IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP5186

This paper appears in *Managing Modern Organizations Through Information Technology*, Proceedings of the 2005 Information Resources Management Association International Conference, edited by Mehdi Khosrow-Pour.

A Framework for Language-Action Modeling in UML

Peter Rittgen

University College of Borås, 501 90 Borås, Sweden, peter.rittgen@hb.se

ABSTRACT

The Language-Action Perspective (LAP) considers communication as the backbone of any organization. The structure of communication mirrors the structure of the organization and its processes, so by modeling the former we can understand the latter. When designing an information system (IS) to support the organization, the focus shifts from communication to information which is typically accompanied by a shift in the modeling paradigm. The Unified Modeling Language (UML) is a prime candidate for IS design. Here we study a framework for integrating both approaches, LAP and UML, and thereby organizational and IS modeling.

INTRODUCTION

Communication is at the heart of almost everything that happens in an organization. It is used not only to transmit information but also to act. With its help we negotiate agreements, make commitments, settle disputes, utter requests and so on. From a language-action perspective a business is understood as a network of such language actions which are complemented by physical actions that change the state of the material world. By putting communication in the centre helps us in understanding an organization because most business processes involve the use of language. In LAP this also covers all communication with the customer which implies a strengthening of customer orientation. The LAP community suggests a number of methods for modeling organizations. Examples of such approaches are Dynamic Essential Modelling of Organizations (DEMO) (Dietz, 1999) and ActionWorkflow (Medina-Mora, et al., 1992, Denning & Medina-Mora, 1995). They are based both on the speech-act theory (Austin, 1962; Searle, 1969; Habermas, 1984) and the conversation-for-action schema (Winograd & Flores, 1986).

The focus in LAP is on the communicative acts, i.e. on that part of communication where we use language to act. Less attention is paid to the part where we communicate to deliver or receive information. But in the design of information systems the latter is obviously of prime importance. So while it is still possible to use LAP for this task, it seems more appropriate to employ a modeling language that was specifically developed for this purpose. After a period during the early 90s where competing design methods had been at war with each other, a unified language emerged which, after years of standardization, has gained considerable ground in large parts of the IS community and is now reaching a state of consolidation. This makes UML a prime candidate for the IS design language.

If we use LAP for organizational modeling and UML for IS modeling, we need to integrate an LAP-based language with UML. This will give us (at least) three benefits:

1. The information that is present in the organizational model is preserved in the IS model. No information is lost.
2. The information system will meet the requirements of the organization (at least as put forth in the organizational model).
3. The design of the information system does not have to start at zero. The transformed diagrams from the organizational model can serve as a solid basis.

Therefore it is worthwile to investigate a conceptual framework for integrating an LAP-based language and UML.

THE LANGUAGE-ACTION PERSPECTIVE

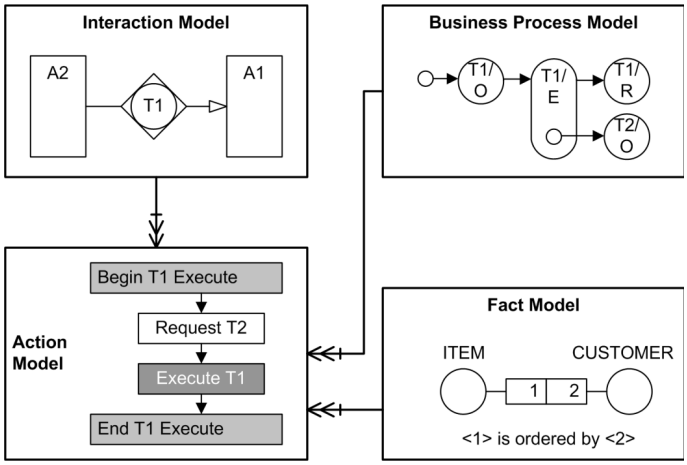
Among the approaches that consider organizations from the Language-Action Perspective we can find Action-Based Modeling (Lehtinen & Lyytinen, 1986), Action Workflow (Medina-Mora, et al., 1992), Business Action Theory & SIMM (Goldkuhl & Röstlinger, 1993, Goldkuhl, 1996) and the Speech Act-Based Approach (Johannesson, 1995). Two of the more elaborated methods, SIMM and DEMO, have been compared in (Reijswoud & Lind, 1998). For our purpose DEMO (Dynamic Essential Modeling of Organizations) represents the better choice because it offers a rich selection of diagram types that, on the one hand, are thoroughly rooted in the Language-Action Perspective but which, on the other hand, also exhibit at least an outward resemblance to certain UML diagrams. While this does not guarantee a successful integration, it justifies at least a more thorough investigation.

In LAP the structure of an organization is understood as a network of commitments. As these commitments are the result of communication, it follows that a model of the organization is essentially a model based on purposeful, communicative acts. In DEMO, all acts that serve the same purpose are collected in a *transaction* in which two roles are engaged: the *initiator* and the *executor*. Each transaction is assumed to follow a certain pattern which is divided into 3 sequential phases and 3 layers. The phases are: *order* (O), *execute* (E) and *result* (R). The layers are: success, discussion and discourse. On the success layer the phases are structured as follows. In the order phase the contract is negotiated. This involves typically a *request* being made by the initiator and a *promise* by the executor to carry out the request. In the next phase the contract is executed which involves factual changes in the object world (as opposed to the intersubject world of communication). Finally, in the result phase the executor *states* that the agreed result has been achieved and the initiator *accepts* this *fact*. If anything goes wrong on the success layer, the participants can decide to move to the discussion or discourse layer. For details on these layers see (Reijswoud, 1996).

Fig. 1 gives an overview of the architecture of DEMO. The Interaction Model shows actors and their relations to transactions but abstracts from time. The Business Process Model, on the other hand, abstracts from the actors but refines the transactional logic in two ways: it breaks each transaction into its phases and specifies how they are ordered causally and conditionally. This allows us to determine the order of the communicative acts in time. The Fact Model describes all information that is created or used by an organization. A fact is the result of a successful transaction and imples that the proposition of the request has become true. The Interstriction Model (not shown in fig. 1) is similar to the Interaction Model but in addition to the communication that is part of the transactions it also exhibits informative communication. All models are linked to the Action Model which gives a detailed account of activities carried out within a transaction phase (which can also involve links to other transactions).

Fig. 2 gives examples of an Interaction and a Business Process Model. They are taken from (Reijswoud, & Dietz, 1999) and show a part of the

Figure 1. Architecture of DEMO (simplified)



business process of an organization called SGC, a non-profit organization that mediates consumer complaints in the Netherlands.

The transactions of the example are as follows:

- T6: Handling_complaint
- T7: Defending_complaint
- T8: Giving_advice
- T9: Passing_judgement

The actor A2, who is responsible for mediating the claim, requests that actor A6 handles the complaint. Both A2 and A6 are internal actors (represented by white boxes). The latter will give the supplier a chance to defend the complaint, ask an expert to give advice and request that the committee passes a judgement. S2 – S4 are external actors as the grey boxes show. A simple line connects the initiator with the transaction, an arrow points from it to the executor. The grey line represents the system boundary. Observe that fig. 2 shows only a fragment of the model.

The right side of fig. 2 contains a part of the Business Process Model. It shows details of the execution phase of transaction 6: Handling_complaint. This phase is called T6/E. From inside it, the transactions T7, T8 and T9 are started. This is represented by arrows from the initiation points (small, white circles) to the order phases of the respective transactions. Solid arrows indicate a causal relation, dashed ones a conditional relation. The inside of a phase is viewed as a concurrent region, so all 3 triggered transactions could start at the same time if it were not for the dashed lines. An arrow that is crossed by a line represents an optional relation. In short: the supplier is asked to defend the complaint in any case and the expert is possibly consulted. After T7 (and possibly T8) have been completed, the committee is asked to pass judgement. Only when T9/R has been finished can we also terminate T6/E. Observe that this is due to the dashed arrow between them. As a rule

Figure 2. Examples of Interaction Model and Business Process Model

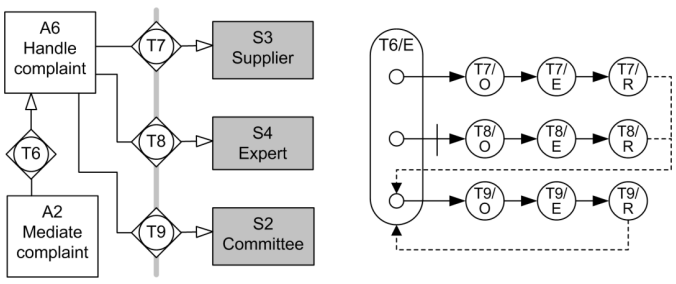
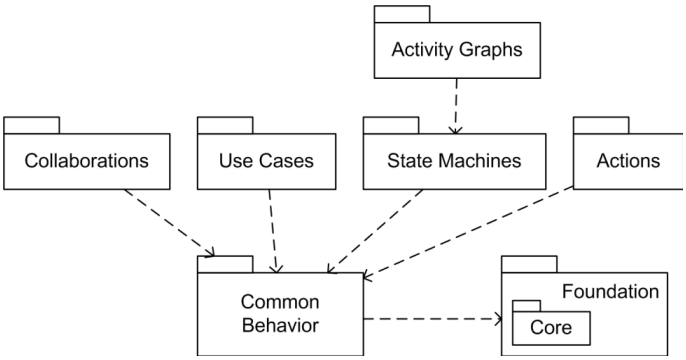


Figure 3. Architecture of UML (simplified)



the initiation points inside a phase trigger transactions in an asynchronous manner without waiting for their completion.

THE UNIFIED MODELING LANGUAGE

The specification of the Unified Modeling Language (OMG, 2003) is divided into two parts: semantics and notation. The first part introduces the concepts of UML with the help of metamodels and natural language. It is organized in packages (which are themselves a concept of UML). See fig. 2 for an overview of the relevant packages.

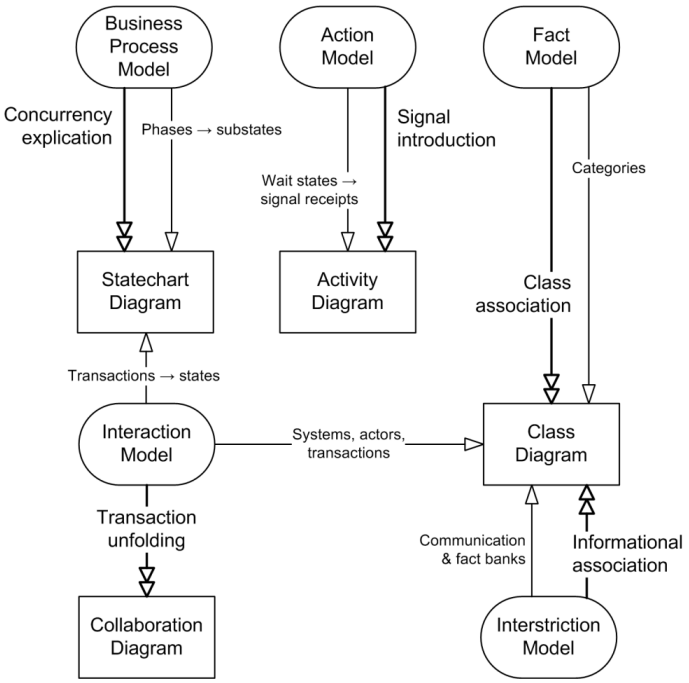
All concepts for structural models are defined in the Foundation. They comprise static aspects of a system such as classes, interfaces and attributes. Based on the Foundation the elements of the behavioral (dynamic) models are specified which consist of Common Behavior (such as signals, procedures, instances etc.) and diagram-specific behavior (one package for each, see fig. 3). Their purpose is defined in (OMG, 2003, pp. 2-92 f.) as: "The Collaborations package specifies a behavioral context for using model elements to accomplish a particular task. The Use Case package specifies behavior using actors and use cases. The State Machines package defines behavior using finite-state transition systems. The Activity Graphs package defines a special case of a state machine that is used to model processes. The Actions package defines behavior using a detailed model of computation."

The notation part of the language specification introduces a number of diagrams that define how the elements of the semantics packages can be represented graphically. For the purpose of this paper the relevant diagrams (and the primary packages they refer to) are: Collaboration Diagram (Collaborations), Statechart Diagram (State Machines), Activity Diagram (Activity Graphs) and Class Diagram (Foundation). For a detailed description of these diagrams refer to (OMG, 2003).

A FRAMEWORK FOR INTEGRATING DEMO AND UML

The integration of DEMO and UML proceeds in two steps: first, we map each concept of DEMO to a corresponding one in UML; and, second, we transform each diagram type of DEMO into one of UML. Both steps are not necessarily possible for any combination of two languages. The first requires that the semantics of the target language is a superset of that of the source language. In our case this condition holds because UML is sufficiently rich so that we can find a matching concept for each element of DEMO. The second step is more difficult because it requires that the first step was successful and that the concepts are grouped into diagram types in the same way. Even if two languages agree on the concepts, it is still quite likely that they will define different views. So it might easily happen that the information contained in a particular diagram type in one language is represented by two different diagram types in the other. In such a case the integration of the languages is much more involved but it may still be possible. In our case it is possible to transform the five diagram types of DEMO into four of UML where two of the former are merged into one of the latter (see fig. 4).

Figure 4. Framework for Integrating DEMO and UML



The DEMO diagrams are represented by rounded boxes, the UML diagrams by rectangular boxes. The mapping of concepts is visualized by single-headed arrows, the transformation of diagrams by double-headed arrows. Each diagram conversion involves a transformation of the notation but will also require some more sophisticated transformation process (e.g. transaction unfolding). The Interaction Model introduces systems, actors and transactions that all become classes in UML. But the transactions (the most important concept of DEMO) also form states in the Statechart Diagram. The Business Process Model refines transactions into phases which in turn become substates of the respective transaction state in the Statechart Diagram. The basic elements of the Fact Model are the categories. They correspond to classes in UML. The Interstriction Model introduces fact and communication banks to store records of facts and communication. They also correspond to classes in UML. The Action Model introduces wait states which map to signal receipts in Activity Diagrams.

The Interaction Model is transformed into the Collaboration Diagram. Apart from a notational conversion this requires an unfolding of the transactions, a concept which has no immediate dynamic counterpart in UML. Each transaction is split into its communicative acts which then are represented by messages in UML. An example of that is given in the next section.

The Business Process Model is transformed into the Statechart Diagram. Again this involves a change in notation but also an explication of the inherent concurrent behavior of a phase. A phase can have many concurrent initiation points but each state has only one initial (sub)state. Dividing the state into concurrent regions is not feasible due to the asynchronous nature of the threads triggered by the initiation points. Hence the initial state is forked into as many threads as there are initiation points that have no arrows pointing at them (plus one that leads to the final state if no arrow points to the phase). An arrow pointing at a phase maps to one pointing at the corresponding final state. If more than one arrow points at a phase or initiation point the respective arrows in the Statechart Diagram are joined by a synchronization bar. Optional relationships map to guarded transitions. An example for such a transformation is given in the next section.

The Action Model is transformed into the Activity Diagram. Apart from the usual notational conversion this means that a signal receipt has to be introduced into the Activity Diagram for each wait state that is found in the Action Model. Likewise, a signal sending is introduced after the actitivity that corresponds to the action that is waited for.

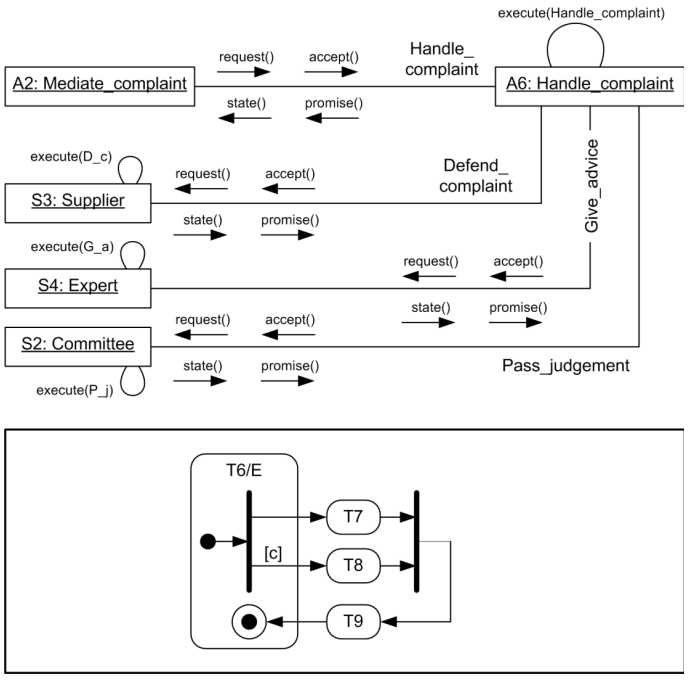
The Fact Model is transformed into the Class Diagram. This involves that each fact (which is an *n*-ary relations between categories) is mapped to an association class that has associatations to each of the classes corresponding to the categories. That process is called class association.

The Interstriction Model introduces further associations into the Class Diagram, one for each informational link between an actor and a transaction, fact bank or communication bank. We call that process informational association.

EXAMPLES OF THE MODEL TRANSFORMATION

Due to the limited space we give examples for the first two transformations only. Fig. 5 shows the Collaboration Diagram (upper half) for the Interaction Model of fig. 2 (left) and also the Statechart Diagram (lower half) for the Business Process Model of fig. 2 (right). Each system or actor of the Interaction Model becomes an object (instance) in the Collaboration Diagram. A transaction is represented by a (communication) link that bears the name of the transaction (i.e. its purpose). This link is bidirectional (i.e. it does not have an arrowhead that restricts the navigability) because a transaction involves communication in both directions, from initiator to executor and back. This link can now be used to exchange the messages that correspond to the communicative acts in DEMO. Each executor has also a link to itself which means that the execution phase is self-induced. A request and an accept message are introduced along the link with arrows that point from the initiator to the executor. They represent the first and the last communicative acts of a transaction, respectively. In the same way, a promise and a state message are attached to the link. They are passed from the executor to the initiator and form the second and penultimate speech acts, respectively. Observe that a Collaboration Diagram does not require us to specify the order of messages but we could do so with the help of sequence numbers in front of the message names.

Figure 5. Collaboration Diagram and Statechart Diagram



The lower half of fig. 5 shows the Statechart Diagram that corresponds to the excerpt from the Business Process Model of fig. 2. The execution phase of T6 becomes a state (which itself is a substate of the transaction state T6). Within T6/E the initial state is forked into two concurrent threads to trigger transactions T7: Defending_complaint and T8: Giving_advice. While T7 is triggered in any case, the transition to T8 is guarded by [c], which means that the expert is asked to give advice under a condition that has not yet been specified; the Business Process Model only indicates that T8 is optional, not under which circumstances it is carried out. On completion of T7 (and possibly T8), T9: Passing_judgement is carried out. After that we enter the terminal state of T6/E which concludes the execution phase of T6.

CONCLUSION

We have presented a framework that allows for the integration of two modeling languages, DEMO and UML, by mapping the respective concepts and eventually transforming diagrams of the former into corresponding ones of the latter. The next step is to gather experience with the application of this framework in the context of a suitable reengineering project. From a theoretical point of view the two languages represent completely different paradigms that are hard to reconciliate: DEMO is an approach that is deeply rooted in linguistics and the study of human communication, while UML has many of its roots in computer science and the study of software artefacts (though by far not all). It is therefore surprising that a tight integration of them can be undertaken at all. It should be noted, though, that we have chosen from the LAP approaches the one that best facilitates the integration with UML. Nevertheless, we hope that our work can contribute to closing the gap between organizational modeling and modeling of information systems.

REFERENCES

Austin, J. L. (1962). How to Do Things with Words. Oxford University Press.

Denning, P.J., & Medina-Mora, R. (1995). Completing the Loops. Interfaces, 25 (3), 42-57.

Dietz, J.L.G. (1999). Understanding and modeling business processes with DEMO. In Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I. and Métais, E. (eds.): Conceptual modeling - ER '99: proceedings (Lecture notes in computer science 1728). Berlin: Springer, 188-202.

Goldkuhl, G. (1996). Generic business frameworks and action modelling. In Dignum, F., Dietz, J., Verharen, E. and Weigand, H. (eds.): Communication Modeling – The Language/Action Perspective, Proceedings of the First International Workshop on Communication Modeling, Electronic Workshops in Computing, Berlin: Springer.

Goldkuhl, G., Röstlinger, A. (1993). Joint elicitation of problems: An important aspect of change analysis. In Avison, D., Kendall, J. and DeGross, J. (eds.): Human, Organizational, and Social Dimensions of Information Systems Development, Amsterdam: North-Holland.

Habermas, J. (1984). The Theory of Communicative Action 1, Reason and the Rationalization of Society. Boston: Beacon Press.

Johannesson, P. (1995). Representation and Communication: A Speech Act Based Approach to Information Systems Design. Information Systems, 20 (4), 291-303.

Lehtinen, E., & Lyytinen, K. (1986). An Action Based Model of Information Systems. Information Systems, 11 (4), 299-317.

Medina-Mora, R., Winograd, T., Flores, R., & Flores, F. (1992). The Action Workflow Approach to Workflow Management Technology. In Turner, J. and Kraut, R. (eds.): Proceedings of the Conference on Computer-Supported Cooperative Work, CSCW'92. New York: ACM Press.

OMG (Ed.) (2003). Unified Modeling Language Specification: Version 1.5, March 2003. Needham: OMG.

Reijswoud V E van (1996) The Structure of Business Communication: Theory, Model and Application. PhD Thesis. Delft, The Netherlands: Delft University of Technology.

Reijswoud, V.E. van, & Dietz, J.L.G. (1999). DEMO Modelling Handbook, Volume 1. TU Delft. Online version available at http://www.demo.nl/documents/handbook.pdf.

Reijswoud, V.E. van, & Lind, M. (1998). Comparing Two Business Modelling Approaches in the Language Action Perspective. In Goldkuhl, G., Lind, M. and Seigerroth, U. (eds): Proceedings of the Third International Workshop – The Language Action Perspective on Communication Modelling. Jönköping, Sweden: International Business School. Online available at http://www.ihh.hj.se/eng/vits/lap98/lap98_07.pdf.

Searle, J. R. (1969). Speech Acts, An Essay in the Philosophy of Language. London: Cambridge University Press.

Winograd, T., & Flores, F. (1986). Understanding Computers and Cognition: A New Foundation for Design. Norwood, NJ: Ablex.