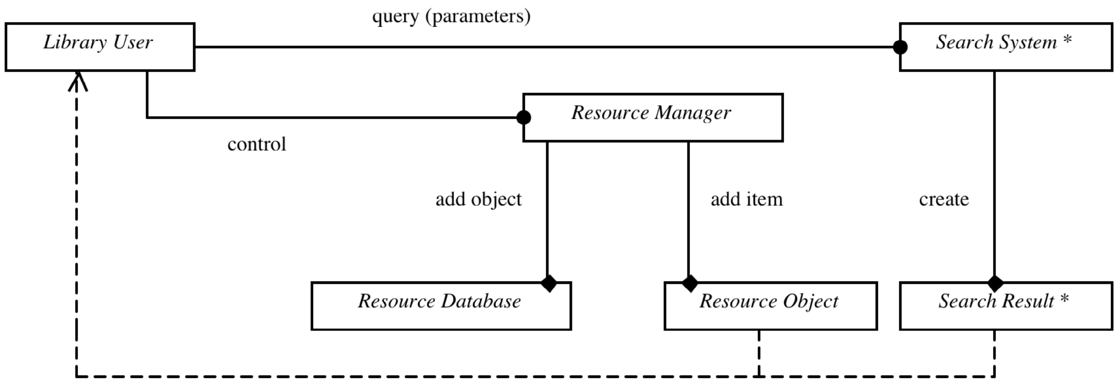

Figure 9 – Service 3 (futur self-service)

Our interaction diagrams can be used to supplement flow-oriented process models created by languages like BPMN (White 2004). Briefly, such models emphasize the flow of control through paths of activities in terms of action sequences, alternated actions, iterated actions, or concurrent actions. Our interaction diagrams repesent a supplementary view that highlights the structures of the participants and their mutual interactions. BPMN diagrams and interaction diagrams can supplement each other in a way that is similar to the way UML sequence and collaboration diagrams supplement each other. A combination of BPMN diagrams and interaction diagrams may be used to model the socio-technical aspects of information systems before sequence diagrams and collaboration diagrams are used to model the involved software.

Conclusion

We have defined interaction as an activity that involves two or more participants. At least one of the participants must be an agent. This implies that interaction plays two roles in information systems. First, interaction is a source of dynamics that causes an activity system to change. Second, interaction relates the elements of an activity system to each other in a way that supplements logical relations like contracts and functional dependencies. Viewed in this way interaction is a much more fundamental and general concept than its specialized siblings human-computer interaction and human-artifact interaction (Rogers, Sharp et al. 2002).

We have presented and discussed four interaction primitives that play important roles in information systems. The primitives can be used to characterize interaction within information systems and interaction between information systems and their environments because flow of objects and flow of requests occur both within information systems and between information systems and their environments.

We have illustrated how interaction primitives can be used to model interaction scenarios for socio-technical services in a library. We have modeled the service “book a librarian” in a structural manner that resembles UML collobaration diagrams (Booch, Rumbaugh et al. 1999). Rather than modeling the flow of activity in a swimlane manner we have used the interaction primivitives to emphasize the participating actors and resources and the types of interaction between these.

Future work includes case studies in which interaction scenarios are used to model socio-technical activity systems as a supplement to swimlane-based process modeling languages like BPMN (White 2004). And it includes formulation of methodological guidelines for the use of interaction scenarios.

References

1. Alter, S. (2006). The Work System Method. Connecting People, Processes and IT for Business Results. Larkspur, CA, Work System Press.
2. Andersen, P. B. (2006). "Activity-Based Design." European Journal of Information Systems 15: 9-25.
3. Bardram, J. E. and C. Bossen (2005). "Mobility Work – The Spatial Dimension of Collaboration at a Hospital." Journal of Computer Supported Cooperative Work 14(2): 131–160.
4. Booch, G., J. Rumbaugh, et al. (1999). The Unified Modeling Language User Guide, Addison-Wesley.

5. Bækgaard, L., J. B. Jørgensen, et al. (2007). On Industrial Use of Requirements Engineering Techniques. 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA'07). St. Goar, Germany.

6. Checkland, P. and S. Holwell (1998). Information, Systems, and Information Systems, Wiley.

7. Cockburn, A. (2001). Writing Effective Use Cases, Addison-Wesley.

8. De Marco, T. (1978). Structured Analysis and System Specification, Prentice-Hall.

9. Dehnert, J. (2002). Making EPCs Fit for Workflow Management. Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (EPK'02). Trier, Germany.

10. Dietz, J. L. G. (2006). Enterprise Ontology. Theory and Methodology, Springer-Verlag.

11. Dik, S. C. (1989). The Theory of Functional Grammar. Dodrecht, Foris Publications.

12. Dunlop, C. E. M. (1990). "Conceptual Dependency as the Language of Thought." Synthese 82(2): 275-296.

13. Fillmore, C. H. J. (1968). The Case for Case. Universals in Linguistic Theory. E. Bach and R. T. Harms. London, New York, Sydney, Toronto, Holt, Rinehart and Winston: 1-90.

14. Fillmore, C. H. J. (1977). The Case for Case Responded. Syntax and Semantics: 8. Grammatical relations. P. Cole and G. M. Sadock. New York, Academic Press: 59-81.

15. Goldkuhl, G. (1996). Generic Business Frameworks and Action Modeling. First International Workshop on Communication Modeling. Tilburg, The Netherlands.

16. Goldkuhl, G. and M. Lind (2004). The Generics of Business Interaction - Emphasizing Dynamic Features Through the BAT Model. International Working Conference on the Language-Action Perspective on Communication Modelling (LAP´04) New Brunswick, NJ, USA.

17. Hevner, A. R., S. T. March, et al. (2004). "Design Science in Information Systems Research." MIS Quarterly 28(1): 75-105.

18. Lübke, D., T. Lüecke, et al. (2006). Using Event-Driven Process Chains for Model-Driven Development of Business Applications. XML Integration and Transformation for Business Process Management (GI-Workshop XML4BPM'06). Passau, Germany.

19. March, A. T. and G. F. Smith (1995). "Design and Natural Science Research on Information Technology." Decision Support Systems 15: 251-266.

20. Nurcan, S., A. Etien, et al. (2005). "A Strategy Driven Business Process Modeling Approach." Business Process Management Journal 11(6): 628-649.

21. Rogers, Y., H. Sharp, et al. (2002). Interaction Design. Beyond Human-Computer Interaction, John Wiley & Sons, Inc.

22. Rumbaugh, J., I. Jacobson, et al. (1999). The Unified Modeling Language Reference Manual, Addison-Wesley.

23. Sowa, J. F. (2000). Knowledge Representation: Logical, Philosophical, and Computational Foundations. Pacific Grove, CA, Brooks Cole Publishing Co.

24. White, S. A. (2004). Introduction to BPMN, BPMN.ORG.