Interaction in Information Systems – Beyond Human Computer Interaction

Lars Bækgaard

Department of Business Studies, Aarhus School of Business, Denmark (www.asb.dk)
Email: lab@asb.dk

Abstract
The purpose of this paper is to discuss and analyze the role of interaction in information systems. Interaction represents dynamic relations between actors and other elements in information systems. We introduce a semi-formal notation that we use to describe a set of interaction patterns and we illustrate how the notation can be used to describe mediated interaction. We use the interaction patterns to evaluate a set of modeling languages. No single language supports all relevant aspects of interaction modeling. We use the interaction patterns to identify to general and supplementary forms of interaction—interaction based on exchange of objects and interaction based on exchange of commands. None of the modeling languages that we analyze support both forms in a rich way.

Keywords. Action. Interaction. Information systems. Interaction patterns.

1 Introduction

The purpose of this paper is to discuss interaction in information systems. Interaction plays two roles in information systems. First, it can be viewed as "dynamic glue" that relates the elements in an information system to each other. Second, it is a source of change. Exchange of information between two elements in an information system is a source of internal change. Exchange of information between an information system and its environment is a source of change in the relations between an information system and its environment.

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 development. Many information systems contain IT-systems that perform selected actions. All information systems exist within a larger system of activity.

Interaction is a central concept in the area of human-computer interaction that focuses on the interaction between human beings and IT-systems (Rogers, Sharp et al. 2002). Human-computer interaction is, however, only one specialized form of interaction in information systems. The interaction between two human actors that interact cannot be captured in a rich manner by concepts that focus on human-computer interaction. Our notion of interaction is far more general than the notion of human-computer interaction. We focus on interaction between actors and interaction between actors and things or information.

Many development methods support interaction modeling in restricted an somewhat ad hoc ways. For example, OOAD uses use case modeling to model interaction between a system and actors in its environment (Mathiassen, Munk-Madsen et al. 2000). And RUP uses interaction diagrams and state charts to model interaction between software components (Jacobson, Booch et al. 1999). In Section 5 we demonstrate that existing modeling languages support interaction modeling in rather arbitrary and incomplete ways.

The paper is organized as follows. In Section 2 we define the notion of activity systems on which the papers' is based. In Section 3 we present and discuss five interaction patterns. In Section 4 we extend the interaction in order to support modeling of mediated interaction. In Section 5 we discuss and analyze a set of modeling languages with respect to their support for

interaction modeling. In Section 6 we conclude the paper and suggest directions for future work.

2 Activity systems

The purpose of this section is to present and discuss a set of concepts that can be used to understand and characterize information systems as activity systems. This makes it possible for us to think and talk about activity in business systems, information systems, and IT-systems in a uniform manner. And it makes it possible for us to identify interactive actions as the dynamic relations in information systems.

Activity plays important roles in information systems and their environments. Organizations depend on material activities and information activities that deal with movement, manipulation, and consumption of things and information and they depend on coordination activities that deal with requests for, agreements about, control of, and evaluation of work processes (Denning and Medina-Mora 1995). We view an activity system as a system that is composed of mutually related actors, things, and information.

Actors carry out activities. They manipulate things and information in favor of customers that benefit from the results created by the activities (Checkland 1981). Actors use, create, manipulate, and exchange things and information with or without the support of tools. For example, employees create products and customers receive invoices. We use the term actor as a reference to both persons and IT-systems (Rose, Jones et al. 2003). There are many fundamental differences between persons and IT-systems (Weizenbaum 1984; Dreyfus and Dreyfus 1986; Winograd and Flores 1986). However, during the design process where the borders between the activity of persons and the activity of IT-systems are discussed it is can be useful to focus on an important similarity between persons and IT-systems, namely their ability to perform useful (material) actions.

Things are physical objects like materials, tools, buildings, products etc. Things have properties that can be modified by the actions taken by actors.

Information is a problematic concept in the sense that no common agreement about its definition exists (Mingers 1995). Information can be viewed as signs that may be interpreted and attached to meaning by persons (Checkland and Holwell 1998). This implies that information have a material, thing-like side that can be manipulated and they have an immaterial side that resides in the heads of the interpreting persons. When persons try to express the meaning of information they do it by means of new signs whose meaning is created by a new process of interpretation. When we refer to information in the discussion of interaction patterns in the following section we refer to the material side that can be manipulated by both persons and IT-systems.

A *logical relation* relates two or more elements in an activity system. Actors, things, and information can be related by means of logical relations. A static relation can be viewed as a contract like a marriage, the ordering of a book, and a membership agreement. Logical relations may relate parts to wholes and phenomena to descriptions. Usually, logical relations are relatively stable but they are less stable than the related phenomena. For example, the fact that a book is temporarily borrowed by a borrower at a library may be viewed as logical relation between the book and the borrower. The relation may exist for less than one month while the borrower and the book may exist for years.

Interaction is a dynamic relation between two or more elements in an activity system. At least one of the elements must be an actor. An actor that interacts with one or more elements is dynamically related to the elements. Actors may interact with actors by, say, exchanging things and information. Actors may interact with things and information by, say, modifying the things and information. Human actors and IT-systems interact when they carry out activities. An activity system is dynamic because the actors perform actions. Most actions have interactive characteristics in the sense that they involve two actors or one actor and one object.

An *IT-actor* is a coherent unit of software and hardware. It is composed of sensors, actuators, memory, and action potential. An IT-actor uses sensors to register (the material side of) information about its environment in terms of selected events, measurement values etc. An IT-actor uses its memory to remember information about selected aspects of its history. An IT-actor can influence its environment by means of actuators that deliver information, turns switches of/on etc. An IT-actor can perform actions in terms of, say, modification and movement of information.

We view *information systems* as activity systems that capture, move, store, retrieve, manipulate, and display information (Alter 1999). Activity systems plays a double role in information systems. First, information systems *are* activity systems. They exist within larger activity systems that involve manipulation of both things and information. Increasingly, information systems becomes tied to things and actors by means of chips that are attached to things and actors. These chips may contain information about the corresponding things and their history. A bottle of milk may contain information about the origin and production of the milk. A parcel may contain information about its destination and purpose. This implies that things play important roles for and in information systems even though information systems do not as such contain things. Second, information process information *about* activity systems (and other relevant phenomena). This may include information about actors, things, information, logical relations, and interaction within and outside the activity system within which the information system resides.

3 Interaction patterns

In this section we present and discuss five interaction patterns that can be used to enable discussions about the roles of human actors and IT-actors in information systems. As argued in the previous section we view interaction as a dynamic relation between two elements in an activity system. At least one of the elements must be an actor. In order to visualize the patterns we have created an informal notation.

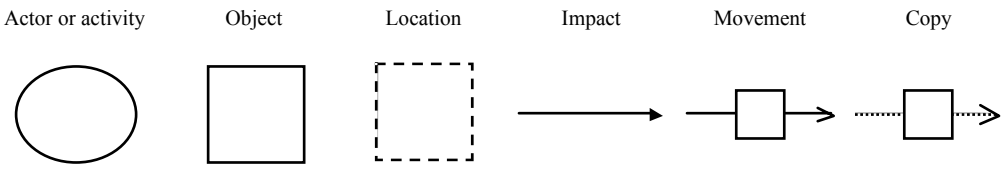

Figure 1 Notation

Our notation contains five symbols (Figure 1). We treat actors and activities as mutually replaceable phenomena because a specific activity must be performed by one or more actors. An object is a thing or a piece of information. A location is a physical or informational place that can contain things and/or information. An impact arrow symbolizes a modifying action that an actor makes to another element. A movement arrow symbolizes an object that is

physically moved. A copy arrow symbolizes that an actor gets access to an object that is a copy of aspects of another object.

We would like to emphasize three characteristics of our notation and the corresponding interaction patterns. *First*, our notation is not intended to be a modeling language that can be used to create large activity models. It is intended to serve as an semi-formal language that we can use to identify and discuss properties of modeling languages and modeling approaches.

Second, our notion of interaction is based on a basic set of uni-directional interaction patterns. 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 patterns can be combined to represent bi-directional interactions.

Third, we do not claim that our five interaction patterns constitute a complete set of patterns that cover all imaginable forms of interaction. But as will be revealed in our discussion of modeling languages in Section 5 our patterns cover the overall interaction forms that are used in the most popular modeling languages.

Actors can sense objects and use sensory information to create understandings of objects. Persons can use their senses to hear, see, smell, taste, and feel. Persons can see shapes and fell hardness. Person can read texts. Users can read information on interfaces. IT-actors can use their sensors to sense their environments.

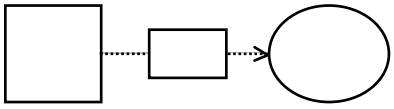

Figure 2 SENSE - Pattern

SENSE is an interaction pattern where an actor senses aspects of an object. In Figure 2 we have used our notation to show the general form of SENSE. The object on the arrow represents the information that the actor senses about the object. This implies that the aspects of the sensed objects that the actor senses are represented by another object.

Figure 3 SENSE - Examples

Figure 3 contains two examples of SENSE. Example 1 represents a customer that senses the sound of a radio. The radio is the sensed object. The sound is a dynamic object of the radio. Example 2 represents a customer that gets a copy of a digital product. For example, this copy may represent a piece of software or a piece of music that the customer has downloaded from a web site.

An actor can give an object to another object. For example, employees can deliver products to customers, employees can inform customers, customers can inform customers, and IT-actors can deliver information to business intelligence software.

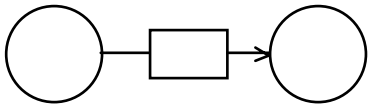

Figure 4 GIVE –Pattern

GIVE is an interaction pattern where two or more actors exchange objects. In Figure 4 we have used our notation to show the general form of GIVE. The object that is given can be a thing or a piece of information.

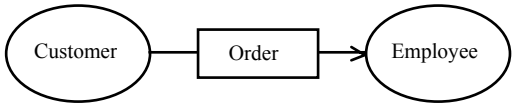

Figure 5 GIVE - Example

Figure 5 contains an example of GIVE. The example represents a customer that gives an order to an employee.

Actors can move things and information. Employees can send invoices. Customers can send orders. Employees can move products. Customers can bring products them to their homes. Persons can use their bodies to move objects. IT-actors can use their actuators to move objects.

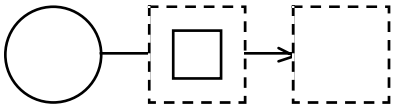

Figure 6 MOVE - Pattern

MOVE is an interaction pattern where an actor moves an object from one location to another. In Figure 6 we have used our notation to show the general form of MOVE. The actor moves the object from its initial location (the dotted rectangle on the arrow) to a new location.

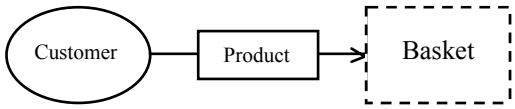

Figure 7 MOVE - Example

Figure 7 contains an example of MOVE. The example represents a customer that moves a product to a basket. The products initial location is not defined in this example. It could be, say, a shelf in a warehouse.

Actors can modify objects. They can change the properties of things. They can combine things into new things. They can divide things into sets of things. Employees can modify raw materials into products. Actors can modify information. Journalists can write articles. Programmers can modify software. Persons can articulate verbal expressions. Persons use their bodies to modify objects. IT-actors can use their actuators to modify objects.

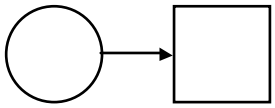

Figure 8 MODIFY - Pattern

MODIFY is an interaction pattern where an actor modifies an object. In Figure 8 we have used our notation to visualize the general form of MODIFY. The actor acts in a way that

changes the object. The arrow symbolizes the various forms of modifying actions that the actor may perform.

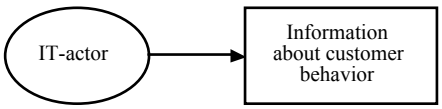

Figure 9 MODIFY - Example

Figure 9 contains an example of MODIFY. The example represents an IT-actor that manipulates information about customer behavior.

Actors can control actors. For example, a department manager may ask an employee to undertake a certain task, or a business intelligence system may ask for certain pieces of information in a database. Actors can initiate activity, they can redirect a flow of activities, and they can suspend and terminate etc.

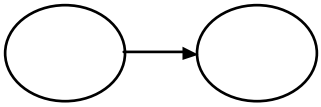

Figure 10 CONTROL - Pattern

CONTROL is an interaction pattern where an actor controls another actor. In Figure 10 we have used our notation to show the general form of CONTROL where one actor uses commands to control the actions of another actor.

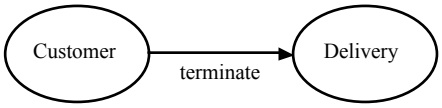

Figure 11 CONTROL - Example

Figure 11 contains an example of CONTROL. The example represents a customer that terminates a delivery activity.

An IT-actor can be controlled by a set of commands that the IT-actor can respond to. Such a command set may include commands like trigger, pause, resume, terminate, and other commands. The command set of a controlled IT-actor defines an action space for the controlling actor. The action space depending on the command set and the IT-actor's responses to each command.

Goldkuhl and Ågerfalk use the term actability to refer to the action space that an information system offers to actors in its environments. According to this view information systems consist of action potential, actions, and action memory. Information systems can support actions, they can perform actions, and they can remember past actions. In this perspective information systems are viewed as actors that perform business actions (Goldkuhl and Ågerfalk 1998). In Section 4 we demonstrate that actors that can react to a relevant set of commands is a crucial element if we want to mediate the various forms of interaction.

Actors that modify objects with or without mediating tools is a central theme in activity theory (Leontiev 1978; Engeström 1991). Consequently, the structure and intention of the pattern MODIFY is very close to the central ideas of activity theory in which the acts of actors cause objects to be changed. Activity theory has inspired several theories about systems

development (Christiansen 1990; Korpela, Soriyan et al. 2000; Korpela, Mursu et al. 2002; Roque, Almeida et al. 2003).

Our interaction patterns are based on the material aspects of (inter)actions—actions that are characterized by actors that interact with physical objects. As argued by Goldkuhl (2001) many actions are characterized by a combination of material and communicative aspects (Goldkuhl 2001). The actions that underlie our five interaction patterns have both material and communicative aspects. However, when we try to represent (inter)actions by means of explicit notations it would be very problematic to claim that the representations capture the communicative aspects that are inherently related to the actor's interpretations and perceptions.

The distinction between material and material and communicative actions can be used to clarify one of the most important differences between human actors and IT-actors. Both categories of actors can perform material actions that involve things and (the material aspects of) information. However, IT-actors cannot perform communicative actions. This implies that the material actions performed by IT-actors cannot be guided by unforeseen interpretations of the situation at hand unless one or more human actors participate and use their interpretations and judgments to guide the IT-actors.

4 Mediated interaction

In this section we use the interaction patterns from the previous section to discuss potential for mediation of interaction. It is not our intention to present a complete analysis of mediated interaction. It is our intention to illustrate that our notation can be used to define a set of patterns for mediated interaction. Further work is necessary in order to make a more comprehensive analysis of potential patterns for mediated interaction.

Interaction can be direct or it can be mediated by persons, IT-actors, tools, machines etc. Direct interaction involves contact between the involved elements (actors, things, information). When an employee moves a product by hand the interaction between employee and product is direct. The interaction between an employee and a customer is direct when they interact "face-to-face".

Mediated interaction involves a third-party (actor or object) between the interacting parties. When an employee uses a truck to move a product the interaction between employee and product is mediated by the truck. When an employee exchanges information with a customer in a chat room the interaction between employee and customer is mediated by the chat room. Mediated interaction introduces new interactions between actors and mediators. When two actors communicate in a chat room they must interact with the chat system in order to interact with each other. The original interactions between the actors are the primary interactions. The interactions between actors and mediators are secondary interactions that are introduced as consequence of the mediation of the original interactions.

IT-actors can be used to mediate interaction. Word processing software mediates interaction between writer and text. Chat software mediates interaction between the communicating actors. E-commerce software mediates shared events in terms of business transactions that involve businesses, products, and customers.

Mediated interaction plays important roles in material and coordination activities. Material activities may be mediated by digitally controlled machinery and many coordination activities

are communicative actions in which actors express requests, requirements, contracts, and evaluations. In the following examples we use the pattern CONTROL to facilitate mediation of SENSE, GIVE, MOVE, MODIFY, and CONTROL.

The basic idea of SENSE is that an actor senses aspects of an object. Often, actors gets access to aspects of objects without any direct interaction with the objects.

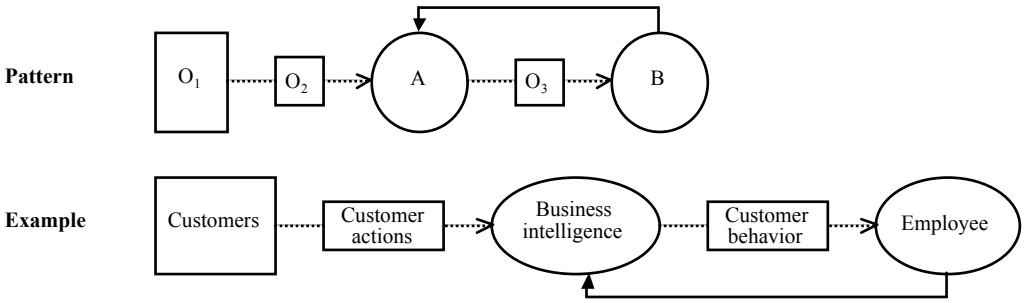

Figure 12 SENSE – Mediated

Sensing of things and information can be mediated as illustrated in the pattern in Figure 12. The actor B senses aspects O_3 of the object O_1 indirectly via the actor A. In the example, an employee senses customers indirectly via business intelligence information about customer behavior. The business intelligence actor senses the customers directly in terms of their behavior. The employee can control the business intelligence actor by means of commands that influences the generated information about customer behavior.

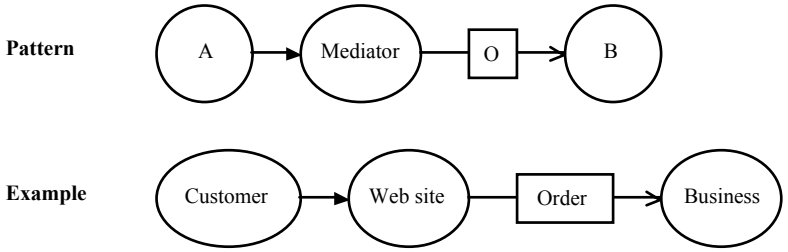

Figure 13 GIVE – Mediated

Exchange of things and information can be mediated as illustrated in the pattern in Figure 13. The actor A gives the object O to the actor B indirectly via a mediator. The actor A controls the mediator that gives the object O to the actor B. In the example, a customer gives an order to a business via the business' web site that acts as a mediating actor.

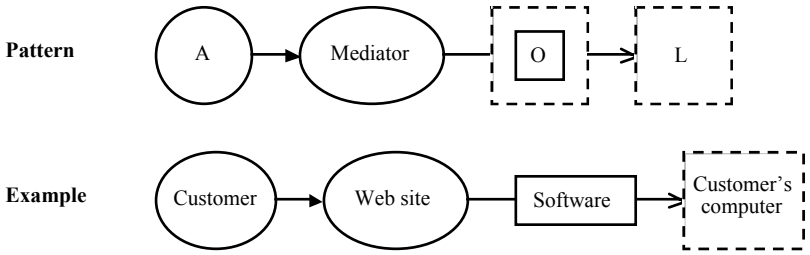

Figure 14 MOVE – Mediated

Movement of things and information can be mediated as illustrated in the pattern in Figure 14. The actor A moves the object O to the location L indirectly via a mediator. The actor A controls the mediator that moves the object O to the location L. In the example, a customer uses a web site to download software to his computer.

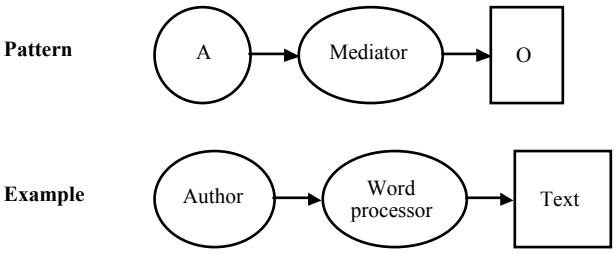

Figure 15 MODIFY – Mediated

Modification of things and information can be mediated as illustrated in the pattern in Figure 15. The actor A modifies the object O indirectly via a mediator. The actor A controls the mediator that modifies the object O. In the example, an author modifies a text via a word processor.

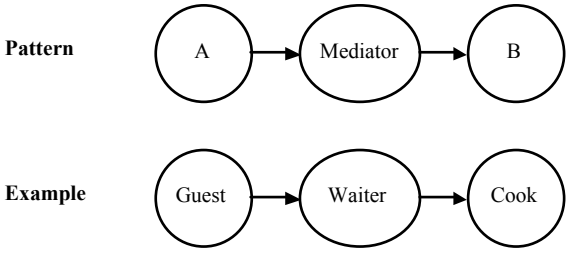

Figure 16 CONTROL – Mediated

Control of actors can be mediated as illustrated in the pattern in Figure 16. The actor A controls the Actor B indirectly via a mediator. The actor A controls the mediator that controls the actor B. In the example, a restaurant guest controls the restaurant's cook via a waiter.

It should be clear that CONTROL is a very important pattern if we want to understand interaction and mediation of interaction. The notion of a controllable actor that responds to selected commands is inevitable if we want to model mediated interaction. Which commands can an actor receive and execute? What is the set of actions that these commands enables an actor to perform?

5 Modeling languages

In this section we discuss a set of modeling languages with respect to their support of interaction modeling. We describe the basic idea of each language and evaluate its support for modeling of the five interaction patterns.

One of the most important results of the preceding analysis of interaction and mediation is that a modeling language must combine two general types of flow modeling in order to support interaction modeling in a rich way. First, a modeling language must support modeling of object flow in order to support modeling of SENSE, GIVE, and MOVE. Second, a modeling language must support modeling of command flow in order to support modeling of MODIFY and CONTROL.

A *use case* represents a coherent unit of visible functionality that is offered by a system to one or more actors (Rumbaugh, Jacobson et al. 1999; Kruchten 2000; Cockburn 2001). A use case defines the behavior of some system in terms of its activity and interaction with actors in its environment. The behavior is triggered by an action performed by an actor that interacts with the use case in order to obtain some goal. A *use case specification* defines interaction in terms of a sequence of actions. Some actions are carried out by actors. Other actions are carried out

by use cases. Use case specifications can be defined in terms of algorithms, interaction diagrams, state chart diagrams etc. A *use case diagram* relates one or more use cases to one or more actors. Each relation represents potential interaction. Use cases diagrams do not support modeling of SENSE, GIVE, MOVE, and MODIFY. The reason is that use case diagrams focus solely on interaction between actors. Use case specifications may refer to information objects and consequently the specifications may support the modeling of SENSE, GIVE, MOVE, and MODIFY. This, however, is not a quality of use case diagrams but of the language(s) used for use case specifications. Use case diagrams support modeling of CONTROL in the sense that it is implicitly assumed that the external actor controls the use case.

Object-oriented models like state chart diagrams and interaction diagrams are based on the notion of encapsulated objects that interact by sending messages to each other (Rumbaugh, Jacobson et al. 1999). A message is a command that activate a set of actions in the receiving object. Messages can contain parameters with information that is exchanged but the overall flow paradigm is based on a flow of commands. A *state chart diagram* can be used to define rules for the sequencing of messages send to and from state-dependent software objects. In a given state the object can respond to a specific set of messages. All other messages will be rejected. *Interaction diagrams* model interaction in terms of messages that are passed among two or more software objects. State chart diagrams and interaction diagrams support modeling of SENSE. When an object receives a message it can be interpreted as a sensing of the content of the message. GIVE can be modeled in terms of information that is exchanged as parameters to messages. Modeling of MOVE is not supported because there are no references to locations. Interaction diagrams do not as such support modeling of MODIFY. A state chart diagram can be used to model MODIFY in terms of objects that change states. State chart diagrams and interaction diagrams supports modeling of CONTROL in the sense that an object that sends a message to another object the first controls the receiving object. Each object type can be defined in order to respond to any number of commands (messages).

A *data flow diagram* represents a set of information activities that manipulate and exchange information (De Marco 1978). The environment is represented by a set of information sources and sinks. Information can flow between activities and it can be stored in files. Data flow diagrams support modeling of SENSE in terms of input information that is received by information activities or by an information sinks. Data flow diagrams support modeling of GIVE with the restriction that only information can be exchanged between information activities. Data flow diagrams so not support modeling of MOVE because they do not contain any references to locations. Data flow diagrams do not as such support modeling of MODIFY. They can indicate modification of the information that is passed as flows between activities. And they can indicate modification of files. Data flow diagrams do not support modeling of CONTROL because information activities cannot send commands to each other. Modern Structured Analysis extends data flow diagrams with essential events that trigger a subset of the activities in a dataflow diagram in a way that is similar to use cases (Yourdon 1989).

An *activity diagram* represents a set of activities (Rumbaugh, Jacobson et al. 1999) that can be executed concurrent or sequentially in sequences, alternations, iterations. The termination of one activity may trigger the initiation of another activity. Activities can be located in swim lanes that define their locations. Software objects with changing states can be passed between activities. Activity diagrams support modeling of SENSE in terms of the input objects that activities have access to. Modeling of GIVE is supported in the sense that activities can

exchange software objects. Modeling of MOVE is supported in terms of swim lanes that define locations for the activities that are included in an activity diagram. Modeling of MODIFY is supported in the sense that exchanged objects may have visible states that change as they passed from activity to activity. CONTROL can only be modeled in a rather primitive manner that is not based on explicit commands. The termination of one activity may trigger initiation of another activity.

Event-activity diagrams are extensions of activity diagrams (Bækgaard 2004). 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. They contain a notation for object exchange between parallel activities. Events are occurrences without duration (Jackson 1983). Shared events have multiple participants. Therefore, they can be used to coordinate the activities of two or more interacting actors. Shared events define synchronization points for two or more activities. Events can be used to trigger and terminate activities. Event-activity diagrams have two advantages over activity diagrams with respect to interaction modeling. First, event-activity diagrams support asynchronous object sharing among independent activities. This gives a richer support for modeling of GIVE. Second, event-activity diagrams are based on named, shared events with participants. This implies that one activity can control other by requesting a named event that can be interpreted as a command.

Business-oriented models like EPC-diagrams (Dehnert 2002) and BPMN-diagrams (White 2004) use events to control the sequence in which activities are executed in a workflow like manner. These approaches do not support the CONTROL pattern by means of explicit command flows. They are based on a workflow paradigm in which the major focus is on the sequencing and concurrent execution of activities within a larger business process. The basic mechanisms for controlling the sequencing of activities are implicit transfer of control from a terminating activity to a triggered activity *and* conditioned branching among alternative sequences (van der Aalst, ter Hofstede et al. 2000). In this sense EPC and BPMN are based on specializations of CONTROL that use a restricted and implicit set of commands.

6 Conclusion

We have defined interaction as an activity that involves two or more elements. At least one of the elements must be an actor. 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 five interaction patterns that play important roles in information systems. For each of these patterns we discussed potential mediation of the corresponding forms of interaction. An understanding of such mediation is essential if we want to utilize the mediating potential that is an inherent property of information technology. The patterns can be used to characterize interaction within information systems and interaction between information systems and their environments because flow of objects and flow of commands occur both within information systems and between information systems and their environments.

We have analyzed a set of modeling languages with respect to their support of interaction modeling. The conclusion is that none of the analyzed languages support modeling of all the interaction patterns and the potential mediation of the corresponding interaction forms.

One of the most important reasons for this is that none of the analyzed languages support the modeling of both object flow and command in a rich manner. In order to support modeling of all five interaction patterns it is necessary to combine the two types of flow modeling. The object-based languages subordinate flow of objects to flow of commands. Dataflow diagrams and activity diagrams favor flow of objects. Dataflow diagrams do not support flow of commands. Activity diagrams support flow of commands in a rather limited manner. The event-based languages use events to support flow of commands in a more sophisticated way. Event-activity diagrams utilize shared events with multiple participants to support a general flow of commands where one actor can impose a command on two or more other actors. Business-oriented diagrams like EPC-diagrams and BPMN-diagrams are based on the workflow paradigm that primarily focuses on the sequencing of activities within larger business processes.

Future work includes further analysis of and experimentation with interaction patterns and their support for analysis and design of IT-mediated human works systems. In particular, the relations between general interaction, IT-mediation, and human-computer interaction must be articulated and analyzed. How do we decide what actions IT-actors are going to perform? Whenever an actor is supposed to do something IT-actors can play two roles. First, they can play the role of the actor. Second, they can fully or partially mediate the actions. It seems likely that the interaction patterns can be used to supplement modeling techniques like task descriptions (Lauesen 2003) and activity cases (Bækgaard 2005).

Also, future work includes a comparison of the interaction patterns with models of more complex interaction patterns like the BAT model of business interaction (Goldkuhl and Lind 2004), DEMO (van Reijswoud, Mulder et al. 1999), and ActionLoops (Weigand and de Moor 2001). All these approaches view business interaction as a composite process that involve several basic interactions. For example, DEMO divides business transaction into three phases (order, execution, result) each of which involve one or more basic interactions.

References

ALTER, S. (1999). "A General, Yet Useful Theory of Information Systems." *Communications of the Association for Information Systems* **1**(13).

BÆKGAARD, L. (2004). "Event-Based Activity Modeling". *ALOIS'04 - Action in Language, Organisation and Information Systems*. Linköping, Sweden.

BÆKGAARD, L. (2005). "From Use Cases to Activity Cases". *ALOIS'05. Action in Language, Organisation and Information Systems*. Limerick, Ireland.

CHECKLAND, P. (1981). *Systems Thinking, Systems Practice*, Wiley.

CHECKLAND, P., HOLWELL, S. (1998). *Information, Systems, and Information Systems*, Wiley.

CHRISTIANSEN, E. (1990). "On Organizational Competence in Systems Development. An Activity Approach". *Organization Competence in System Development. A Scandinavian Contribution*. G. Bjerkness, Dahlbom, B., Studentlitteratur.

COCKBURN, A. (2001). *Writing Effective Use Cases*, Addison-Wesley.

DE MARCO, T. (1978). *Structured Analysis and System Specification*, Yourdon.

DEHNERT, J. (2002). "Making EPCs fit for Workflow Management". *EPK'2002*. Trier, Germany.

DENNING, P. J., MEDINA-MORA, R. (1995). "Completing the Loops." *Interfaces*.

DREYFUS, P. J., DREYFUS, S. E. (1986). *Mind Over Machine*. New York, The Free Press.

ENGESTRÖM, Y. (1991). "Developmental Work Research: Reconstructing Expertise Through Expansive Learning". *Human Jobs and Computer Interfaces Conference*. University of Tampere.

GOLDKUHL, G. (2001). "Communicative vs Material Actions: Instrumentality, Sociality and Comprehensibility". *Sixth International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2001)*, Montreal, Canada.

GOLDKUHL, G., LIND, M. (2004). "The Generics of Business Interaction - Emphasizing Dynamic Features Through the BAT Model". *Ninth International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2004)*, New Brunswick, NJ, USA.

GOLDKUHL, G., ÅGERFALK, P. J. (1998). "Action within Information Systems: Outline of a Requirements Engineering Method". *REFSQ'98 - 4th International Workshop on Requirements Engineering: Foundation for Software Quality*. Pisa, Italy.

JACKSON, M. (1983). *System Development*, Prentice Hall International.

JACOBSON, I., BOOCH, G, RUMBAUGH, J. (1999). *The Unified Software Development Process*, Addison-Wesley.

KORPELA, M., MURSU, A., SORIYAN, H. A. (2002). "Information systems development as an activity." *Computer Supported Cooperative Work*, **11**: 111-128.

KORPELA, M., SORIYAN, H. A., OLUFOKUNBI, K. C. (2000). "Activity Analysis as a Method for Information Systems Development." *Scandinavian Journal of Information Systems*, **12**.

KRUCHTEN, P. (2000). *The Rational Unified Process - An Introduction*, Addison-Wesley.

LAUESEN, S. (2003). "Task Descriptions as Functional Requirements". *IEEE Software*.

LEONTIEV, A. N. (1978). *Activity, Consciousness, and Personality*, Englewood Cliffs: Prentice-Hall.

MATHIASSEN, L., MUNK-MADSEN, A., NIELSEN, P. A., STAGE, J. (2000). *Object-Oriented Analysis and Design*, Marko.

MINGERS, J. C. (1995). "Information and Meaning: Foundations for an Intersubjective Account", *Information Systems Journal*.

ROGERS, Y., SHARP, H., PREECE, J. (2002). *Interaction Design. Beyond Human-Computer Interaction*, John Wiley & Sons, Inc.

ROQUE, L., ALMEIDA, A., DEFIGUEIREDO, A. D. (2003). "Context Engineering: An IS Development Approach". *ALOIS'03. Action in Language, Organisations, and Information Systems*. Linköping, Sweden.

ROSE, J., JONES, M., TRUEX, D. (2003). "The Problem of Agency: How Humans Act, How Machines Act". *ALOIS'03. Action in Language, Organisations, and Information Systems*. Linköping, Sweden.

RUMBAUGH, J., JACOBSON, I., BOOCH, G. (1999). *The Unified Modeling Language Reference Manual*, Addison-Wesley.

VAN DER AALST, W. M. P., TER HOFSTEDE, A. H. M., KIEPUSZEWSKI, B., BARROS, A. P. (2000). "Workflow Patterns". *BETA Working Paper Series*. Eindhoven, Eindhoven University of Technology.

VAN REIJSWOUD, V. E., MULDER, H. B. F., DIETZ, J. L. G. (1999). "Communicative Action Based Business and Information Systems Modelling with DEMO". *International Journal of Information Systems*.

WEIGAND, H., DE MOOR, A. (2001). "A Framework for the Normative Analysis of Workflow Loops". *Sixth International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2001)*, Montreal, Canada.

WEIZENBAUM, J. (1984). *Computer Power and Human Reason*, Penguin Books.

WHITE, S. A. (2004). "Introduction to BPMN", WWW.BPMN.ORG.

WINOGRAD, T., FLORES, F. (1986). *Understanding Computers and Cognition*, Addison-Wesley.

YOURDON, E. (1989). *Modern Structured Analysis*, Prentice-Hall.