TABLE OF CONTENTS

MODELING OF BUSINESS SYSTEMS USING SOM

Otto K. Ferstl, Elmar J. Sinz[1]

SOM is an object-oriented methodology for comprehensive and integrated modeling of business systems. It is based on a framework consisting of the layers business plan, business process model, and business application system as well as views on these layers focusing on specific aspects. This chapter presents the SOM language for business process modeling and shows how business application systems can be linked to business process models. The language uses concepts of systems theory and is based on the notions of business object, business transaction, task, event, and service. Business objects are coordinated by feedback control or by negotiation. Decomposition rules allow a stepwise refinement of a business process model. The chapter includes a detailed example to illustrate the methodology.

1 Introduction

SOM is a methodology for modeling business systems [FeSi90, FeSi91, FeSi95]. The abbreviation means **Semantic Object Model**, expressing that the SOM methodology is fully object-oriented and designed to capture business semantics explicitly. General basis of the SOM methodology are concepts of systems theory.

SOM supports the core phases of business engineering, such as analysis, design, and redesign of a business system. A business system is an open, goal-oriented, socio-technical system. Thus the analysis of a business system focuses on the interaction with its environment, goal pursuing business processes, and resources. Moreover, the dynamic behavior of a business system requires investigation of properties such as stability, flexibility, and complexity [Bahr92].

The backbone of the SOM methodology is an enterprise architecture which uses different perspectives on a business system via a set of models. These models are grouped into three model layers referring to a business plan, business process models and resource models. Each layer describes the business system as a whole, but with respect to the specific perspective on the model. In order to reduce complexity, each model layer is subdivided into several views, each focusing on specific aspects of a model layer. On the meta level, the modeling language of each layer is defined by a meta model and derivated view definitions. Thus the enterprise architecture provides a modeling framework which helps to define specific semantics and to manage complexity of the model [Sinz97]. In this chapter we outline the methodological framework of SOM as well as its modeling language.

[1]{Univ.-Prof. Dr. Otto K. Ferstl, Univ.-Prof. Dr. Elmar J. Sinz}, Information Systems, University of Bamberg, D-96045 Bamberg, Germany. Phone ++49 951 863 {2679, 2512}, Fax ++49 951 863 {2680, 2513}, e-mail {otto.ferstl, elmar.sinz}@sowi.uni-bamberg.de.

2 Characteristics of Business Systems

In terms of systems theory a business system is an open, goal-oriented, socio-technical system [FeSi98]. It is open because it interacts with customers, suppliers, and other business partners transferring goods and services. The business system and its goods/services are part of a value chain which in general comprises several consecutive business systems. A corresponding flow of finance runs opposite the flow of goods and services.

The behavior of a business system is aimed at business goals and objectives. Goals specify the goods and services to be provided by the system. Objectives (e.g. profit and turnover) are defined measurable levels against which business performance can be measured.

Actors of a business system are humans and machines. Human actors are persons in different roles. Machine actors in general are plants, production machines, vehicles, computer systems etc. SOM pays specific attention to application systems which are the machine actors of the information processing subsystem of a business system (information system). An application system consists of computer and communication systems running application software. The degree of automation of an information system is the ratio of tasks carried out by application systems to all tasks of the information system.

The notion of a business system as open and goal-oriented reflects a perspective from outside the system. An inside perspective shows a distributed system of autonomous, loosely coupled components which cooperate in pursuing the system's goals and objectives. The autonomous components are business processes [FeSi93, FeSi95] which produce goods and services and deliver them to other business processes.

The cooperation of business processes is coordinated primarily through process specific objectives which are derived from the overall objectives of a business system. This is done by the business system's management. Within the degrees of freedom defined by the process specific objectives a secondary coordination is done by negotiation between the business processes.

Inside a business process there are components which also cooperate and have to be coordinated. This coordination is done by an intra-process management which controls the activities of the process components by sending instructions to them and supervising their behavior. In contrast to the coordination between business processes, the components inside a business process are guided closely by the process management.

The components of a business process as well as the business processes as a whole take care of functions which are essential to every business system. The following classification of these functions helps to identify business processes and their components: (1) input-output-function to implement the characteristic of openness, e.g. a production system, (2) supply function to provide material resources and energy, (3) maintenance function to keep the system running, (4) sensory function to register disturbances or defects inside or outside the system, (5) managing function to coordinate the subsystems [Beer81].

3 Architecture of Business Systems

The SOM methodology utilizes an **enterprise architecture** which consists of three layers (fig. 1) [FeSi95]:

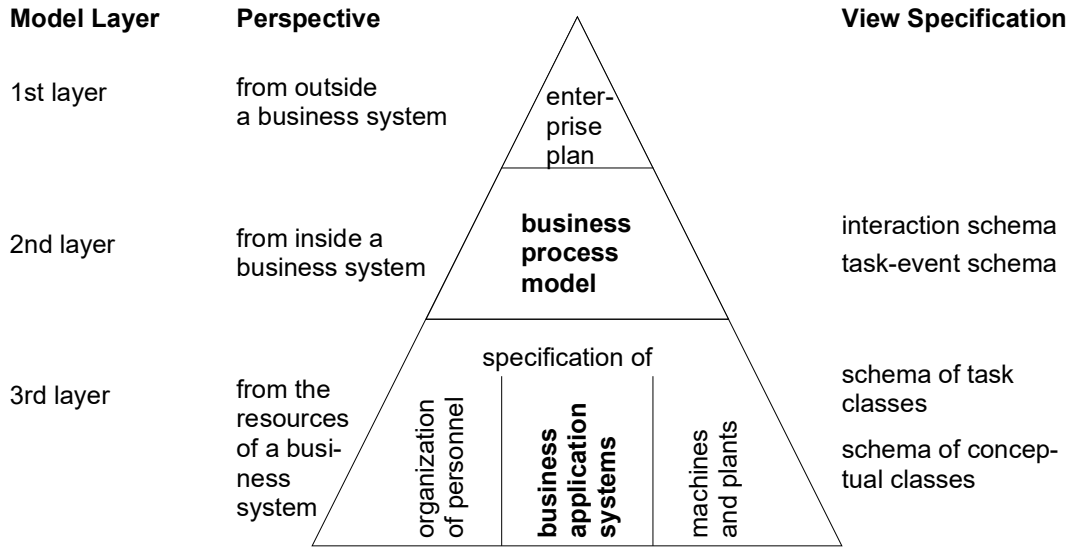

Fig. 1: Enterprise architecture [FeSi95]

- **Enterprise plan**: The enterprise plan constitutes a perspective from outside a business system. It focuses on the global task and the resources of the business system. The specification of the global task includes the universe of discourse, the goals and objectives to be pursued, as well as the goods and services to be delivered. Requirements on resources are derived from the global task and have to be cross-checked to the capabilities of available resources. So both global task and resources determine themselves mutually.

  A first evaluation of an enterprise plan is done by an analysis of chances and risks from a perspective outside the business system, and an additional analysis of the strengths and weaknesses of the business system from an inside perspective. Strategies on products and markets, strategic actions, constraints, and rules serve as guidelines to realize an enterprise plan.

- **Business process model**: The business process model constitutes a perspective from inside a business system. It specifies main processes and service processes. Main processes contribute directly to the goals of the business system, service processes provide their outcome to main processes or other service processes. The relationships between business processes follow the client/server concept. A client process engages other processes for delivering the required service. Business processes establish a distributed system of autonomous components. They cooperate in pursuing joint objectives which are derived from the overall objectives of a business system.

- **Specification of resources**: In general, personnel, application systems as well as machines and plants are resources for carrying out the tasks of business processes. In the following

we focus on information processing tasks and therefore omit machines and plants. Tasks of the information system are assigned to persons or to application systems classifying a task as non-automated or fully-automated. A task partly-automated has to be split into sub-tasks which are non-automated or fully-automated. The assignment of persons or application systems is aimed at optimal synergy of person-computer cooperation.

The different layers of the enterprise architecture help to build business systems in a flexible and manageable way. They cover specific aspects of an overall model which are outside perspective (enterprise plan), inside perspective (business process model), and resources. The relationships between the layers are specified explicitly. Each layer establishes a distributed system of autonomous, loosely coupled components. In contrast to a single-layered monolithic model, the multi-layered system of three models allows local changes without affecting the overall architecture. For example, it is possible to improve a business process model (inside perspective) yet retaining goals and objectives (outside perspective), or to replace actors of one type by other ones.

Following an outside-in approach it is advisable to build the three model layers top down the enterprise architecture. But the architecture does not force this direction. There may be good reasons to depart from this guideline e.g. when analyzing existing business systems. Here it is sometimes difficult to find an elaborated enterprise plan, so modeling starts at the business process layer focusing on the inside perspective. The enterprise plan may be completed when the other layers are fully understood. In each case effects on other layers have to be balanced and approved.

The enterprise architecture implies that functionality and architecture of the business application systems are derived from the business process model. The relationships between both layers are formalized to a high degree. Design decisions and results at the business process layer are translated automatically into the layer of application systems. The architecture of the layer of application systems uses the concept of object-integration to combine conceptual and task classes [Fers92]. Alternatively it is possible to link a business process model to an existing, traditional application system which follows the traditional concepts of function integration or data integration. In this case tasks to be automated are linked to functional units of the application system.

4 Language for Business Process Modeling

In this section we define the language for business process models. The language is specified by a meta model (section 4.1) and a set of decomposition rules (section 4.2). The section is completed by an example, introducing the business process *distribution* of a trading company (section 4.3).

4.1 Meta Model for Business Process Models

The meta model for business process modeling shows notions and relationships between notions (fig. 2). It is specified as a binary entity-relationship schema. Relationships between notions are associated with a role name as well as two cardinalities to denote how many instances of the one notion can be connected to one instance of the other notion at least and at most. Within the meta model the notions are represented by entities. Each entity also contains the symbols used for representation within a business process model.

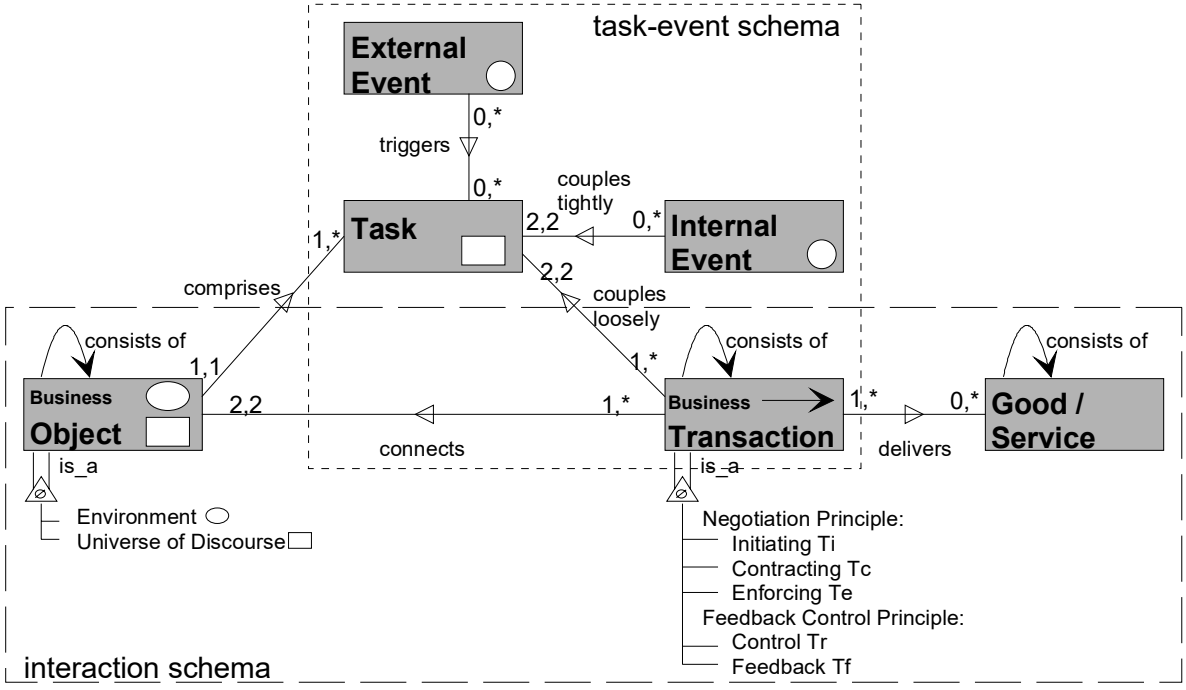

Fig. 2: Meta Model for Business Process Models [FeSi95]

As introduced in section 3, a business process model specifies a set of business processes with client/server relationships among each other. A business process pursues its own goals and objectives which are prescribed and tuned by the management of a business system. Cooperation between processes is a matter of negotiation. The term **business process** denotes a compound building block within a business process model and therefore it is not a basic notion of the language. A business process consists of at least one business object and one or more business transactions.

At the initial level of a business process model, a **business object** (object in short) produces goods and services and delivers them to customer business processes. Each business object belongs exclusively to a business process of the universe of discourse or to the environment of a business system. A **business transaction** (transaction in short) transmits a **good or service** to a customer business process or receives a good or service from a supplier business process. A transaction connecting different business processes belongs to both processes.

A business process may be refined using the decomposition rules given below. At a more detailed level of a business process model, each business object appears in one of two different roles: an operational object contributes directly to producing and delivering of a good/service while a management object contributes to managing one or more operational objects using messages. A business transaction transmits a good/service or a message between two operational objects or a message between two management objects or between a management object and an operational object.

A business transaction connects two business objects. Conversely, a business object is connected with one to many („*“) in-going or out-going business transactions. From a structural viewpoint a transaction denotes an interaction channel forwarding goods, services, or messages. From a behavioral viewpoint a transaction means an event which is associated with the transmission of a specific good, a service package, or a message.

A business object comprises one to many **tasks**, each of them driving one to many transactions. A transaction is driven by exactly two tasks belonging to different business objects. The tasks of an object share common states and are encapsulated by the object. These tasks pursue joint goals and objectives which are attributes of the tasks.

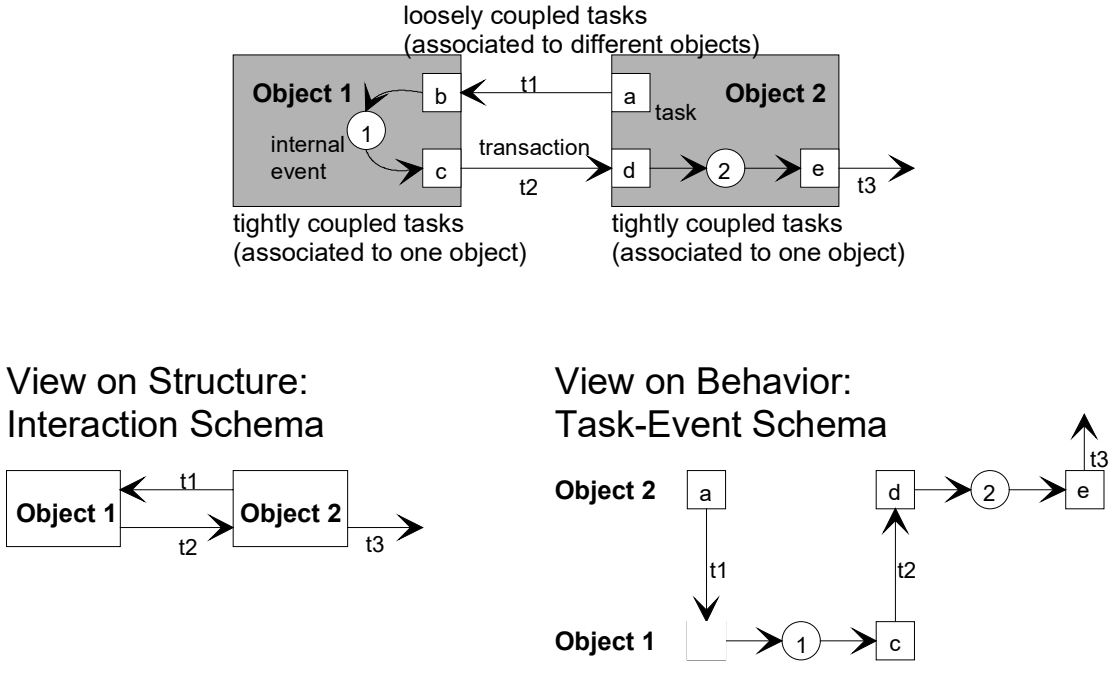

Fig. 3: Representation of structure and behavior in a business process model

The SOM methodology uses two different concepts of coupling tasks (fig. 3, top): Loosely coupled tasks belong to different objects and therefore operate on different states. The tasks are connected by a transaction which serves as an interaction channel for passing states from one task to the other. A task triggers the execution of another task by an event (good, service package, or message) riding on the interaction channel. Tightly coupled tasks belong to the same object and operate on the same states. The tasks are connected by an **internal event** which is sent from one task to trigger the execution of the other. The concept of encapsulating tightly coupled tasks by an object and loosely coupling the tasks of different objects via transactions is a key feature of the object-oriented characteristic of the SOM methodology.

A third type of event is the external event. An **external event** denotes the occurrence of an event like „the first day of a month“ which is not bound to a transaction.

Due to its complexity, a business process model is represented in two different diagrams (fig 3 bottom and fig 2): The **interaction schema** is the view on structure. It shows business objects which are connected by business transactions. The **task-event schema** is the view on behavior. It shows tasks which are connected by events (transactions, internal events, or external events).

4.2 Decomposition Rules

The SOM methodology allows a business process model to be decomposed by stepwise refinement. Decomposition takes place with the components of the interaction schema specifying the structure of a business process model, i.e. business objects, business transactions, and goods/services (see the relationship *consists of* in fig. 2). The components of the task-event schema which specify the behavior of a business process model (tasks, events riding on transactions, internal events, and external events) are not decomposed but redefined on subsequent decomposition levels of a business process model. The decomposition rules for business objects and business transactions are shown in fig. 4. Specific rules for decomposition of goods/services are not required because of simply decomposing them into sub-goods/subservices.

Decomposition rules for business objects:

O	::= { O', O'', T_r(O', O''), [T_f(O'', O')] }	(1)
O	::= { O', O'', [T(O', O'')] }	(2)
O	::= { spec O' }$^+$	(3)
O' | O''	::= O	(4)

Decomposition rules for business transactions:

T(O, O')	::= [[T_i(O, O') seq] T_c(O', O)] seq T_e(O, O')	(5)
T_x	::= T'$_x$ { seq T''$_x$ }$^+$ | T'$_x$ { par T''$_x$ }$^+$ (x = i, c, e, r, f)	(6)
T_x	::= { spec T'$_x$ }$^+$ (x = i, c, e, r, f)	(7)
T_i | T_c | T_e	::= T	(8)
T_r | T_f	::= T	(9)

Fig. 4: Decomposition rules for business objects and business transactions (::= replacement, {} set, {}$^+$ list of repeated elements, [] option, | alternative, seq sequential order, par parallel order, spec specialization)

The decomposition of a business process model helps to manage its complexity, allows to separate the management system of a business process from its operational system, and uncovers the coordination of a business process.

The SOM methodology uses two basic coordination principles within decomposition [FeSi95]:

- Applying the **feedback control principle** (rule 1) a business object is decomposed into two sub-objects and two transactions: a management object O' and an operational object O'' as well as a control transaction T_r from O' to O'' and a feedback transaction T_f in opposite direction. These components establish a feedback control loop. The management object prescribes objectives or sends control messages to the operational object via the control transaction. Conversely the operational object reports to the management object via the feedback transaction.

- Applying the **negotiation principle** (rule 5) a transaction is decomposed into three successive transactions: (1) an initiating transaction T_i, where a server object and its client learn to know each other and exchange information on deliverable goods/services, (2) a contracting transaction T_c, where both objects agree to a contract on the delivery of goods/services, and (3) an enforcing transaction T_e, where the objects transfer the goods/services.

The types of transactions resulting from the decomposition are shown in the meta model (fig. 2) as specialized transactions.

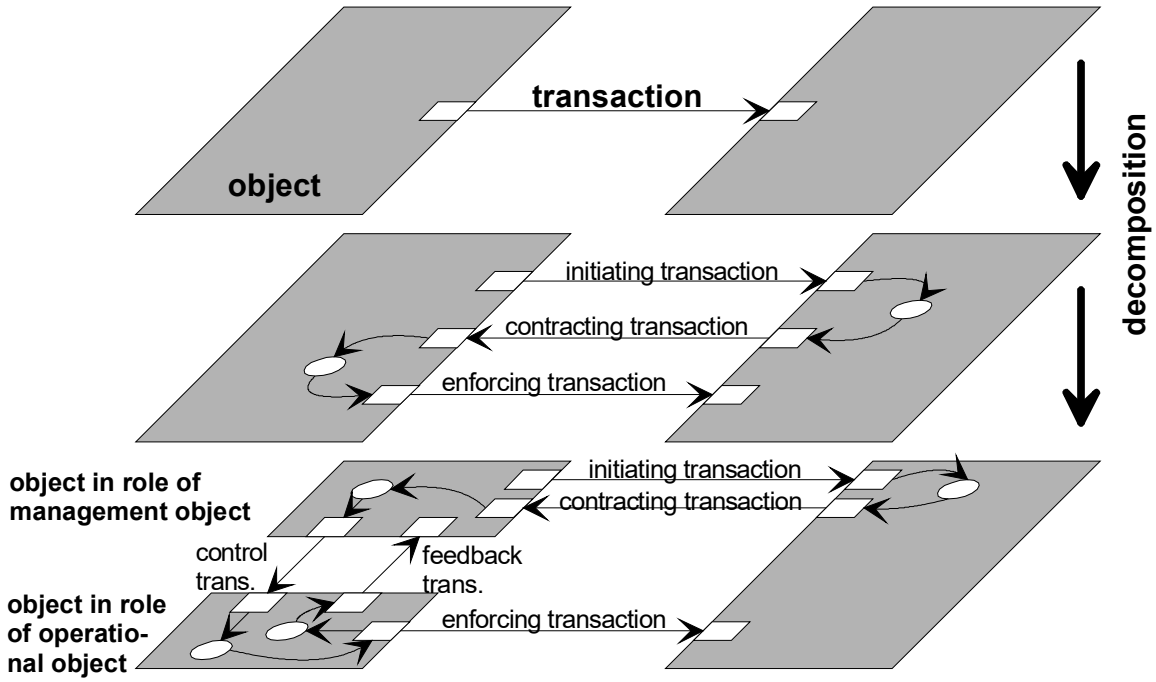

Fig. 5: Decomposition of business process models

Fig. 5 illustrates the application of the coordination principles for the decomposition of business process models. The decomposition of the first level into the second level is done by applying the negotiation principle. Applying the feedback control principle leads to the third level.

In addition to the coordination principles given above, a transaction may be decomposed into sub-transactions of the same type which are executed in sequence or in parallel (rule 6). Correspondingly, a business object may be decomposed into sub-objects of the same type (management object or operational object) which may be connected by transactions (rule 2). Objects as well as transactions may be specialized within the same type (rules 3 and 7). The other rules (4, 8, and 9) are used for replacement within successive decompositions.

It is important to state that successive decomposition levels of a business process model do not establish new, different models. They belong to exactly one model and are subject to the consistency rules defined in the meta model.

4.3 Example: Business Process *Distribution*

To give an example, figure 6 (left) introduces the business process *distribution* of a trading company. At the initial level, the interaction schema consists of three components, (1) the business object *distributor* which provides a service, (2) the transaction *service* which delivers the service to the customer, and (3) the business object *customer* itself. *Distributor* is an internal object belonging to the universe of discourse while *customer* is an external object belonging to the environment. At this level the entire cooperation and coordination between the two business objects is specified by the transaction *service*. Figure 6 (right) shows the

corresponding sequence of tasks which is very simple. The task names in the task-event schema are derived from the name of the transaction. Here, the task *service>* (say „send service“) of *distributor* produces and delivers the service, the task *>service* (say „receive service“) of *customer* receives it. The arrow *service* here defines the sequence of the two tasks belonging to the transaction *service* which is represented in the interaction schema by an arrow, too.

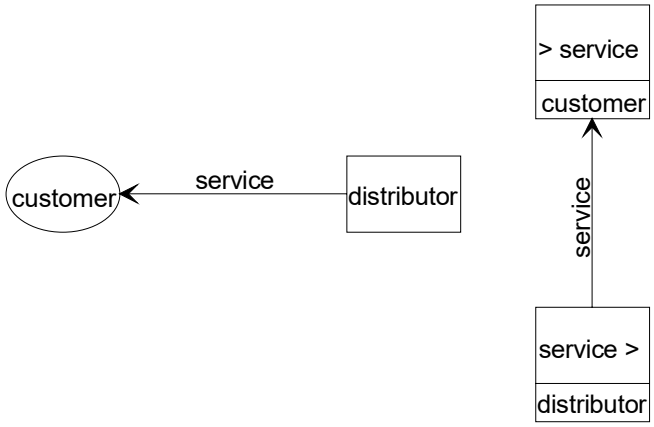

Fig. 6: Interaction schema (left) and task-event schema (right) of business process *distribution* (1st level)

Transactions like *service* connect business objects inside the universe of discourse and link business objects to the environment. When modeling a value chain the business process model of a trading company includes a second business process *procurement*, which receives services from a business object *supplier*, belonging to the environment, and delivers services to *distributor*.

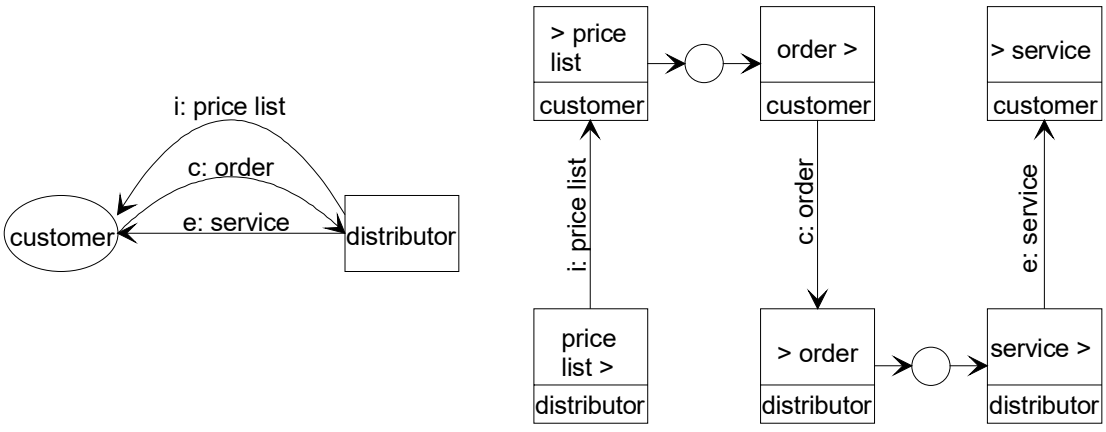

Fig. 7: Interaction schema (left) and task-event schema (right) of business process *distribution* (2nd level)

The example (fig. 6) will be continued now. As *customer* and *distributor* negotiate about the delivery of a service, the *service* transaction is decomposed according to the negotiation principle into the sub-transactions *i: price list* (initiating), *c: order* (contracting), and *e: service* (enforcing transaction). The corresponding task-event schema is determined implicitly because the sub-transactions are executed in sequence (fig. 7). The tasks of each business object are connected by object-internal events.

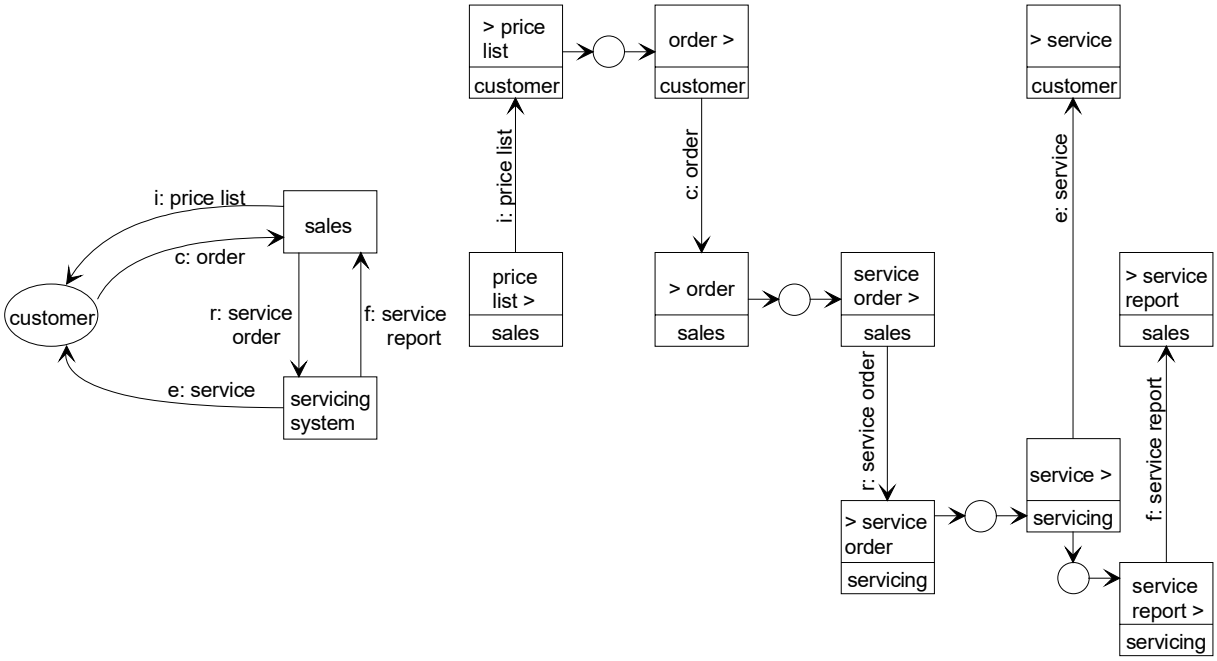

Fig. 8: Interaction schema (left) and task-event schema (right) of business process *distribution* (3rd level)

In the next step, the feedback control principle is applied to *distributor* to uncover the internal management of the business object. This leads to the sub-objects *sales* (management object) and *servicing system* (operational object) as well as the transactions *r: service order* (controlling transaction) and *f: service report* (feedback transaction). At the same time the transactions assigned to the parent object *distributor* are re-assigned to the new sub-objects. The *sales* sub-object deals with *price list* and *order*, the *servicing system* operates the *service transaction* (fig. 8).

Continuing the example, the final decomposition of the business process *distribution* uses the additional rules given above (fig. 9 and 10). Here, the *servicing system* and the *service* transaction are decomposed to find business objects and transactions which operate homogeneous goods or services. First, the *e: service* transaction is decomposed into the sequence *e: delivery* and *e: cash up*. The *cash up* transaction is decomposed again according to the negotiation principle into the sequence *c: invoice* and *e: payment*. The initiating transaction is omitted because the business objects already know each other. The contract of the *invoice* transaction refers to amount and date of payment, not to the obligation to pay in principle which is part of the transaction *c: order*.

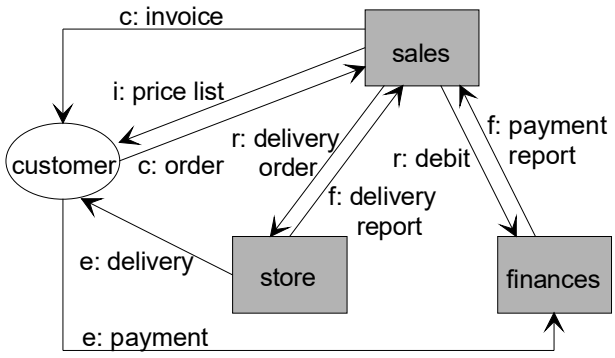

Fig. 9: Interaction schema of business process *distribution* (4th level)

As a result of this refinement, some other decomposition are necessary. The business object *servicing system* is decomposed into *store* and *finances*, responsible for goods and payments respectively. The transaction *r: service order* is decomposed into the parallel transactions *r: delivery order* and *r: debit.* And likewise the transaction *f: service report* is decomposed into *f: delivery report* and *f: payment report*.

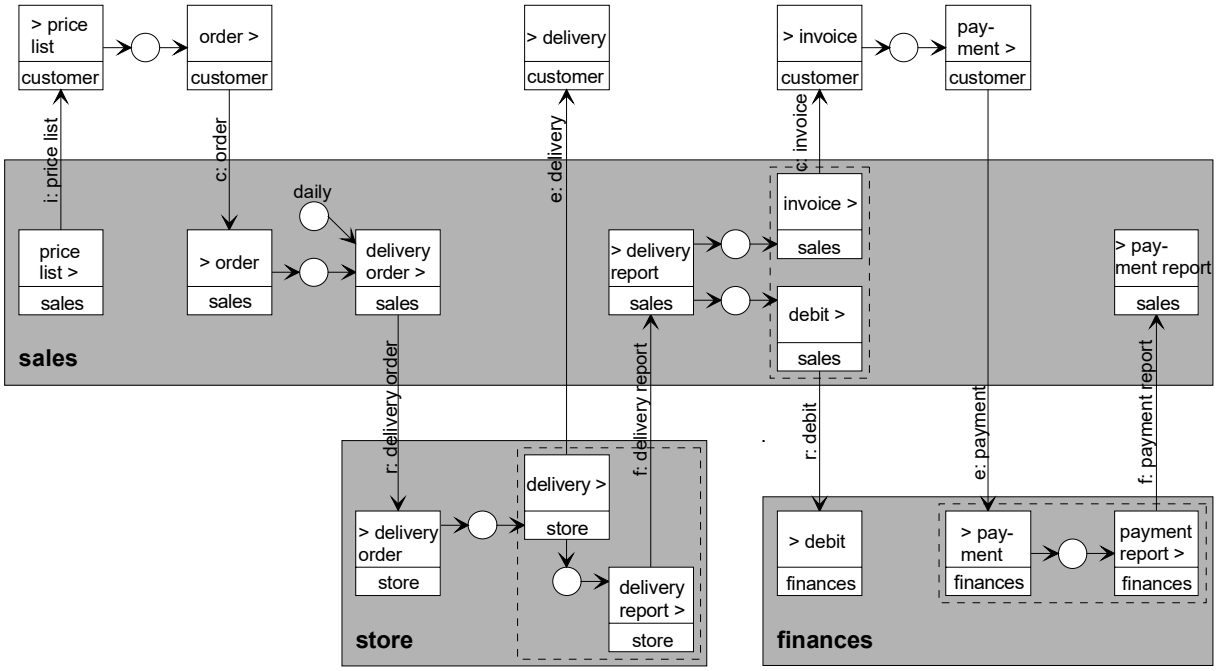

Fig. 10: Task-event schema of business process *distribution* (4th level)

5 Linking Business Application Systems to Business Process Models

As outlined in section 3, personnel and business application systems are resources to carry out business processes. In addition to the language for business process modeling, the SOM methodology provides a concept for explicitly linking business application systems to business process models. To introduce this concept, we investigate the automation of business processes using business application systems, define a meta model for the domain-specific specification of business application systems and discuss the impact of this concept on the architecture of business application systems.

5.1 Automation of Business Processes

The automation of a business process is determined by the automation of tasks and transactions. An information processing task is **fully-automated**, if it is carried out completely by an application system, it is **non-automated** if it is carried out by a person, and it is **partly-automated** if it is carried out by both a person and an application system cooperating [FeSi98].

Similar considerations hold for the **automation of transactions** within information systems. A transaction is **automated** if it is performed by an electronic communication system and it is **non-automated** if it is performed e.g. paper-based or orally.

Prior to defining the degree of automation, a task or a transaction have to be investigated if they are suitable for automation. A task is suitable for automation if its states and operations can be handled by a computer system. A transaction is suitable for automation if message passing and protocol handling can be done by an electronic communication system.

The relationship between business process model and business application systems is based exactly on the concept of automation of tasks and transactions. The interaction schema of a business process model is convenient to record the extent of both the **achievable** and the **achieved degree of automation**. Figure 11 (left) shows degrees of automation of tasks and transactions (see also [Kru97]) and applies them to the business object *sales* of the business process *distribution* (fig. 11 right).

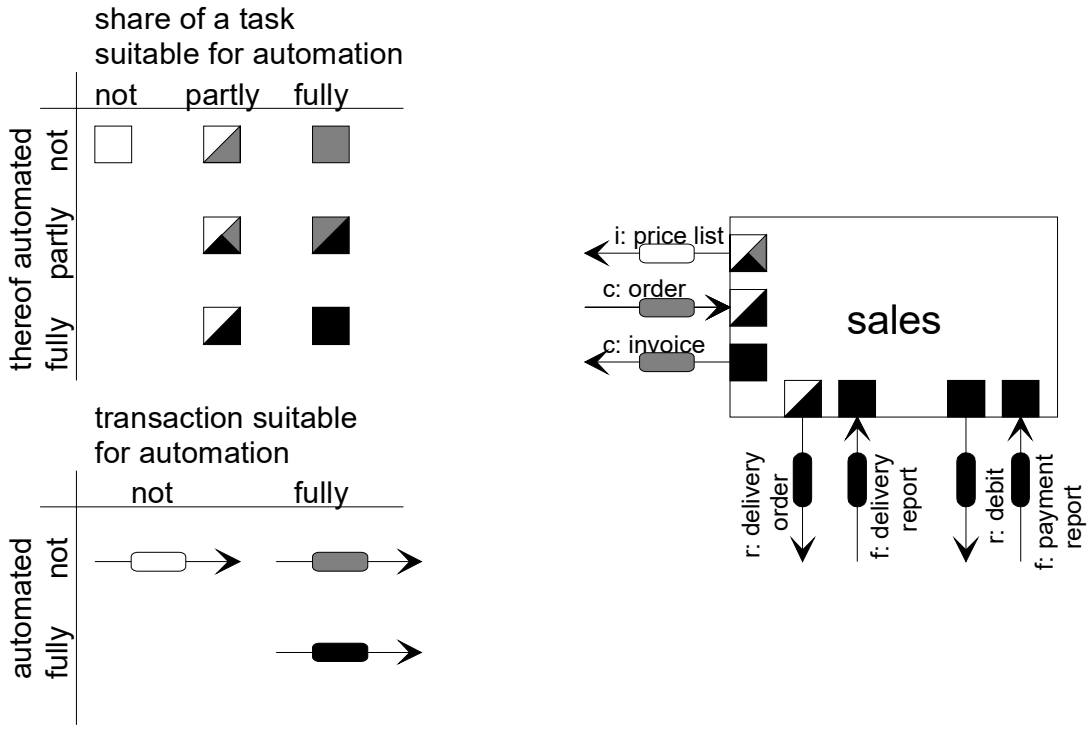

Fig. 11: Automation of tasks and transactions of the sales business object

5.2 Meta Model for Specifications of Business Application Systems

The SOM methodology uses an object-oriented approach for the domain-specific specification of business application systems. The corresponding meta model is shown in figure 12. The notion of *class* follows the general understanding of object-orientation. Classes have *attributes* and *operators* and they are connected by binary *relationships*. Relationships are either *is_a*, *interacts_with*, or *is_part_of* relationships. *interacts_with* relationships denote channels for message passing between two classes, *is_a* relationships are used to model the specialization of a class using inheritance, and *is_part_of* relationships allow the specification of the component classes of a complex class.

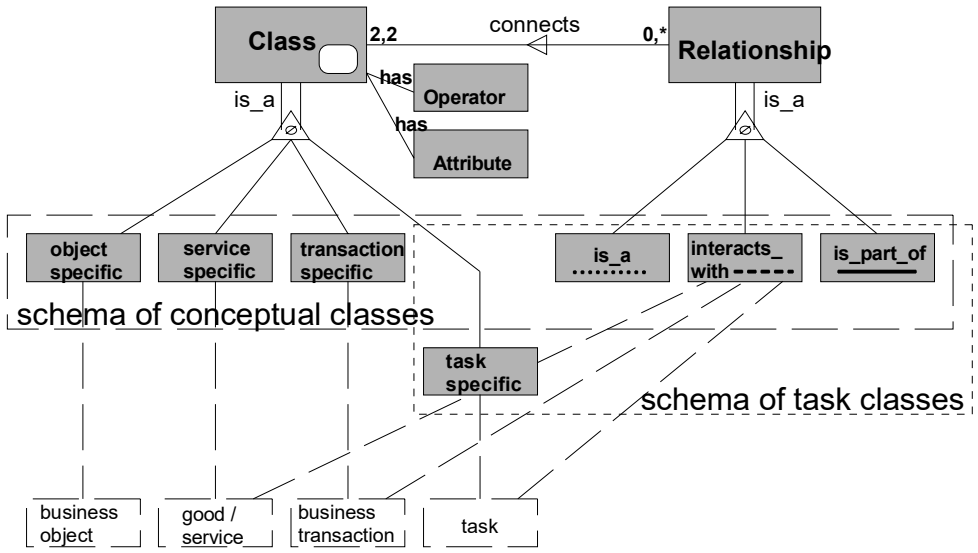

Fig. 12: Meta Model of Business Application Systems

To specify the linkage of business application systems to business process models the meta model in figure 12 is related to the meta model in figure 2. The relationships represented as dashed lines connect notions of a business process model to notions of a specification of an application system. A business object is connected to an *object-specific* class. A *good/service*, *business transaction*, or *task* is connected to a *service-specific*, *transaction-specific*, or *task-specific* class respectively as well as some *interacts_with* relationships. O*bject-specific*, *service-specific*, and *transaction-specific* classes together with their relationships are arranged to the **schema of conceptual classes**. *Task-specific* (*task class* in short) together with their relationships belong to the **schema of task classes**. *is_a* relationships and *is_part_of* relationships cannot be linked directly to a business process model. They have to be included during the further specification of the schema of conceptual classes or the schema of task classes.

5.3 Architecture of Business Application Systems

The way of linking a business application system to a business process model following the SOM methodology has impact on the architecture of business application systems. Again we concentrate on domain-specific aspects and omit details of design and implementation.

The SOM methodology leads to (1) strictly object-oriented, (2) distributed, (3) object-integrated, and (4) evolutionary adaptable specifications of business application systems [FeSi96]:

1. The domain-specific specifications of the schema of conceptual classes and of the schema of task classes are strictly object-oriented. Conceptual classes encapsulate (a) the states of the (automated) tasks of a business object as well as the states of the corresponding transactions and goods/services, and (b) the operations defined directly and exclusively on these states. Using the linkage of business process models and specifications of business application systems in figure 12, the initial structure of the schema of conceptual classes

can be derived from the most detailed level of the interaction schema in conjunction with the task-event schema of the corresponding business process model.

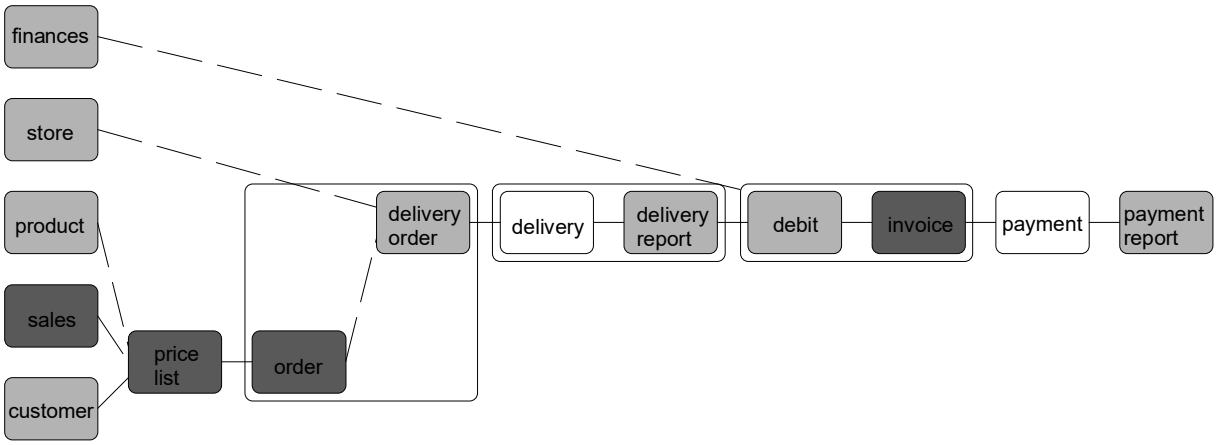

Fig. 13: Initial schema of conceptual classes of the business application system sales

Figure 13 shows the initial schema of conceptual classes derived from the business process model in figures 9 and 10. The classes at the left side correspond to the business objects and the *product*. The class *price list* is derived from the corresponding transaction, connecting *sales* and *customer* with reference to *product*. The same way the other classes are derived from transactions. Figure 13 refers to the complete *distribution* process. The shaded classes belong to the *sales* application system. Dark shaded classes belong exclusively to the *sales* application system, light shaded classes are shared with other application systems.

Task classes coordinate the cooperation of conceptual classes and/or other task classes when executing a task automated fully or partly. In other words, task classes specify the work-flow within a business application system. The initial structure of the schema of task classes is almost identical to the most detailed level of the task-event schema of the corresponding business process model. Tasks lead to task classes, internal events and transactions lead to *interacts_with* relationships. Therefore figure 10 illustrates the schema of task classes too. The shaded areas delimit the schema of task classes for the *sales* business application system as well as for *store* and *finances*.

2. A distributed system is an integrated system which pursues a set of joint goals. It consists of multiple autonomous components which cooperate in pursuing the goals. There is no need for a component which has global control of the system [Ens78].

   Starting with a business process model with business objects loosely coupled by business transactions, the SOM methodology leads to a specification of distributed business application systems in a very natural way. Initially, each conceptual class and each task class derived from a business process model is an autonomous component. During the further specification process classes may be merged due to domain-specific reasons. For instance

in figure 13 *debit* and *invoice* are merged to reduce redundancy of attributes, in fig. 102 *invoice>* and *debit>* are merged to avoid sources of functional inconsistency.

3. The most common way to integrate application systems is data integration. Several application systems share a common database, the functions of the application systems operate on this database via external views. Although this kind of integration preserves consistency and avoids redundancy of data, it is not sufficient to support flexibility and evolution of application systems. The SOM methodology completes the concept of data integration by the concept of object integration [Fers92, FeSi98]. This concept supports distributed application systems consisting of autonomous and loosely coupled sub-systems which themselves may be internally data integrated. To achieve consistency of the application system as a whole, the sub-systems exchange messages according to detailed communication protocols. These protocols are derived from the transaction-oriented coordination of business objects as specified in the business process models.

4. The SOM methodology uses similar structures of distributed systems at the business process model layer and the business application systems layer [FeSi96]. A balanced and synchronized development of business process models and business application systems allows a simultaneous evolution of both layers during their life cycle [FeSi97]. There is a strong need that local changes in the business process model should only effect local changes in the business application systems. Both features, distributed systems at the two layers and the synchronized evolution, show that the business process model of a business system proves to be the backbone of a widespread architecture of business application systems.

6 Related Work

In literature and practice, there are several approaches to business process modeling. The approaches take different perspectives on a business system and specify models based on different views. The differences will be illustrated exemplary at the modeling languages IDEF and CIMOSA.

IDEF (Integration Definition) is a family of languages which evolved since the 1970s adapting to different modeling methods [MeMa98]. Applied to business systems, it basically covers the universe of discourse which is supported by an application system. Personal actors of a business system are not subject of these languages. With respect to the SOM enterprise architecture the IDEF languages refer to the model layers of business process model and specification of business application systems. They use traditional views on functions, data, and processes to specify structure and behavior of a system. The first language IDEF0 is based on the method Structured Analysis and Design Technique (SADT). It helps to specify the functions of the universe of discourse hierarchically. IDEF1X is suitable for modeling database schemes and IDEF3 is aimed at processes. IDEF4 refers to software design. IDEF5 as the end of the chain supports the construction of enterprise ontologies. It comes closest to the requests

taken up for business process modeling within the SOM methodology. The IDEF languages viewing functions, data, and processes fail to integrate the three views within a single object-oriented concept.

Another approach for modeling of business processes and business application systems corresponding to the model layers 2 and 3 of the SOM enterprise architecture are the CIMOSA languages [Verna98]. CIMOSA is an open system architecture for enterprise integration in manufacturing. Like the IDEF family the CIMOSA modeling languages also use views on functions, data, and processes to specify structure and behavior of a system. They supplement views on resources and organizational aspects. There are different types of flows within a system of business processes i.e. control flows defined as workflows, material flows and information flows. This approach also fails to integrate the views within an object oriented concept.

IDEF and CIMOSA views a business process as a sequence of activities (also called steps, process elements, functions), which are tied together by joint marks and which have to be equipped with resources [FeSi93, VoBe96]. From the viewpoint of the SOM methodology, IDEF and CIMOSA describe the behavior of a business system. In contrast, the SOM methodology specifies structure and behavior of a business system. The specification of structure consists of business objects and transactions and refers to the handling of goods and services. The coordination of the business objects involved in the handling of goods and services is specified explicitly.

7 Summary and Outlook

The previous sections give a brief introduction to the SOM methodology for business systems modeling. A comprehensive enterprise model consists of sub-models for each layer of the enterprise architecture (fig. 1). The sub-models are balanced carefully within the architectural framework. It is not necessary to start top down with the enterprise plan, followed by the business process model and ending with the specification of business application systems. The starting point depends on the goals pursued in the specific project.

More and more, enterprise models prove to be indispensable for business engineering, information management, and organization. Enterprise models following the SOM methodology show several characteristics which support the management of large enterprise models: (a) several model layers, each focusing on specific characteristics of a business system, (b) definition of views on each model layer, outside and inside perspectives, (c) different levels of abstraction and decomposition within a single model, and (d) notions with precise semantics which are arranged to meta models. Compared to other approaches of enterprise-wide modeling, e.g. enterprise-wide data modeling, a comprehensive model of a business system offers advantages and is more likely to be handled successfully.

There is a lot of research around the kernel of the SOM methodology which cannot be shown in this paper due to limitation of space. These features include management of complexity (i.e. decomposition of large business process models into models of main and service processes), reuse of model components (using patterns, reference models, application objects), tool support (for modeling, reporting, business process management, information management, work-flow management) [FeSi+94], an in depth consideration of distributed business processes and distributed business application systems [FeSi96] as well as first findings on virtual business processes [FeSi97].

8 Literature

Bahr92 **Bahrami H.:** The Emerging Flexible Organization: Perspectives from Silicon Valley. In: California Management Review, Summer 1992, p. 33 - 52

Beer81 **Beer S.:** The Brain of the Firm. 2nd Edition, Wiley, Chichester 1981

Ens78 **Enslow P.H.:** What is a 'Distributed' Data Processing System? In: IEEE Computer, Vol. 11, No. 1, January 1978, p. 13 - 21

Fers92 **Ferstl O.K.:** Integrationskonzepte betrieblicher Anwendungssysteme. Fachbericht Informatik 1/92. Universität Koblenz-Landau 1992

FeSi90 **Ferstl O.K., Sinz E.J.:** Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM). In: WIRTSCHAFTSINFORMATIK 32 (1990) 6, S. 566-581

FeSi91 **Ferstl O.K., Sinz E.J.:** Ein Vorgehensmodell zur Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM). In: WIRTSCHAFTSINFORMATIK 33 (1991) 6, S. 477-491

FeSi93 **Ferstl O.K., Sinz E.J.:** Geschäftsprozeßmodellierung. In: WIRTSCHAFTSINFORMATIK 35 (1993) 6, S. 589-592

FeSi98 **Ferstl O.K., Sinz E.J.:** Grundlagen der Wirtschaftsinformatik. Band 1, 3. Auflage, Oldenbourg, München 1998

FeSi95 **Ferstl O.K., Sinz E.J.:** Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen. In: WIRTSCHAFTSINFORMATIK 37 (1995) 3, S. 209 - 220

FeSi+94 **Ferstl O.K., Sinz E.J., Amberg, M., Hagemann, U., Malischewski, C.:** Tool-Based Business Process Modeling Using the SOM Approach. In: Wolfinger B. (Hrsg.): Innovationen bei Rechen- und Kommunikationssystemen. 24. GI-Jahrestagung im Rahmen des 13th World Computer Congress, IFIP Congress'94, Hamburg 28.8. - 2.9.94, Springer, Berlin 1994

FeSi96 **Ferstl O.K., Sinz E.J.:** Multi-Layered Development of Business Process Models and Distributed Business Application Systems - An Object-Oriented Approach. In: König W., Kurbel K., Mertens P., Preßmar D. (Hrsg.): Distributed Information Systems in Business. Springer, Berlin 1996, p. 159 - 179

FeSi97 **Ferstl O.K., Sinz E.J.:** Flexible Organizations Through Object-Oriented and Transaction-oriented Information Systems. In: Krallmann H. (Hrsg.): Wirtschaftsinformatik '97. Internationale Geschäftstätigkeit auf der Basis flexibler Organisationsstrukturen und leistungsfähiger Informationssysteme. Physica-Verlag, Heidelberg 1997, S. 393 - 411

Kru97 **Krumbiegel J.:** Integrale Gestaltung von Geschäftsprozessen und Anwendungssystemen in Dienstleistungsbetrieben. Deutscher Universitätsverlag, Wiesbaden 1997

MeMa98 **Menzel Ch., Mayer R.J.:** The IDEF Family of Languages. In: Bernus P., Mertins K., Schmidt G. (ed.): Handbook on Architectures of Information Systems

Sinz97 **Sinz E.J.**: Architektur betrieblicher Informationssysteme. In: Rechenberg P., Pomberger G. (Hrsg.): Informatik-Handbuch, Hanser-Verlag, München 1997, S. 875 - 887

Verna98 **Vernadat F.B.:** The CIMOSA Languages. In: Bernus P., Mertins K., Schmidt G. (ed.): Handbook on Architectures of Information Systems

VoBe96 **Vossen G., Becker J. (Hrsg.):** Geschäftsprozeßmodellierung und Workflow-Management. International Thomson Publishing, Bonn 1996