MEMO: A Tool Supported Methodology for Analyzing and (Re-) Designing Business Information Systems

Ulrich Frank
Institut für Wirtschaftsinformatik, Universität Koblenz
Rheinau 1, 56075 Koblenz, Germany
Email: ulrich.frank@informatik.uni-koblenz.de

Abstract

The paper presents a conceptual framework as well as a design environment to develop object-oriented enterprise models. It helps to coordinate the design of a business information system with the modelling of corporate strategies and organizational (re-) design. For this purpose the framework introduces concepts for illustratively modelling and integrating three main views on the enterprise. The views focus on corporate strategy, business (process) organization, and information system design. Thereby the methodology contributes to overcome the communication barriers that commonly exist between the different views. The integration of the various stages of system development is promoted by linking concepts of different views. Since business information systems are usually not built from scratch the methodology provides means for integrating existing software or data structures. The development environment that is based on the framework allows the user to navigate through the views of an enterprise model on various levels of detail. It maintains a model's integrity and allows for simulation and fast prototyping.

1 The Need for Models of the Enterprise

Designing, implementing and maintaining business information systems faces a number of challenges. On the software level it is - among others - desirable to support integration on a high level of semantics, to foster integrity and to allow for convenient reuse of existing artifacts. Penetrating a company with information technology allows for or may require new ways to organize the business. During the last years different aspects of this problem have been addressed by a variety of approaches. They are related to notions like "paperless office", "lean management" or "business (process) re-engineering" ([Hammer 93], [Talwar 93]). Reorganizing a firm or parts of it should be compatible with its long term goals. On the other hand the range of strategic options is more or less influenced by the available information technology - no matter whether you regard it as a "strategic weapon" ([Porter/Millar 85], [Wiseman 85]) or as a constraint.

The various approaches to deal with those different challenges usually have one characteristic in common: they are based on models - either of the whole enterprise or of parts of it. It has been well accepted for long that conceptual models are crucial for designing and implementing integrated information systems. While those models used to be data-oriented, object-oriented design methodologies are currently gaining more and more attention. Methodologies to support the analyst with business re-engineering are usually based on models as well. They are mainly focused on business processes ([Davenport 90], [Dennis 94]). Furthermore there is a wide range of models to help with analyzing and shaping a firm's strategy. They usually stress a more abstract view with highly aggregated data (for an overview see [Hassey 92] and [Scott Morton 86]). [Keen 91] explicitly promotes the use of information technology to develop corporate strategies. All these models have been introduced to reduce the complexity of strategic planning in order to help the analyst to concentrate on the essentials - and to communicate them to others who should be involved.

It is no surprise that strategic, organizational and information system models are usually based on

Published in: Ege, R.; Singh, M.; Meyer, B. (Eds.): Technology of Object-Oriented Languages ans Systems. Englewood Cliffs 1994, S. 367-380

different concepts. However, treating these different aspects independently bares the risk of redundant work and friction - a well known phenomenon for long. In order to allow for a more synergetic approach a number of authors ([Zachman 87], [ESPRIT 91], [Katz 90], [Sowa 92], [Peters 93]) have suggested enriched modelling frameworks - often named "enterprise modelling" (a term which is however not used in a unique way). Such methodologies differentiate between a number of views on the enterprise and intend to capture the relationships between these views. Studying them however shows that they either remain on a rather abstract level (for instance: [Sowa 92], [Zachman 87]) or lack sophisticated concepts - both from a managerial and a software engineering point of view (they are usually rather data- than object-oriented).

This paper presents a methodology as well as a set of integrated tools for developing object-oriented enterprise models that cover the main aspects of analyzing, designing, implementing and maintaining business information systems. It puts special emphasis on the following aspects:

- Different ways to conceptualize an enterprise in an illustrative way, called perspectives or views, are supported.
- An object-oriented design methodology that is specially suited for modelling business information systems.
- Concepts to support the integration of existing components, like applications or data structures, are provided.
- A systematic approach to analyze a firm's competitive position and to generate strategic options is included.
- There is support for (re-) designing information intensive business processes.
- An enterprise model can be very complex. However, for economic reasons there may be the need to be less ambitious - both in extent and detail. Therefore the methodology can be adapted to the constraints of a specific project.
- The development environment provides various ways of browsing through an enterprise model and maintaining its integrity. It also allows for fast prototyping and simulation.

2 Multi Purpose Enterprise Modelling

Inspired by the vision of highly integrated business information systems and additionally motivated by the various problems with existing systems a group of researchers at GMD started in 1990 to develop scenarios of how to deal with this complex challenge. Very soon it became obvious that it was a key issue to design multi-view models of the enterprise - mainly inspired by the framework presented in [Zachman 87]. We decided on three main views. The *strategic view* was to model the enterprise in a way that fits the perception common to senior executives. It should allow for illustratively describing a firm's competitive position and its strategic options. The organizational view was to be focussed on a company's organizational structure and the way its tasks/processes are performed. The *information system view* should provide the basic modelling constructs (in other words: the meta model for modelling) together with a framework to analyze existing systems and (re-) design and maintain them.

The methodology that has been developed in the following years - called *Multi Purpose Enterprise Modelling* (MEMO) - provides the analyst with various concepts to describe each view. The concepts are presented on different levels of detail and precision: a conceptual framework, guidelines to develop scenarios (mainly to describe tasks/processes), heuristics that help to identify important aspects, structured questionnaires to guide interviewing, and templates to gather formal aspects.

There are three dimensions to structure each of the three main views. *Stage* serves to describe a particular model's position within the continuum between analysis, design and maintenance. Each view is described in terms of the required resources, the operations or processes, the results which are produced, and the relevant features of external systems. This dimension is called *focus*. Usually it is desirable to model a view on a high level of abstraction. However, for certain types of analysis you may need to consider the state of instances. Within this dimension, called *aggregation*, MEMO allows the analyst to use concepts (like classes), particular instances and so called prototypical instances which represent the average state of a relevant set of instances (like the salary of the "average clerk").

2.1 The Strategic View

A methodology to guide analysis and design of corporate strategies should be suited to represent the required concepts in a way that is familiar to those who are commonly dealing with strategic planning - like senior executives. It should be based on generalized assumptions and should also allow to be con-

figured/specialized to a particular firm's needs. Among the wide range of methodologies (see [Scott Morton 86]) we found Porter's *value chain approach* to be most appropriate. Due to the complexity and contingency of the domain the methodology does not offer a precise guideline for developing strategies. However, Porter's approach provides the analyst with familiar concepts which support him to develop a solution of his problem in a systematic way. The value chain concept has gained a high degree of acceptance with many companies and consultants.

On the top level Porter models an enterprise as a system of "activities" which form a "value chain". "The value chain disaggregates a firm into its strategically relevant activities in order to understand the behavior of costs and the existing and potential sources of differentiation. A firm gains competitive advantage by performing these strategically important activities more cheaply or better than its competitors." ([Porter 85], p. 33) *Primary activities* are directly involved in the process to produce the products or services that are offered to a firm's customers. They include inbound logistics, operations, outbound logistics, marketing and sales and service. *Support activities* (firm infrastructure, human resource management, technology development and procurement) serve to support primary activities. An activity is an abstraction of actual business processes. Therefore it does not have to directly correspond to a specific process (for a detailed description see [Porter 85]).

In order to adapt Porter's approach to the MEMO framework we applied the three dimensions stage, focus and abstraction to it. There are two outstanding stages - with an arbitrary number of intermediate stages: the current competitive position and the position aimed at with the future strategy. Resources - like various types of capital or human resource - are used to perform activities. They are described on a high level of aggregation with emphasis on cost and quality issues. The outcome which is produced by an activity is modelled on a similar level - with aggregated figures for quality and price. Processes are basically described as a chain of activities with each activity characterized by the resources it consumes and its outcome. This results in a value chain being modelled as a graph of activities, resources, and outcomes with special emphasis on the interrelationships.

The relevant external world consists of more or less detailed value chains of competitors, suppliers and customers. In order to support the shaping of a firm's value chain the analyst is provided with heuristics like "How can the activity be performed differently or even eliminated?" or "How can a group of linked value activities be reordered or regrouped?" ([Porter 85], p. 110).

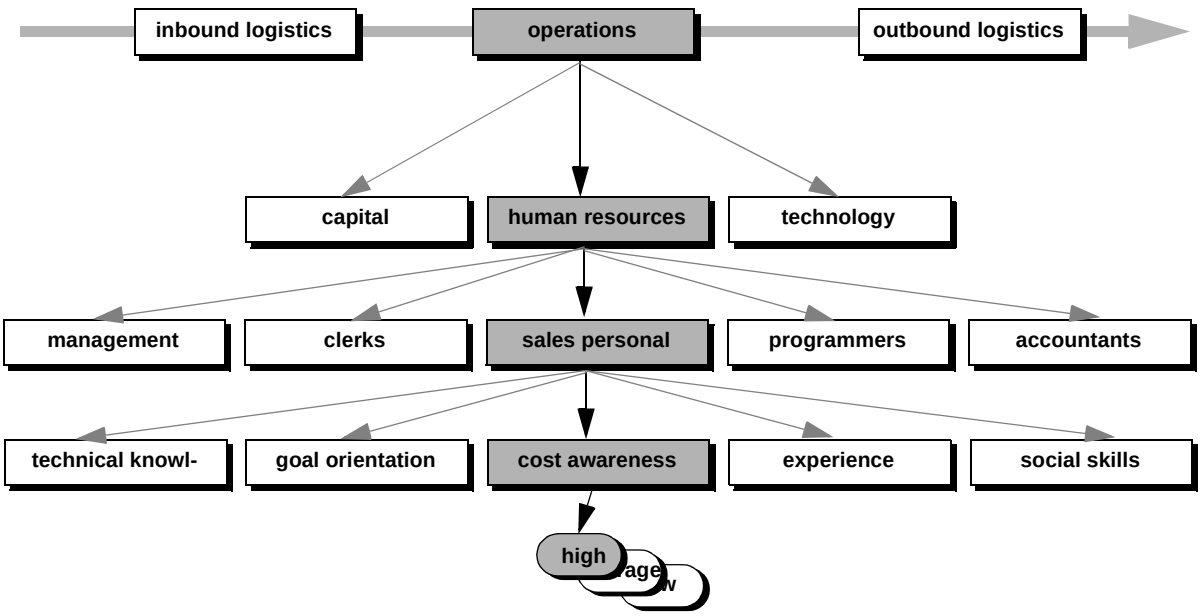

Figure 1. An example for the analysis of resources within the strategic view

Additionally Porter offers a top down approach that suggests to start with one out of a list of “generic strategies” and stepwise specialize it into a new value chain.

It is obvious that both analyzing and designing a value chain require special attention to the interrelationships, which Porter calls “linkages” ([Porter 85], p. 50): "Managing linkages thus is a more complex organizational task than managing value activities in themselves." While the methodology certainly does not allow for automatically generating a firm’s value chain, it can well be mapped to a specialized browser (see 3). The concepts used to describe the strategic view are represented in an object-oriented way according to the meta model characterized in 2.3.

2.2 The Organizational View

On the organizational level an enterprise model comprises a model of the organizational structure and models of business processes - both for analyzing the given state and for designing future states.

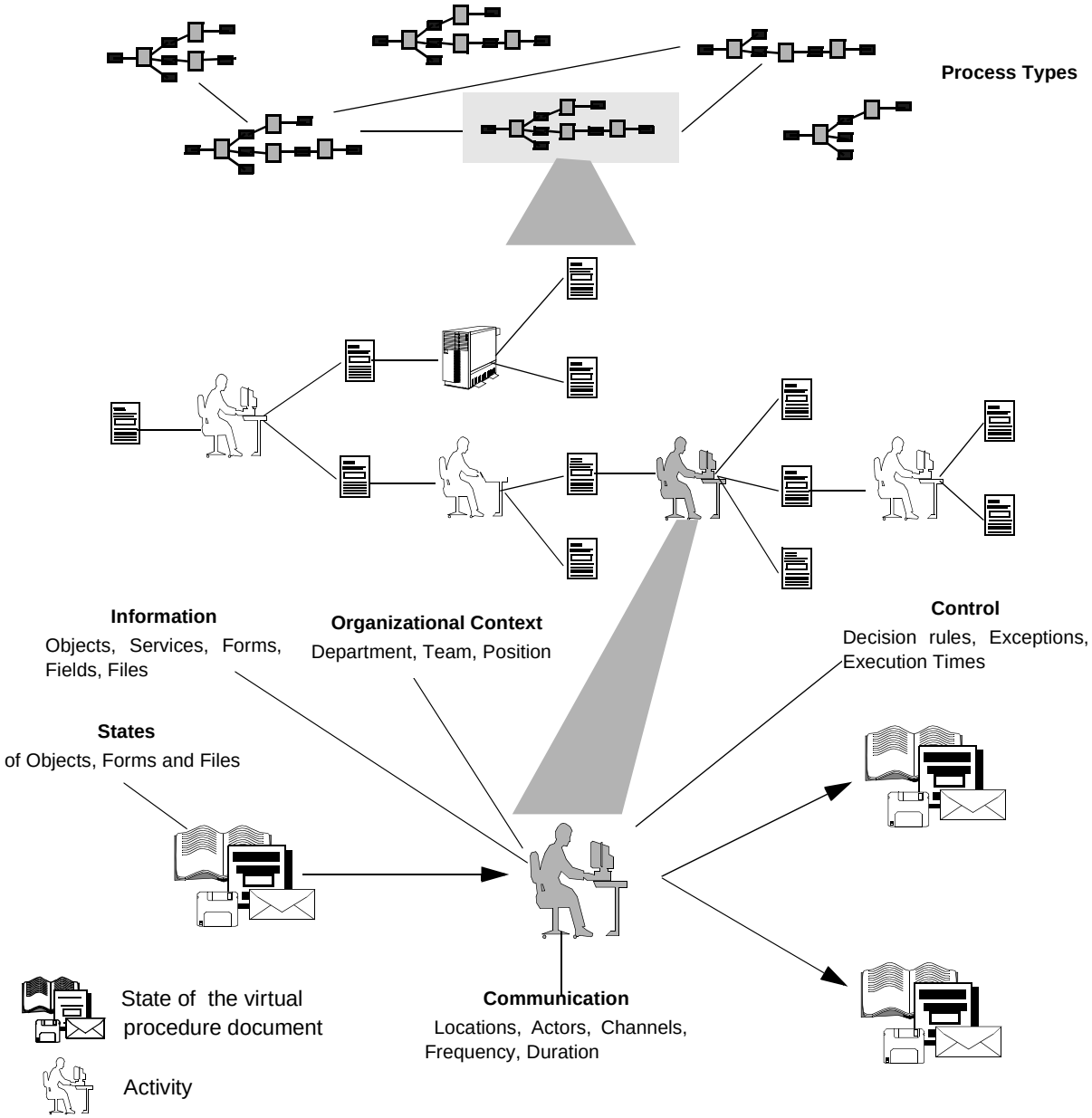

Figure 2. Conceptualization of business processes

The organizational structure is represented by organizational units (such as divisions, departments, groups, positions etc.) and their relationships ("reports to", "is part of" etc.). In addition to modelling a particular structure MEMO encourages the analyst to identify general rules for the division of labor and its coordination. Organizational resources are usually modelled as prototypical instances. Examples for resources on this level are people, buildings, furniture, machinery etc. Computer hardware may also be considered as resource within the organizational view. It is described by referring to the information system view (see below). Features which may be described within prototypical instances include various types of cost, availability, capacity etc. The number of resources and the detail of their description is subject to individual configuration. The external world is represented by relevant roles (like lawyer, consultant, customer, etc.) and services.

A business process is modelled as an ordered graph of subprocesses, where a subprocess in itself can be decomposed into other subprocesses. MEMO's emphasis is on office procedures. Therefore the "material" that is operated on is information. Information is grouped into three categories: objects which reside on the computer based information system, forms, and files. Information that is located in the information system is described by referring to the object model that is part of the information system view (see below). Forms have a formal structure, that is they contain fields, have well defined states (like 'complete', 'incomplete', 'consistent'), and a set of constraints defines the permissible operations. Their content may be changed within a subprocess. The term file is used to summarize documented information that is read-only - like office files, letters or journals. The information a business process operates on is gathered in a "virtual procedure document" that extends the traditional notion of a document in two respects: It may contain references to information that is located somewhere else. Furthermore it can be duplicated and thereby processed in parallel. Each subprocess is triggered by a certain state of the virtual procedure document and produces one or more new states.

Each process is assigned an *organizational context* by referring to the organizational units which are responsible for its execution. By default the organizational context is propagated to the subprocesses where it may be overwritten. Furthermore each process can be characterized by *business rules* like: "All subprocesses should be managed by the same person." The subprocesses can be described in a very detailed way - depending on the effort that is to be spent for analysis. Among the more important features are: minimum and maximum execution time, decision rules, required resources (for instance: copy machine, printer), exceptions, people (roles) needed to communicate with, communication media etc. In order to support the re-design of business processes MEMO provides the analyst with means to analyze existing processes and design heuristics. The model of a business process allows the analyst to detect media frictions (for instance: information that originally resides in the information system and is then being transferred to a form), to identify bottlenecks, or to draw communication nets. By adding statistical data on workload, capacity and probabilities of the subprocesses' possible outcome the model can be utilized to perform simulations (see fig. 8).

By focussing on processes the organizational model serves different purposes: It helps to redesign the business (if necessary), it helps to define the communication technology required to improve efficiency, and - last but not least - it supports to identify the objects/classes required to perform the relevant tasks. As with the strategic view the concepts of the organizational view are described with the object-oriented meta model (see below) in mind. Therefore they can be represented as an object model which fosters their mapping on a tool.

2.3 The Information System View

The information system view includes both a model of the existing system and the system that is to be designed. They are separated into an object-oriented conceptual model and a model of information system resources (like hardware, networks, operating systems). An object-oriented conceptual model includes an object model and other models to express dynamic aspects. Among the still increasing number of object-oriented design methodologies (in a survey we did last year we found more than forty approaches) we felt most inspired by the ones proposed by [Booch 90], [Rumbaugh et al. 90], and [Jacobson et al. 92]. However, none of them was satisfactory for our purpose. While Rumbaugh et al.' methodology suffers from being somewhat superficial and not consequently object-oriented, Booch's approach seems to be overloaded by details of various programming languages - which,

in our opinion, should not be part of a general methodology for the design of *conceptual* models. Jacobson et al. are primarily focussing on analysis and put less emphasis on software engineering issues occurring during design. (for a comparison of important methodologies see [Frank 93], [Hong 93], [Monarchi 92]). Furthermore we were not satisfied with the way dynamic aspects are modelled within these approaches. State transition diagrams are often suggested to capture dynamic aspects. At first sight such diagrams seem to be appropriate for modelling automated business processes, since they allow the analyst to describe events and corresponding state changes. They are however restricted to events and state transitions which may occur during the lifetime of objects of one class. Within an office procedure however one usually needs objects of more than one class.

2.3.1 Conceptualizing Object Models

While from a (re-)using programmer´s point of view it is sufficient to describe an object solely by the services it provides analysis and design require a more detailed view. Within an object model one defines classes and associations between classes or between objects respectively. Like in most other methodologies the outstanding features of a class are attributes and services. An attribute is regarded as an object that is encapsulated within an object. We do not allow attributes - like [Coad 91] - to only hold references to external objects that have an existence of there own in the object space. An attributes semantics is primarily defined by its class. Furthermore a cardinality (using min-max notation) may be assigned. Assigning a default value allows for generating appropriate initialization operations.

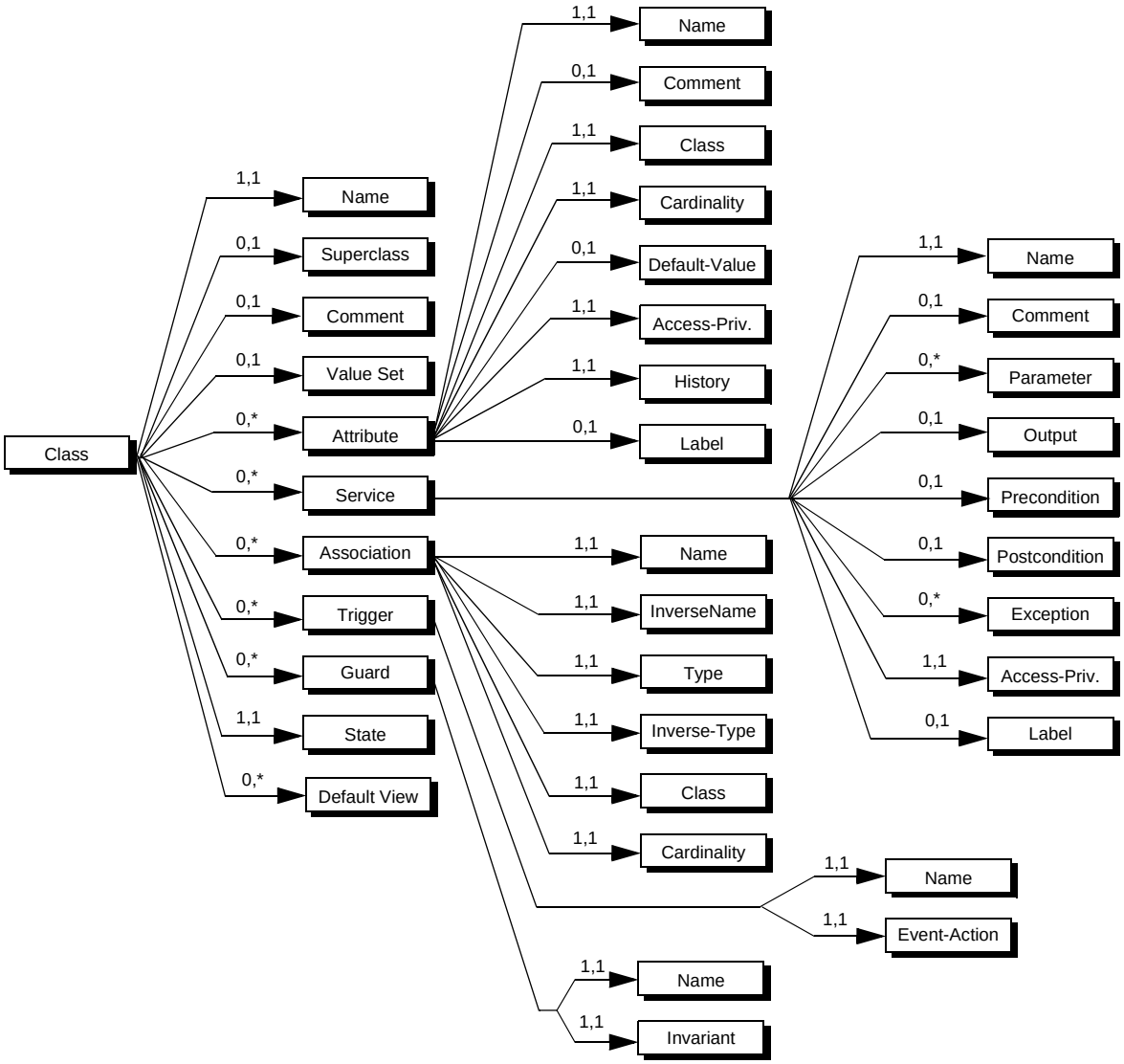

Figure 3. Part of the meta model for conceptualizing object models

Each attribute is also characterized by a history-flag. Setting it to true means that every update should be recorded somehow.

In order to allow for generating prototypical user-interfaces it is possible to assign a default-view to each class. A default view is either a widget or a collection of widgets. One can also define a label that is to be presented with the default view. Additionally features like size and font may be specified. This approach is a first attempt to deal with the complexity of user interaction. It cannot be completely satisfactory: the way a value of a certain class is presented to the user often is not unique but varies with the context of interaction.

In order to specify a service the designer may describe a list of input-parameters (which can be empty) where each parameter is characterized by its class and its name. If a service returns a result it has

A postcondition has to be fulfilled after the service has terminated. Similar to a precondition it can be defined by referring to an object state or to a state of the object that is returned by the service. Each service can be assigned a set of exceptions (like media errors) which should be named in a unified way for a whole object model. Thereby exception handling can be defined for all involved objects in the same way.

During the life time of an object there may be certain events and rules which go beyond the scope of a single service. For this purpose we introduce triggers and guards. A trigger can be generally defined as a tupel consisting of an event and an action. The event is specified by a condition that in turn is defined by referring to attribute states or states of objects which are returned by a service. The action is defined by the service that has to be executed when the event occurs.

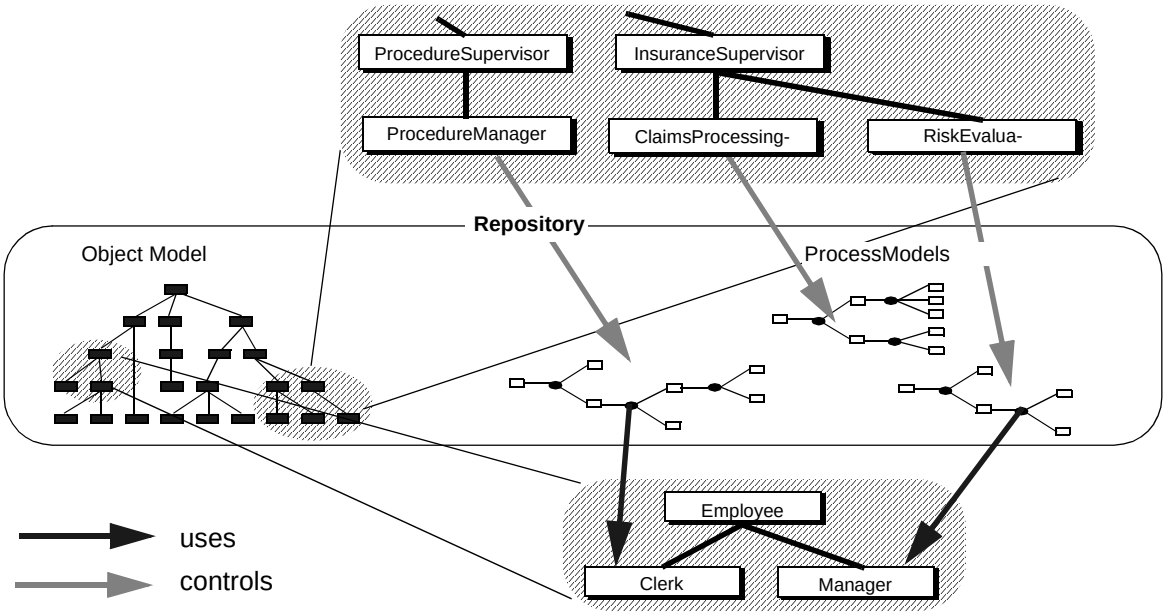

Figure 4. Integration of object model and office procedure models within an enterprise-wide repository

to be exactly one object. So it is sufficient to define the class of this object. It is important to note that such an object may be a composed object (like an array, a container etc.) that contains many other objects. A precondition in general specifies conditions "under which a routine will function properly" ([Meyer 88], p. 114). In our model it can be defined by referring to object or parameter states. For instance: For a service that requires an object of class ’Person’ as a parameter it may be necessary that the service ’sex’ delivers the state ’male’.

To give an example for a trigger: Whenever an object of class ’customerAccount’ has a balance less than x, the object should execute a service that is suited to notify somebody who is managing the account. A guard is a condition that has to be fulfilled during the lifetime of any object of a particular class (similar to what [Meyer 88], p. 124) calls "class invariant"). For instance: The value of attribute ’retailPrice’ within objects of class ’Product’ should never be lower than the value of attribute ’wholesalePrice’.

Every class may have exactly one superclass. Although there are a number of arguments in favor of multiple inheritance we restricted our model to single inheritance. We found that in most cases single inheritance is satisfactory while multiple inheritance increases the complexity of an object model and thereby makes it more difficult to maintain it in a consistent way. On the instance level MEMO distinguishes between two types of associations: interaction and aggregation. However, for a conceptual object model to be illustrative it is desirable to allow for a more detailed differentiation. For this reason each association has to be assigned a domain level identifier. Such identifiers do not include any semantics, they only improve readability of the model and allow for enhanced retrieval capabilities.

In order to allow for smoothly integrating existing elements - like applications or data structures - MEMO suggests that the modeller regards them as objects. Application classes are subclasses of the class "Application" that provides attributes and services to manage information like installation date, version, cost etc. An application class is usually modelled by the set of services it offers to the outside (which is rather poor for the average legacy application) and details about the communication protocol (for instance: OS-call, RPC, UDP or CORBA). Existing data structures, such as relation types of an RDBMS or a document file of a word processor, are represented as dummy classes: They do not encapsulate any attributes, instead they offer services that allow to access an existing element. For instance: class 'WordDocument' represents documents produced by a certain word processor. In order to integrate them with a more sophisticated model one can use an association with the reserved name "corresponds to". For instance: "RelTypeCustomer corresponds to Customer" or "WordDocument corresponds to CompoundDocument". Some existing applications use data that might be shared with other objects. Whenever such a potential source of redundancy is discovered it can be expressed in the model using an association with the reserved name: "should use". For instance: "AccountingSystem should use Customer". To avoid confusion about the implementation state of a model it is important to assign one of four predefined states to each class: "not implemented", "implemented", "dummy", "encapsulated". Fig. 3 gives an overview of the concepts proposed for designing object models. For a more comprehensive documentation see [Frank 92], [Frank 94].

2.3.2 Modelling dynamic aspects

A methodology for modelling dynamic behavior should allow for conveniently expressing temporal and control (in other words: dynamic) aspects. It should also help to avoid inconsistencies, like non terminating cycles, tasks which cannot be reached by any chance, deadlocks, etc. Unlike the object-oriented methodologies mentioned above Peters and Schultz [Peters 93] propose a modelling technique that allows for including objects of more than one class. They use Petri nets where each transition represents a state transition of an object of a particular class. Different transitions within a net may represent objects of different classes. Furthermore they allow - different from traditional concepts - transitions to have an execution time larger than zero. Mapping transitions to operations of an object of a certain class is attractive from a software-engineering point of view since it supports the idea of building procedures by 'glueing' objects together (as it is proposed in [Nierstrasz 90]). Nevertheless such an approach has its deficiencies, too. It is important for a model to allow for direct correspondence to familiar conceptualizations of the relevant domain. Business processes are not necessarily structured in a way that there is always only one object operated on within a subtask.

The approach we have chosen is similar to the one suggested by Peters and Schultz in that we also use semantically enriched Petri nets and allow transitions to have an execution time larger than zero. Our methodology however allows to explicitly assign objects of many different classes to one transition. The information system model of a process is very similar to the model of a business process within the organizational level - and in fact may be deducted from it. The main difference: On the information system level a process is described only by its automated parts. That implies tighter constraints on the specification of the information being processed: only information that is represented in the object model can be regarded.

In order to provide the model with prototyping capabilities it is necessary to enhance it with information about a suitable user-interface. This information can be deducted from those services assigned to a subtask. The widgets needed to interact with the services can be looked up in the object model (see above). Since every suitable class in the object model should have a default view assigned to it, a prototypical user-interface for an activity can be generated.

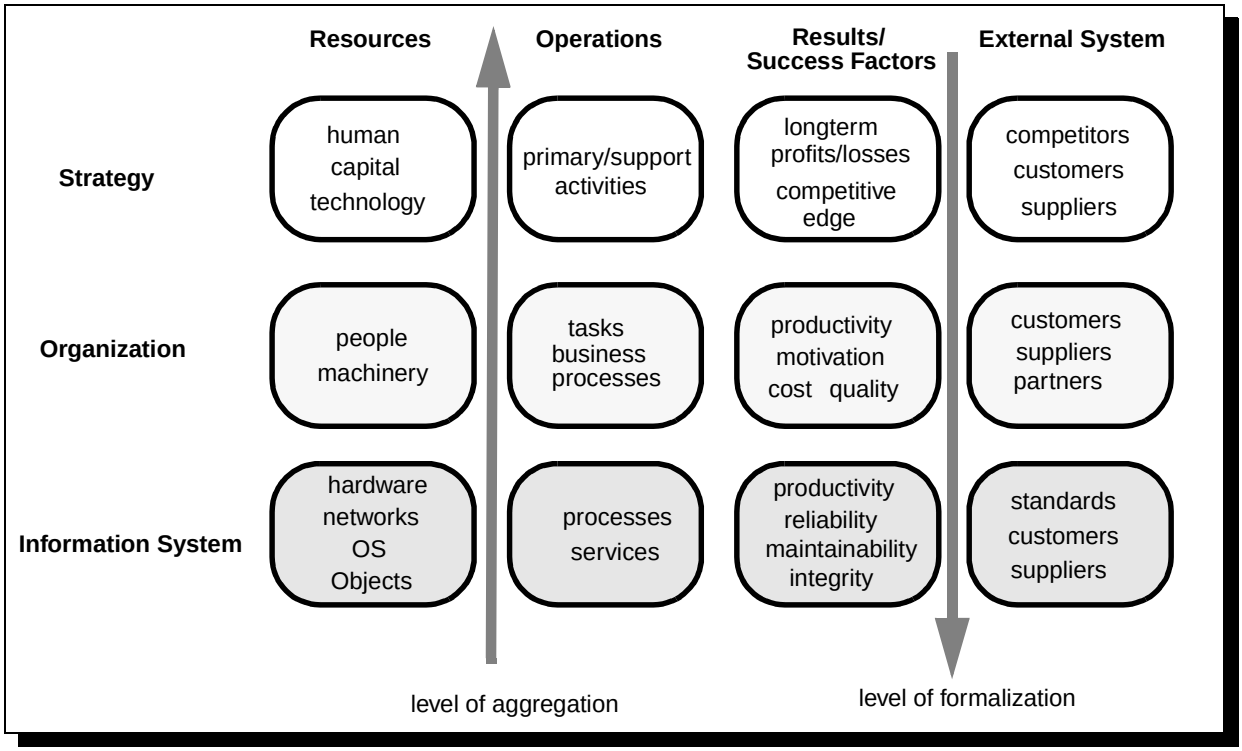

Figure 5. Selected concepts of different views and foci

Procedure models are integrated with an object model in two ways. First they refer to the objects they use, second the management of an office procedure is done by objects that are specified within the object model themselves. Fig. 4 shows how an object model and office procedure models could be represented within an enterprise-wide repository.

2.4 Integrating the Views

While it might be an intriguing vision to deduct the model of the information system from the organizational model and the organizational model from the strategic model, we did not even try to accomplish such a level of integration. This is mainly for two reasons. First: there is hardly general knowledge of how to deduct the organizational concept best suited to accomplish a strategic goal (which is also the case for the relationship between organizational model and information system model). Second: usually a strategy cannot be developed independently from organizational options which in turn are influenced by information technology as well. For those reasons MEMO allows one to explicitly link concepts on different levels. Direct tight coupling is the case, if a concept of a certain view directly corresponds to a concept of another view. For instance: The concept "Customer" within the organizational view corresponds to the class "Customer" within the information system view - although the representations may vary in detail.

The other extreme would be indirect weak coupling. For instance: The goal "customer satisfaction" on the strategic level could be first linked to a model of the firm's order processing on the organizational level. From there one could establish a link to objects within the information system view that provide services compatible with certain EDI-standards.

3 MEMO-Center: The Design Environment

Designing enterprise models according to the proposed methodology can hardly be accomplished without appropriate tools: A complex model requires support for browsing and searching. Because of multiple integrity constraints maintenance that is solely done manually jeopardizes a model's consistency to an unacceptable extent. Finally it is impossible to do without tools when prototyping is to be accomplished. For these reasons we developed an environment based on the

MEMO framework. It was implemented using Smalltalk-80 within the Objectworks® 4.0 environment on Sun4-workstations. High productivity could be achieved by using additional class libraries (see [Frank 92] for more details). The current version runs under Objectworks® 4.1 and VisualWorks® - on all platforms the appropriate Parc Place Smalltalk machine is available for.

The environment consists of three main tools: The *Value Chain Designer* (VCD) serves to design and analyze models of a firm's value chain. The *Object Model Designer* (OMD) supports the specification of object models. It provides various features to search for certain elements of an object model and to browse through the model. The *Workflow Designer* (WFD) allows for conveniently modelling business processes. It provides functions for simulation, fast prototyping and organizational analysis. The three tools are tightly integrated. The integration on the conceptual level has already been characterized above. System integration is accomplished by locating the tools within one Smalltalk image. The concepts managed by the tools can be annotated by using an integrated hypertext system.

The VCD provides a browser through the different elements of a value chain. The description of prototypical instances (like "human resource") is supported by predefined value sets (like "high", "average", "low") which can be modified interactively. This is the case, too, for relationships between activities. They can be characterized using an extensible list of identifiers (like "supports", "supported by", "contains", etc.). In addition to a textual representation relationships can be visualized in a graphical way. Items managed within the VCD can be associated to related items within the other tools. For instance: "activity uses business process". The VCD controls referential integrity of a value chain. Furthermore it provides dialogs to guide with analyzing and re-designing a value chain.

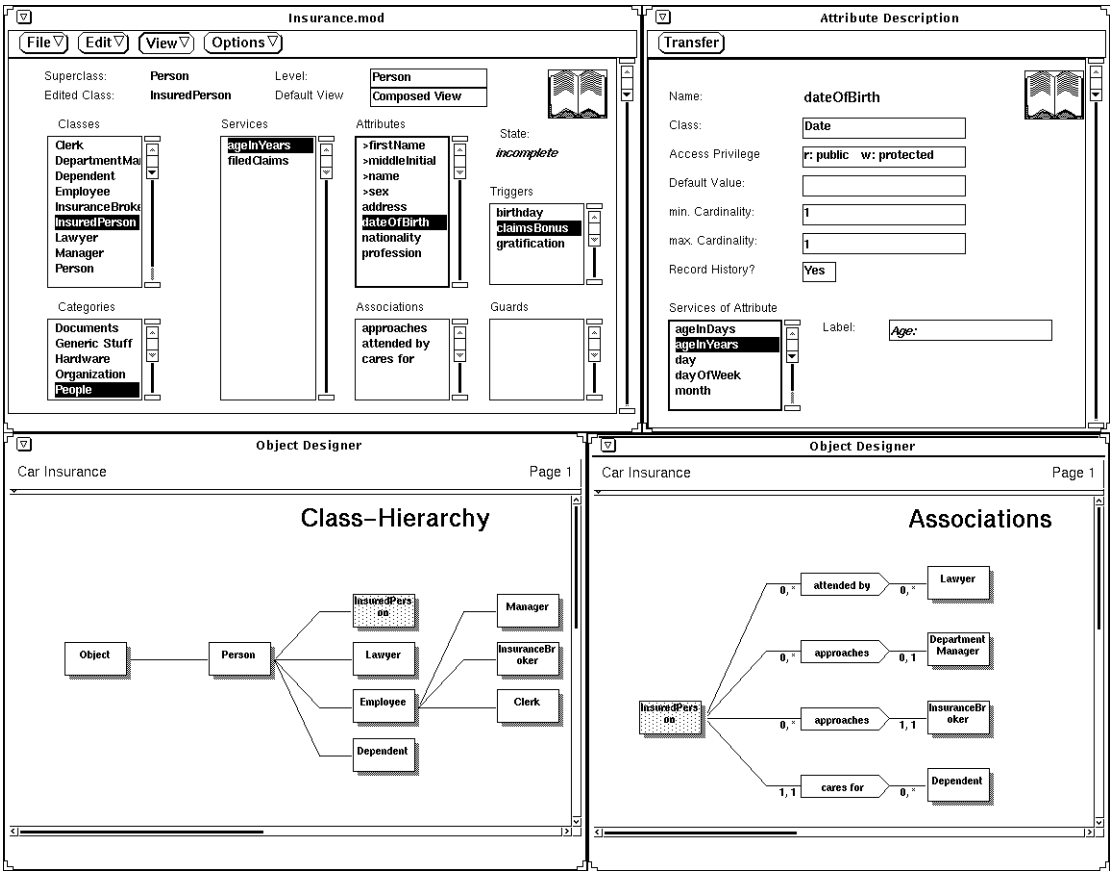

Figure 6. User-interface of the Object Model Designer

According to the meta model described above the OMD offers a number of different levels of abstraction. On the highest level class names are grouped into categories. On the next level a class can be assigned a list of attribute names, service names, guard names, and trigger names. Selecting an attribute, service, etc. causes the corresponding window to pop to the front. It allows for a more detailed description. Fig. 6 shows the window that contains the template for specifying an attribute - 'dateOfBirth' in the given example. Its class is 'Date'. The services which are provided by this class are shown in a listbox. If one of these services is needed for the class that is currently selected (which is 'InsuredPerson' in our example) it can be pasted to the services-listbox of this class - a convenient way to accomplish reusability. The OMD will then establish a reference to this service.

In order to foster system integrity it is not possible to type in a class name directly to characterize an attribute, a superclass, or a parameter. Instead it is required that the dictionary that contains all class names is updated first. Then the name can be pasted to the corresponding field. The OMD controls a number of integrity constraints. It prevents the user from deleting elements which are referenced by other elements, from assigning superclasses or classes of attributes in an inconsistent way, etc. The graphical representation of the class hierarchy can be used as an additional browser. The icons are mouse sensitive and cause the focus to shift to the selected class.

The main focus of the WFD is on business processes - both for analysis (that is mainly the organizational view) and design (organizational and information system view). On the highest level of abstraction the user can draw a business process picking from predefined icons within a specialized editor. When the procedure is described on this level (see fig. 7), the user can 'zoom' into it by selecting an icon - either a subtask or a document state. Within the example shown in fig. 7 the subtask 'Verification of substantial matter' is selected. Within the window titled 'Activity' it can be characterized by assigning execution times, an organizational unit, an organizational position, and possible exceptions. Furthermore the classes (like 'InsuredPerson', 'Policy', etc.), forms, and files which are needed as well as the involved roles can be listed in this window.

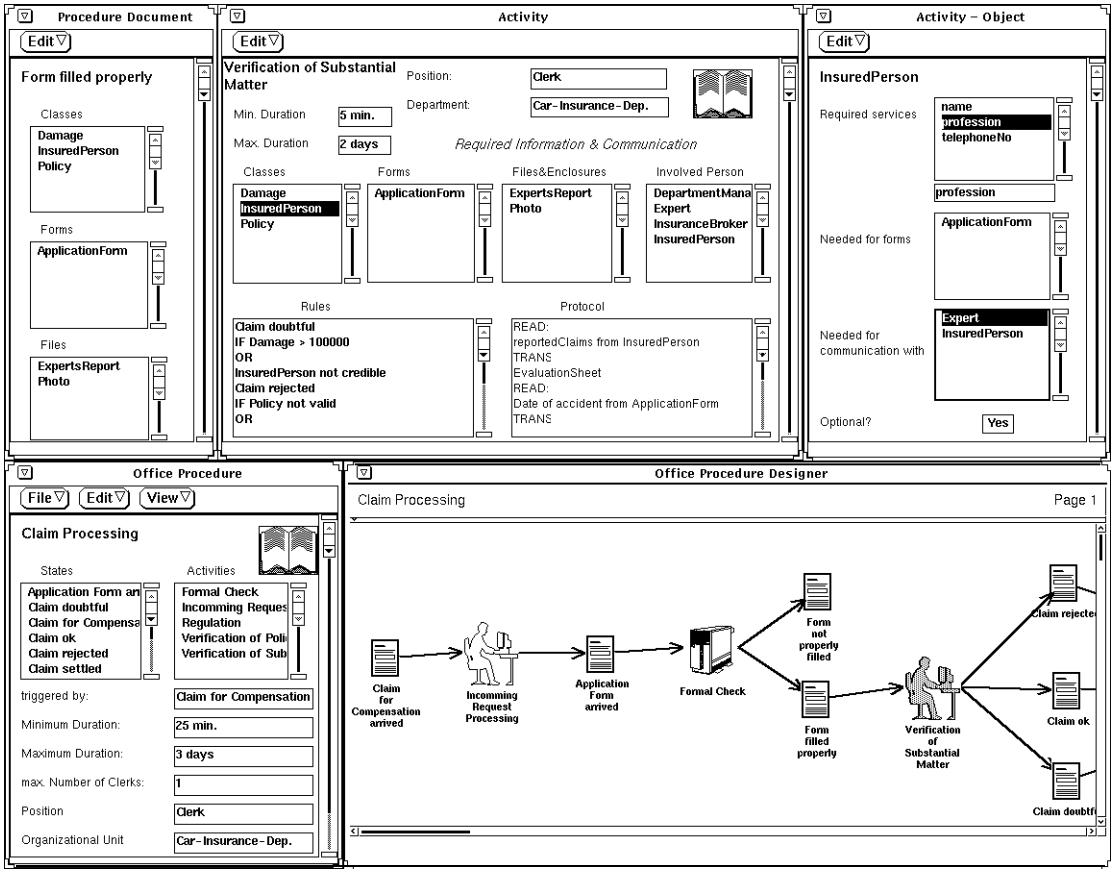

Figure 7. User-interface of the Workflow Designer

Selecting an item causes a window with an appropriate template to pop to the front. Within the example shown in fig. 7 this is the window in top right position. It allows to pick the required services from the selected class. For each service - or the object it delivers respectively - it can be specified what it is needed for: either for a form or for communicating with an involved person. The example shows that the service 'profession' is needed for the 'ApplicationForm' as well as for communicating with the 'InsuredPerson' and the 'Expert'.

Other templates exist to specify the relevant information within forms (fields together with an informal description of their purpose) or files (physical location, subject of interest within the file) and as well as the communication with involved persons (channel, subject). These specifications are used to generate a protocol which is shown in the right bottom corner of the 'Activity' window in fig. 7. Another text-widget within this window presents a template that can be filled to describe decision rules relevant for the focussed activity. A selected document state is specified in a similar way. First the classes, forms, and files which are involved in this state are assigned to it. Then these items have to be characterized in a more detailed way. The state of an object (which is represented by the name of its class) has to be defined by naming a boolean service that checks this state. That requires one to enhance the class with this service by switching to the OMD. For instance: An object of class 'Policy' is required to be in state 'valid'. That requires a service like 'isValid' to be specified for this class. For every form it has to be defined which fields have to be filled in. A file is characterized by its physical location and optionally the means of transportation to get it to the clerk. In order to support the user and to avoid inconsistencies the classes, forms and files assigned to a document state are pasted to the subtask that is triggered by this state. The WFD can generate a prototypical user-interface for every activity - provided the default views have been assigned to the corresponding classes in the object model (see above). The widgets placed within such an interface may be rearranged interactively. In order to use the simulation features built into the WFD it is necessary to first assign probabilities (using percentage values) to the states produced by every subtask. Furthermore it is required to type in the workload as number of procedures in a time period. After that the simulation can be started. Animation is accomplished by dynamically marking the subtasks which are active. If the simulation reveals any chances to improve the organization of the procedure the net can be rearranged interactively. Fig. 8 shows a snapshot of a simulation where the subtask 'Verification of substantial matter' has been expanded since it was found to be a bottleneck.

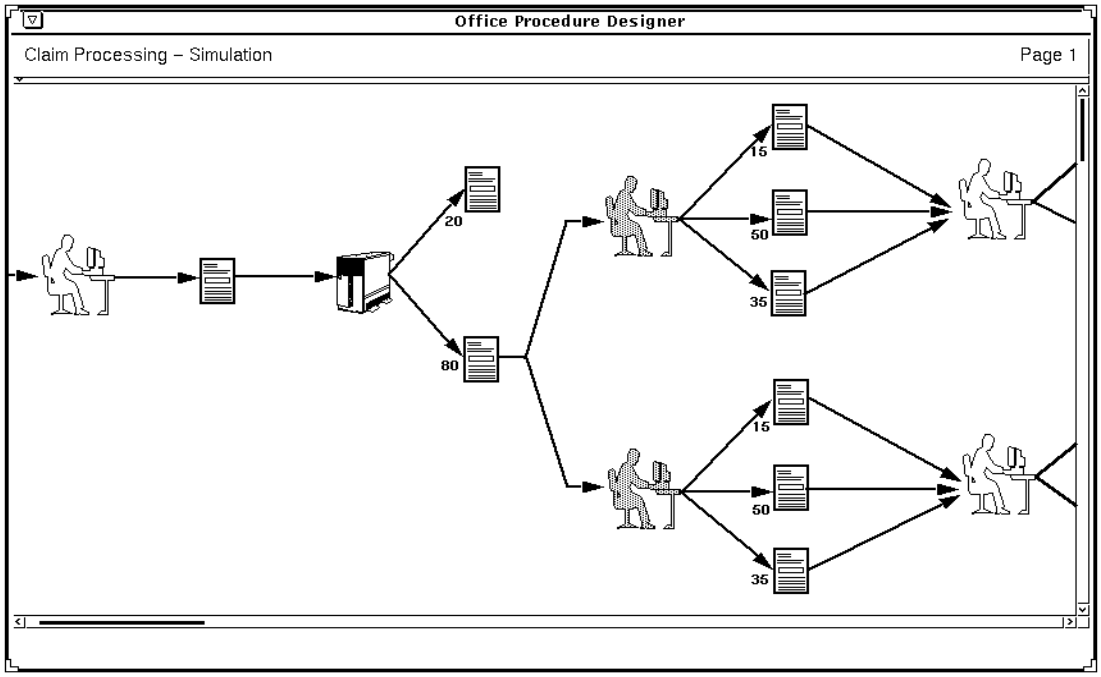

Figure 8. Snapshot of an animated simulation

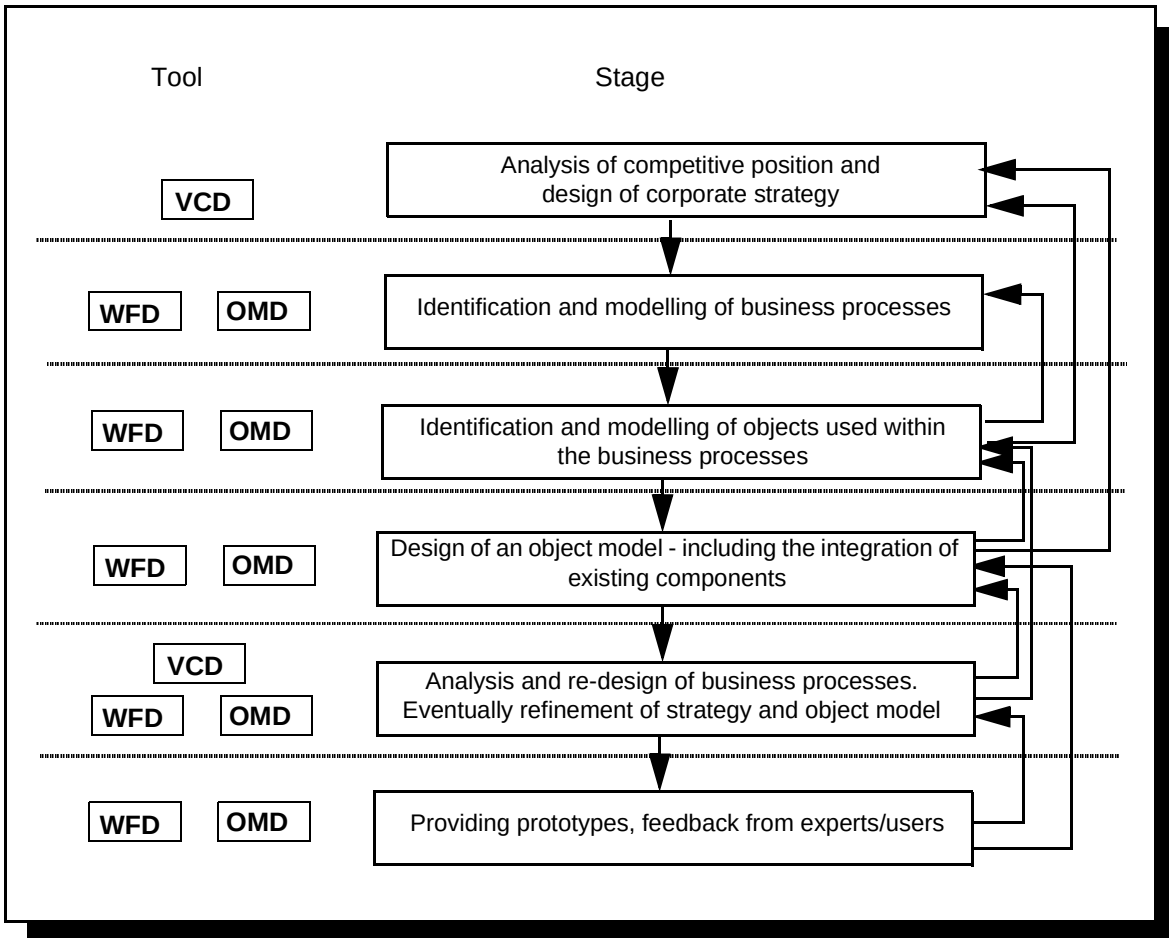

Figure 9. The usage of MEMO Center during important development stages (this is a simplified model!) and selected feedbacks between stages.

The WFD also allows for generating a report on detected media frictions and to graphically visualize communication relationships. Fig. 9 shows which tools are primarily used in the various stages of enterprise modelling.

4 Conclusions

Different from most other methodologies for object-oriented analysis and design MEMO not only focuses on the information system in itself but also on relevant strategic and organizational issues. Thereby it fosters a smooth integration of an information system with business processes and corporate strategies. Our experience with modelling office domains within insurance companies indicates that the proposed representations offer illustrative abstractions of an enterprise. This is especially the case for the representation of business processes. Both system analysts and domain experts intuitively understood the graphical notation. Thereby it proved to be a valuable medium for starting knowledge acquisition or object modelling respectively. We found that asking about strategies and processes is not only a prerequisite for organizational change. It furthermore helps to identify objects and specify their semantics. MEMO does - of course - not guarantee to optimize the organization of work. It can help however to reduce complexity by providing an illustrative representation of important aspects, by detecting certain types of organizational misconception, and - last but not least - promoting a sophisticated object-oriented architecture of a company's information system.

In the current version MEMO Center is restricted to one Smalltalk image. Therefore the prototypical workflow systems only simulate distribution. The objects used within the prototypes are either implemented directly in Smalltalk or refer to C-functions which are linked with the virtual machine. Currently we are working on extensions that will allow

for partial generation of distributed workflow management systems. For this purpose we use HP's "Distributed Smalltalk". It includes a CORBA implementation according to the OMG guidelines. MEMO Center will then allow for integrating any existing component that offers an IDL-interface. In order to allow for an automatic transformation of an object model designed according to MEMO into the OMG object model MEMO Center will be enhanced with means to distinguish between inheritance and subtyping. A text book with a comprehensive documentation of the methodology and the environment will be published by the end of 1994.

References

Booch, G.: Object-oriented design with applications. Redwood 1990

Coad, P.; Yourdon, E.: Object Oriented Design. Englewood Cliffs, NJ 1991

Davenport, T.; Short, J.E.: The New Industrial Engineering: Information Technology and Business Process Redesign. In: Sloan Management Review, Summer 1990, pp. 11-27

Dennis, A.R.; Hayes, G.S.; Daniels, R.M.: Re-Engineering Business Process Modeling. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 27th Hawaii International Conference on System Sciences, Vol. IV. Los Alamitos, Ca. 1994, pp. 245-253

ESPRIT Consortium AMICE: CIM-OSA AD 1.0 Architecture Description. Brussels 1991

Frank, U.; Klein, S.: Three integrated tools for designing and prototyping object-oriented enterprise models. GMD Research Report No. 689, Sankt Augustin 1992

Frank, U.: A Comparison of two Outstanding Methodologies for Object-Oriented Design. GMD Research Report No. 779, Sankt Augustin 1993

Frank, U.: An Object-Oriented Methodology for Analyzing, Designing and Prototyping Office Procedures. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 27th Hawaii International Conference on System Sciences, Vol. IV. Los Alamitos, Ca. 1994, pp. 663-672

Hammer, M.; Champy, J.: Reengineering the Corporation. New York 1993

Hassey, D.E.: Glossary of Management Techniques. In: International Review of Strategic Management. Vol. 3, 1992, pp. 47-75

Hong, S.; Goor, G.: A Formal Approach to the Comparison of Object-Oriented Analysis and Design Methodologies. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 26th International Hawaii International Conferenc on System Sciences, Vol. III., Los Alamitos 1993, pp. 689-698

Jacobson, I.; Christerson, M; Jonsson, P; Overgaard, G.: Object-Oriented Engineering. A Use Case Driven Approach. Reading, Mass. 1992

Katz, R.L.: Business/enterprise modelling. In: IBM Systems Journal, Vol. 29, No. 4, 1990, pp. 509-525

Keen, P.G.W.: Shaping the Future: Business Design through Information Technology. Cambridge 1991

Meyer, B.: Object-Oriented Software Construction. Prentice Hall 1988

Monarchi, D.E.; Puhr, G.: A Research Typology for Object-Oriented Analysis and Design. In: Communications of the ACM, Vol. 35, No. 9, 1992, pp. 35-47

Nierstrasz, O.; Dami, L.; De Mey, V.; Stadelmann, M.; Tsichritzis, D.; Vitek, J.: Visual Scripting: Towards Interactive Construction of Object-Oriented. In: Tsichritzis, D. C. (Hg.): Object Management. Geneva 1990, pp. 315-331

Peters, L.; Schultz, R.: The Application of Petri-Nets in Object-Oriented Enterprise Simulations. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 26th International Hawaii International Conferenc on System Sciences. Vol. III, Los Alamitos 1993, pp. 390-398

Porter, M.E.: Competitive Advantage. New York 1985

Porter, M.E.; Millar, V.E.: How Information Gives You Competitive Advantage. In: Harvard Business Review, July/August 1985, pp. 149-160

Rumbaugh et.al.: Object-Oriented Modelling and Design. Hemel-Hempstead 1990

Scott Morton, M.S.: Strategy formulation methodologies. Cambridge, Mass. 1986

Sowa, J.F.; Zachman, J.A.: Extending and formalizing the framework for information systems architecture. In: IBM Systems Journal, Vol. 31, No. 3, 1992, pp. 590-616

Talwar, R.: Business Re-engineering - a Strategy-driven approach. In: Long Range Planning, Vol. 26, No. 6, 1993, pp. 22-40

Wiseman, C.: Strategy and Computers: Information Systems as Competitive Weapons. Homewood, Ill. 1985

Zachman, J.A.: A framework for information systems architecture. In: IBM Systems Journal, Vol. 26, No. 3, 1987, pp. 277-293

MEMO: A Tool Supported Methodology for Analyzing and (Re-) Designing Business Information Systems

Ulrich Frank
Institut für Wirtschaftsinformatik, Universität Koblenz
Rheinau 1, 56075 Koblenz, Germany
Email: ulrich.frank@informatik.uni-koblenz.de

Abstract

The paper presents a conceptual framework as well as a design environment to develop object-oriented enterprise models. It helps to coordinate the design of a business information system with the modelling of corporate strategies and organizational (re-) design. For this purpose the framework introduces concepts for illustratively modelling and integrating three main views on the enterprise. The views focus on corporate strategy, business (process) organization, and information system design. Thereby the methodology contributes to overcome the communication barriers that commonly exist between the different views. The integration of the various stages of system development is promoted by linking concepts of different views. Since business information systems are usually not built from scratch the methodology provides means for integrating existing software or data structures. The development environment that is based on the framework allows the user to navigate through the views of an enterprise model on various levels of detail. It maintains a model's integrity and allows for simulation and fast prototyping.

1 The Need for Models of the Enterprise

Designing, implementing and maintaining business information systems faces a number of challenges. On the software level it is - among others - desirable to support integration on a high level of semantics, to foster integrity and to allow for convenient reuse of existing artifacts. Penetrating a company with information technology allows for or may require new ways to organize the business. During the last years different aspects of this problem have been addressed by a variety of approaches. They are related to notions like "paperless office", "lean management" or "business (process) re-engineering" ([Hammer 93], [Talwar 93]). Reorganizing a firm or parts of it should be compatible with its long term goals. On the other hand the range of strategic options is more or less influenced by the available information technology - no matter whether you regard it as a "strategic weapon" ([Porter/Millar 85], [Wiseman 85]) or as a constraint.

The various approaches to deal with those different challenges usually have one characteristic in common: they are based on models - either of the whole enterprise or of parts of it. It has been well accepted for long that conceptual models are crucial for designing and implementing integrated information systems. While those models used to be data-oriented, object-oriented design methodologies are currently gaining more and more attention. Methodologies to support the analyst with business re-engineering are usually based on models as well. They are mainly focused on business processes ([Davenport 90], [Dennis 94]). Furthermore there is a wide range of models to help with analyzing and shaping a firm's strategy. They usually stress a more abstract view with highly aggregated data (for an overview see [Hassey 92] and [Scott Morton 86]). [Keen 91] explicitly promotes the use of information technology to develop corporate strategies. All these models have been introduced to reduce the complexity of strategic planning in order to help the analyst to concentrate on the essentials - and to communicate them to others who should be involved.

It is no surprise that strategic, organizational and information system models are usually based on

Published in: Ege, R.; Singh, M.; Meyer, B. (Eds.): Technology of Object-Oriented Languages ans Systems. Englewood Cliffs 1994, S. 367-380

different concepts. However, treating these different aspects independently bares the risk of redundant work and friction - a well known phenomenon for long. In order to allow for a more synergetic approach a number of authors ([Zachman 87], [ESPRIT 91], [Katz 90], [Sowa 92], [Peters 93]) have suggested enriched modelling frameworks - often named "enterprise modelling" (a term which is however not used in a unique way). Such methodologies differentiate between a number of views on the enterprise and intend to capture the relationships between these views. Studying them however shows that they either remain on a rather abstract level (for instance: [Sowa 92], [Zachman 87]) or lack sophisticated concepts - both from a managerial and a software engineering point of view (they are usually rather data- than object-oriented).

This paper presents a methodology as well as a set of integrated tools for developing object-oriented enterprise models that cover the main aspects of analyzing, designing, implementing and maintaining business information systems. It puts special emphasis on the following aspects:

- Different ways to conceptualize an enterprise in an illustrative way, called perspectives or views, are supported.
- An object-oriented design methodology that is specially suited for modelling business information systems.
- Concepts to support the integration of existing components, like applications or data structures, are provided.
- A systematic approach to analyze a firm's competitive position and to generate strategic options is included.
- There is support for (re-) designing information intensive business processes.
- An enterprise model can be very complex. However, for economic reasons there may be the need to be less ambitious - both in extent and detail. Therefore the methodology can be adapted to the constraints of a specific project.
- The development environment provides various ways of browsing through an enterprise model and maintaining its integrity. It also allows for fast prototyping and simulation.

2 Multi Purpose Enterprise Modelling

Inspired by the vision of highly integrated business information systems and additionally motivated by the various problems with existing systems a group of researchers at GMD started in 1990 to develop scenarios of how to deal with this complex challenge. Very soon it became obvious that it was a key issue to design multi-view models of the enterprise - mainly inspired by the framework presented in [Zachman 87]. We decided on three main views. The *strategic view* was to model the enterprise in a way that fits the perception common to senior executives. It should allow for illustratively describing a firm's competitive position and its strategic options. The organizational view was to be focussed on a company's organizational structure and the way its tasks/processes are performed. The *information system view* should provide the basic modelling constructs (in other words: the meta model for modelling) together with a framework to analyze existing systems and (re-) design and maintain them.

The methodology that has been developed in the following years - called *Multi Purpose Enterprise Modelling* (MEMO) - provides the analyst with various concepts to describe each view. The concepts are presented on different levels of detail and precision: a conceptual framework, guidelines to develop scenarios (mainly to describe tasks/processes), heuristics that help to identify important aspects, structured questionnaires to guide interviewing, and templates to gather formal aspects.

There are three dimensions to structure each of the three main views. *Stage* serves to describe a particular model's position within the continuum between analysis, design and maintenance. Each view is described in terms of the required resources, the operations or processes, the results which are produced, and the relevant features of external systems. This dimension is called *focus*. Usually it is desirable to model a view on a high level of abstraction. However, for certain types of analysis you may need to consider the state of instances. Within this dimension, called *aggregation*, MEMO allows the analyst to use concepts (like classes), particular instances and so called prototypical instances which represent the average state of a relevant set of instances (like the salary of the "average clerk").

2.1 The Strategic View

A methodology to guide analysis and design of corporate strategies should be suited to represent the required concepts in a way that is familiar to those who are commonly dealing with strategic planning - like senior executives. It should be based on generalized assumptions and should also allow to be con-

figured/specialized to a particular firm's needs. Among the wide range of methodologies (see [Scott Morton 86]) we found Porter's *value chain approach* to be most appropriate. Due to the complexity and contingency of the domain the methodology does not offer a precise guideline for developing strategies. However, Porter's approach provides the analyst with familiar concepts which support him to develop a solution of his problem in a systematic way. The value chain concept has gained a high degree of acceptance with many companies and consultants.

On the top level Porter models an enterprise as a system of "activities" which form a "value chain". "The value chain disaggregates a firm into its strategically relevant activities in order to understand the behavior of costs and the existing and potential sources of differentiation. A firm gains competitive advantage by performing these strategically important activities more cheaply or better than its competitors." ([Porter 85], p. 33) *Primary activities* are directly involved in the process to produce the products or services that are offered to a firm's customers. They include inbound logistics, operations, outbound logistics, marketing and sales and service. *Support activities* (firm infrastructure, human resource management, technology development and procurement) serve to support primary activities. An activity is an abstraction of actual business processes. Therefore it does not have to directly correspond to a specific process (for a detailed description see [Porter 85]).

In order to adapt Porter's approach to the MEMO framework we applied the three dimensions stage, focus and abstraction to it. There are two outstanding stages - with an arbitrary number of intermediate stages: the current competitive position and the position aimed at with the future strategy. Resources - like various types of capital or human resource - are used to perform activities. They are described on a high level of aggregation with emphasis on cost and quality issues. The outcome which is produced by an activity is modelled on a similar level - with aggregated figures for quality and price. Processes are basically described as a chain of activities with each activity characterized by the resources it consumes and its outcome. This results in a value chain being modelled as a graph of activities, resources, and outcomes with special emphasis on the interrelationships.

The relevant external world consists of more or less detailed value chains of competitors, suppliers and customers. In order to support the shaping of a firm's value chain the analyst is provided with heuristics like "How can the activity be performed differently or even eliminated?" or "How can a group of linked value activities be reordered or regrouped?" ([Porter 85], p. 110).

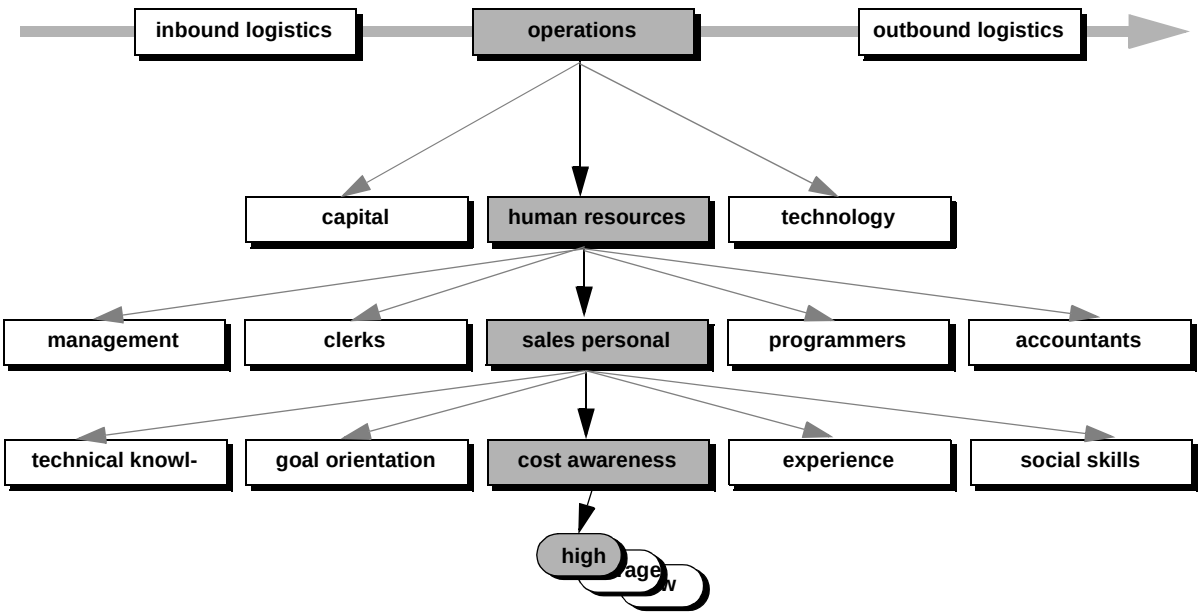

Figure 1. An example for the analysis of resources within the strategic view

Additionally Porter offers a top down approach that suggests to start with one out of a list of “generic strategies” and stepwise specialize it into a new value chain.

It is obvious that both analyzing and designing a value chain require special attention to the interrelationships, which Porter calls “linkages” ([Porter 85], p. 50): "Managing linkages thus is a more complex organizational task than managing value activities in themselves." While the methodology certainly does not allow for automatically generating a firm’s value chain, it can well be mapped to a specialized browser (see 3). The concepts used to describe the strategic view are represented in an object-oriented way according to the meta model characterized in 2.3.

2.2 The Organizational View

On the organizational level an enterprise model comprises a model of the organizational structure and models of business processes - both for analyzing the given state and for designing future states.

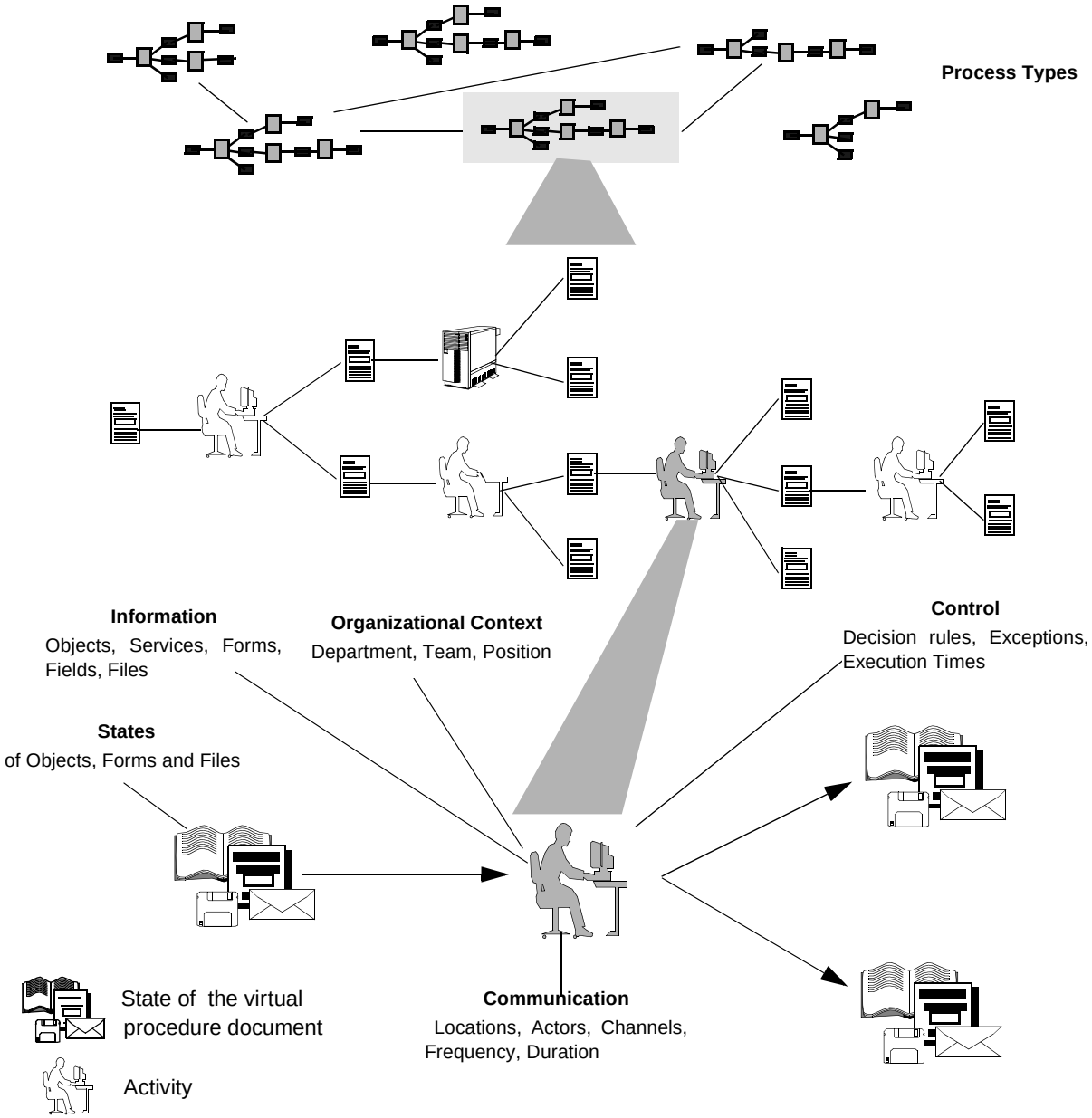

Figure 2. Conceptualization of business processes

The organizational structure is represented by organizational units (such as divisions, departments, groups, positions etc.) and their relationships ("reports to", "is part of" etc.). In addition to modelling a particular structure MEMO encourages the analyst to identify general rules for the division of labor and its coordination. Organizational resources are usually modelled as prototypical instances. Examples for resources on this level are people, buildings, furniture, machinery etc. Computer hardware may also be considered as resource within the organizational view. It is described by referring to the information system view (see below). Features which may be described within prototypical instances include various types of cost, availability, capacity etc. The number of resources and the detail of their description is subject to individual configuration. The external world is represented by relevant roles (like lawyer, consultant, customer, etc.) and services.

A business process is modelled as an ordered graph of subprocesses, where a subprocess in itself can be decomposed into other subprocesses. MEMO's emphasis is on office procedures. Therefore the "material" that is operated on is information. Information is grouped into three categories: objects which reside on the computer based information system, forms, and files. Information that is located in the information system is described by referring to the object model that is part of the information system view (see below). Forms have a formal structure, that is they contain fields, have well defined states (like 'complete', 'incomplete', 'consistent'), and a set of constraints defines the permissible operations. Their content may be changed within a subprocess. The term file is used to summarize documented information that is read-only - like office files, letters or journals. The information a business process operates on is gathered in a "virtual procedure document" that extends the traditional notion of a document in two respects: It may contain references to information that is located somewhere else. Furthermore it can be duplicated and thereby processed in parallel. Each subprocess is triggered by a certain state of the virtual procedure document and produces one or more new states.

Each process is assigned an *organizational context* by referring to the organizational units which are responsible for its execution. By default the organizational context is propagated to the subprocesses where it may be overwritten. Furthermore each process can be characterized by *business rules* like: "All subprocesses should be managed by the same person." The subprocesses can be described in a very detailed way - depending on the effort that is to be spent for analysis. Among the more important features are: minimum and maximum execution time, decision rules, required resources (for instance: copy machine, printer), exceptions, people (roles) needed to communicate with, communication media etc. In order to support the re-design of business processes MEMO provides the analyst with means to analyze existing processes and design heuristics. The model of a business process allows the analyst to detect media frictions (for instance: information that originally resides in the information system and is then being transferred to a form), to identify bottlenecks, or to draw communication nets. By adding statistical data on workload, capacity and probabilities of the subprocesses' possible outcome the model can be utilized to perform simulations (see fig. 8).

By focussing on processes the organizational model serves different purposes: It helps to redesign the business (if necessary), it helps to define the communication technology required to improve efficiency, and - last but not least - it supports to identify the objects/classes required to perform the relevant tasks. As with the strategic view the concepts of the organizational view are described with the object-oriented meta model (see below) in mind. Therefore they can be represented as an object model which fosters their mapping on a tool.

2.3 The Information System View

The information system view includes both a model of the existing system and the system that is to be designed. They are separated into an object-oriented conceptual model and a model of information system resources (like hardware, networks, operating systems). An object-oriented conceptual model includes an object model and other models to express dynamic aspects. Among the still increasing number of object-oriented design methodologies (in a survey we did last year we found more than forty approaches) we felt most inspired by the ones proposed by [Booch 90], [Rumbaugh et al. 90], and [Jacobson et al. 92]. However, none of them was satisfactory for our purpose. While Rumbaugh et al.' methodology suffers from being somewhat superficial and not consequently object-oriented, Booch's approach seems to be overloaded by details of various programming languages - which,

in our opinion, should not be part of a general methodology for the design of *conceptual* models. Jacobson et al. are primarily focussing on analysis and put less emphasis on software engineering issues occurring during design. (for a comparison of important methodologies see [Frank 93], [Hong 93], [Monarchi 92]). Furthermore we were not satisfied with the way dynamic aspects are modelled within these approaches. State transition diagrams are often suggested to capture dynamic aspects. At first sight such diagrams seem to be appropriate for modelling automated business processes, since they allow the analyst to describe events and corresponding state changes. They are however restricted to events and state transitions which may occur during the lifetime of objects of one class. Within an office procedure however one usually needs objects of more than one class.

2.3.1 Conceptualizing Object Models

While from a (re-)using programmer´s point of view it is sufficient to describe an object solely by the services it provides analysis and design require a more detailed view. Within an object model one defines classes and associations between classes or between objects respectively. Like in most other methodologies the outstanding features of a class are attributes and services. An attribute is regarded as an object that is encapsulated within an object. We do not allow attributes - like [Coad 91] - to only hold references to external objects that have an existence of there own in the object space. An attributes semantics is primarily defined by its class. Furthermore a cardinality (using min-max notation) may be assigned. Assigning a default value allows for generating appropriate initialization operations.

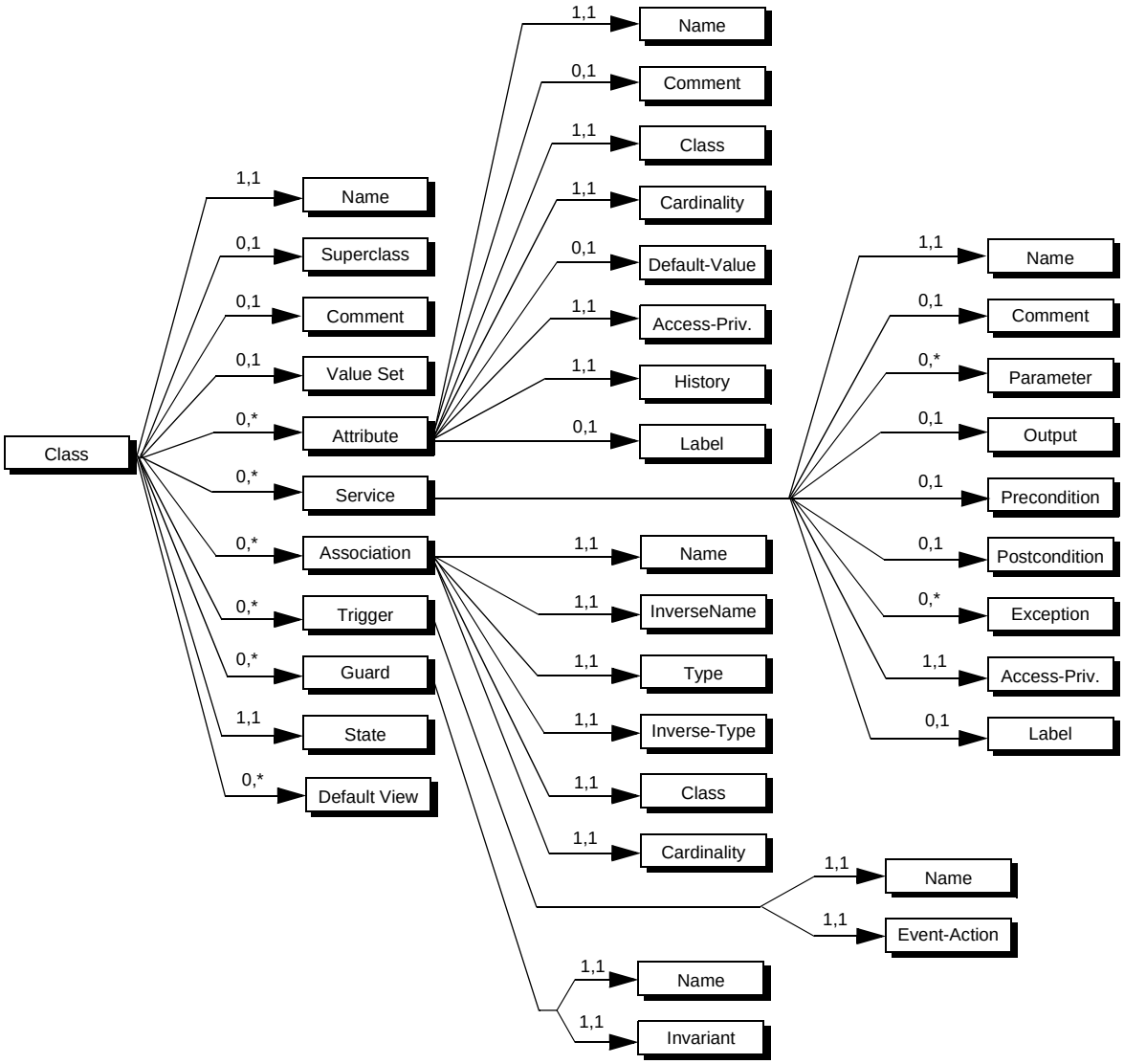

Figure 3. Part of the meta model for conceptualizing object models

Each attribute is also characterized by a history-flag. Setting it to true means that every update should be recorded somehow.

In order to allow for generating prototypical user-interfaces it is possible to assign a default-view to each class. A default view is either a widget or a collection of widgets. One can also define a label that is to be presented with the default view. Additionally features like size and font may be specified. This approach is a first attempt to deal with the complexity of user interaction. It cannot be completely satisfactory: the way a value of a certain class is presented to the user often is not unique but varies with the context of interaction.

In order to specify a service the designer may describe a list of input-parameters (which can be empty) where each parameter is characterized by its class and its name. If a service returns a result it has

A postcondition has to be fulfilled after the service has terminated. Similar to a precondition it can be defined by referring to an object state or to a state of the object that is returned by the service. Each service can be assigned a set of exceptions (like media errors) which should be named in a unified way for a whole object model. Thereby exception handling can be defined for all involved objects in the same way.

During the life time of an object there may be certain events and rules which go beyond the scope of a single service. For this purpose we introduce triggers and guards. A trigger can be generally defined as a tupel consisting of an event and an action. The event is specified by a condition that in turn is defined by referring to attribute states or states of objects which are returned by a service. The action is defined by the service that has to be executed when the event occurs.

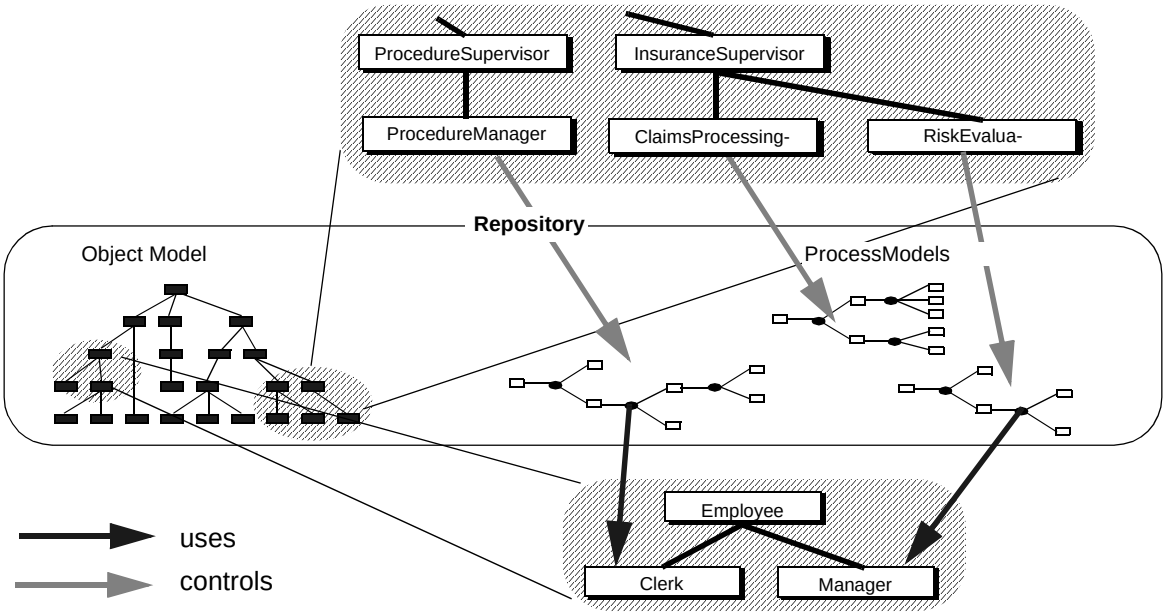

Figure 4. Integration of object model and office procedure models within an enterprise-wide repository

to be exactly one object. So it is sufficient to define the class of this object. It is important to note that such an object may be a composed object (like an array, a container etc.) that contains many other objects. A precondition in general specifies conditions "under which a routine will function properly" ([Meyer 88], p. 114). In our model it can be defined by referring to object or parameter states. For instance: For a service that requires an object of class ’Person’ as a parameter it may be necessary that the service ’sex’ delivers the state ’male’.

To give an example for a trigger: Whenever an object of class ’customerAccount’ has a balance less than x, the object should execute a service that is suited to notify somebody who is managing the account. A guard is a condition that has to be fulfilled during the lifetime of any object of a particular class (similar to what [Meyer 88], p. 124) calls "class invariant"). For instance: The value of attribute ’retailPrice’ within objects of class ’Product’ should never be lower than the value of attribute ’wholesalePrice’.

Every class may have exactly one superclass. Although there are a number of arguments in favor of multiple inheritance we restricted our model to single inheritance. We found that in most cases single inheritance is satisfactory while multiple inheritance increases the complexity of an object model and thereby makes it more difficult to maintain it in a consistent way. On the instance level MEMO distinguishes between two types of associations: interaction and aggregation. However, for a conceptual object model to be illustrative it is desirable to allow for a more detailed differentiation. For this reason each association has to be assigned a domain level identifier. Such identifiers do not include any semantics, they only improve readability of the model and allow for enhanced retrieval capabilities.

In order to allow for smoothly integrating existing elements - like applications or data structures - MEMO suggests that the modeller regards them as objects. Application classes are subclasses of the class "Application" that provides attributes and services to manage information like installation date, version, cost etc. An application class is usually modelled by the set of services it offers to the outside (which is rather poor for the average legacy application) and details about the communication protocol (for instance: OS-call, RPC, UDP or CORBA). Existing data structures, such as relation types of an RDBMS or a document file of a word processor, are represented as dummy classes: They do not encapsulate any attributes, instead they offer services that allow to access an existing element. For instance: class 'WordDocument' represents documents produced by a certain word processor. In order to integrate them with a more sophisticated model one can use an association with the reserved name "corresponds to". For instance: "RelTypeCustomer corresponds to Customer" or "WordDocument corresponds to CompoundDocument". Some existing applications use data that might be shared with other objects. Whenever such a potential source of redundancy is discovered it can be expressed in the model using an association with the reserved name: "should use". For instance: "AccountingSystem should use Customer". To avoid confusion about the implementation state of a model it is important to assign one of four predefined states to each class: "not implemented", "implemented", "dummy", "encapsulated". Fig. 3 gives an overview of the concepts proposed for designing object models. For a more comprehensive documentation see [Frank 92], [Frank 94].

2.3.2 Modelling dynamic aspects

A methodology for modelling dynamic behavior should allow for conveniently expressing temporal and control (in other words: dynamic) aspects. It should also help to avoid inconsistencies, like non terminating cycles, tasks which cannot be reached by any chance, deadlocks, etc. Unlike the object-oriented methodologies mentioned above Peters and Schultz [Peters 93] propose a modelling technique that allows for including objects of more than one class. They use Petri nets where each transition represents a state transition of an object of a particular class. Different transitions within a net may represent objects of different classes. Furthermore they allow - different from traditional concepts - transitions to have an execution time larger than zero. Mapping transitions to operations of an object of a certain class is attractive from a software-engineering point of view since it supports the idea of building procedures by 'glueing' objects together (as it is proposed in [Nierstrasz 90]). Nevertheless such an approach has its deficiencies, too. It is important for a model to allow for direct correspondence to familiar conceptualizations of the relevant domain. Business processes are not necessarily structured in a way that there is always only one object operated on within a subtask.

The approach we have chosen is similar to the one suggested by Peters and Schultz in that we also use semantically enriched Petri nets and allow transitions to have an execution time larger than zero. Our methodology however allows to explicitly assign objects of many different classes to one transition. The information system model of a process is very similar to the model of a business process within the organizational level - and in fact may be deducted from it. The main difference: On the information system level a process is described only by its automated parts. That implies tighter constraints on the specification of the information being processed: only information that is represented in the object model can be regarded.

In order to provide the model with prototyping capabilities it is necessary to enhance it with information about a suitable user-interface. This information can be deducted from those services assigned to a subtask. The widgets needed to interact with the services can be looked up in the object model (see above). Since every suitable class in the object model should have a default view assigned to it, a prototypical user-interface for an activity can be generated.

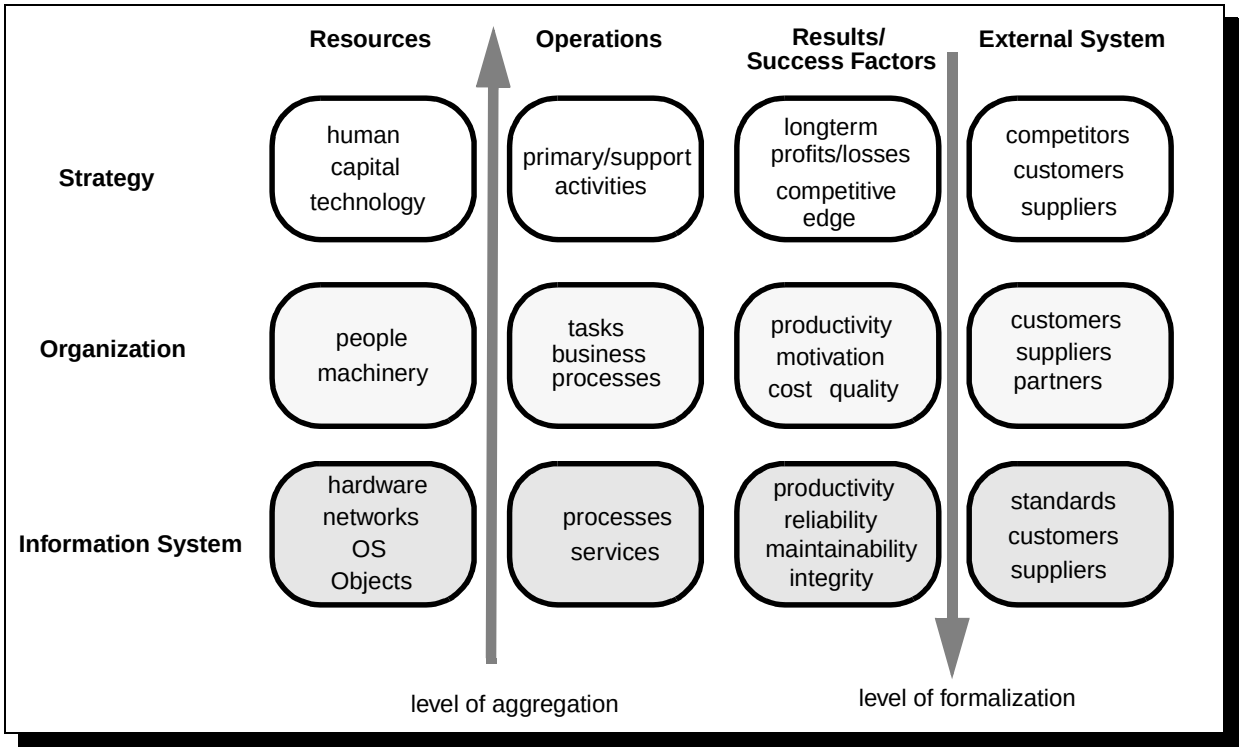

Figure 5. Selected concepts of different views and foci

Procedure models are integrated with an object model in two ways. First they refer to the objects they use, second the management of an office procedure is done by objects that are specified within the object model themselves. Fig. 4 shows how an object model and office procedure models could be represented within an enterprise-wide repository.

2.4 Integrating the Views

While it might be an intriguing vision to deduct the model of the information system from the organizational model and the organizational model from the strategic model, we did not even try to accomplish such a level of integration. This is mainly for two reasons. First: there is hardly general knowledge of how to deduct the organizational concept best suited to accomplish a strategic goal (which is also the case for the relationship between organizational model and information system model). Second: usually a strategy cannot be developed independently from organizational options which in turn are influenced by information technology as well. For those reasons MEMO allows one to explicitly link concepts on different levels. Direct tight coupling is the case, if a concept of a certain view directly corresponds to a concept of another view. For instance: The concept "Customer" within the organizational view corresponds to the class "Customer" within the information system view - although the representations may vary in detail.

The other extreme would be indirect weak coupling. For instance: The goal "customer satisfaction" on the strategic level could be first linked to a model of the firm's order processing on the organizational level. From there one could establish a link to objects within the information system view that provide services compatible with certain EDI-standards.

3 MEMO-Center: The Design Environment

Designing enterprise models according to the proposed methodology can hardly be accomplished without appropriate tools: A complex model requires support for browsing and searching. Because of multiple integrity constraints maintenance that is solely done manually jeopardizes a model's consistency to an unacceptable extent. Finally it is impossible to do without tools when prototyping is to be accomplished. For these reasons we developed an environment based on the

MEMO framework. It was implemented using Smalltalk-80 within the Objectworks® 4.0 environment on Sun4-workstations. High productivity could be achieved by using additional class libraries (see [Frank 92] for more details). The current version runs under Objectworks® 4.1 and VisualWorks® - on all platforms the appropriate Parc Place Smalltalk machine is available for.

The environment consists of three main tools: The *Value Chain Designer* (VCD) serves to design and analyze models of a firm's value chain. The *Object Model Designer* (OMD) supports the specification of object models. It provides various features to search for certain elements of an object model and to browse through the model. The *Workflow Designer* (WFD) allows for conveniently modelling business processes. It provides functions for simulation, fast prototyping and organizational analysis. The three tools are tightly integrated. The integration on the conceptual level has already been characterized above. System integration is accomplished by locating the tools within one Smalltalk image. The concepts managed by the tools can be annotated by using an integrated hypertext system.

The VCD provides a browser through the different elements of a value chain. The description of prototypical instances (like "human resource") is supported by predefined value sets (like "high", "average", "low") which can be modified interactively. This is the case, too, for relationships between activities. They can be characterized using an extensible list of identifiers (like "supports", "supported by", "contains", etc.). In addition to a textual representation relationships can be visualized in a graphical way. Items managed within the VCD can be associated to related items within the other tools. For instance: "activity uses business process". The VCD controls referential integrity of a value chain. Furthermore it provides dialogs to guide with analyzing and re-designing a value chain.

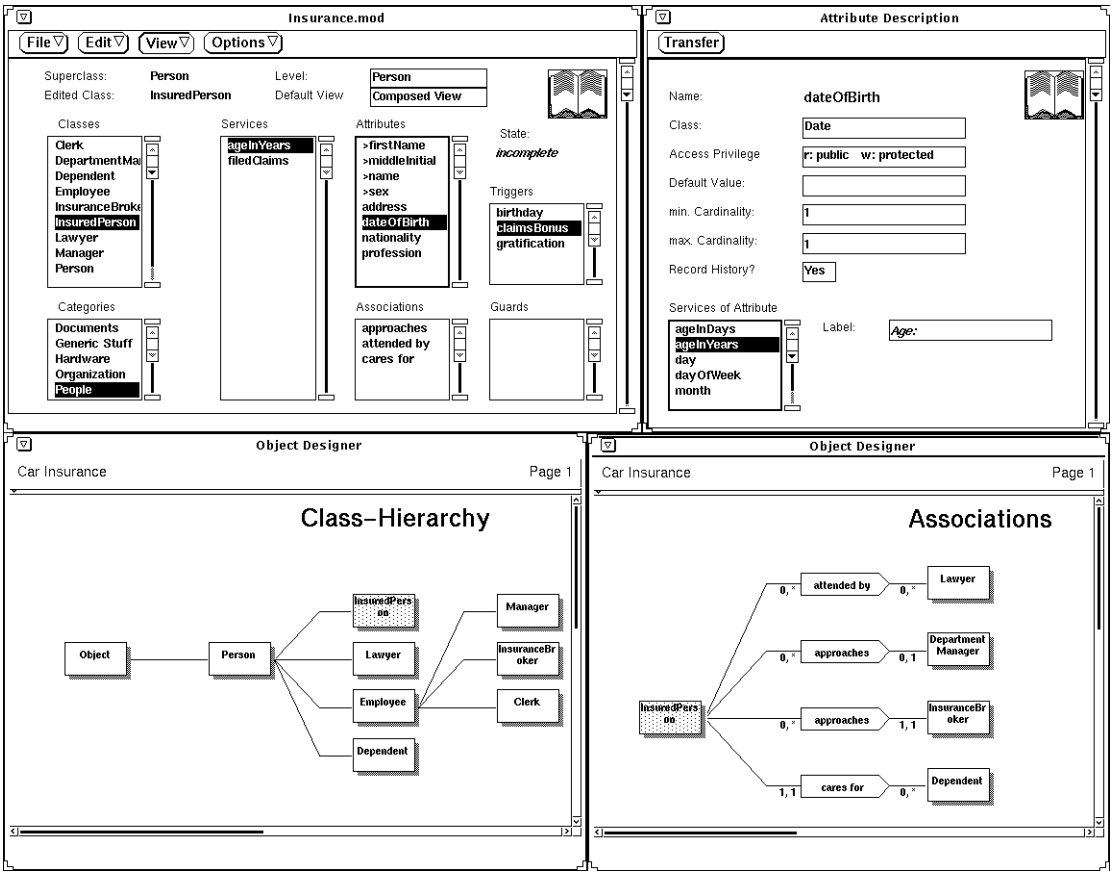

Figure 6. User-interface of the Object Model Designer

According to the meta model described above the OMD offers a number of different levels of abstraction. On the highest level class names are grouped into categories. On the next level a class can be assigned a list of attribute names, service names, guard names, and trigger names. Selecting an attribute, service, etc. causes the corresponding window to pop to the front. It allows for a more detailed description. Fig. 6 shows the window that contains the template for specifying an attribute - 'dateOfBirth' in the given example. Its class is 'Date'. The services which are provided by this class are shown in a listbox. If one of these services is needed for the class that is currently selected (which is 'InsuredPerson' in our example) it can be pasted to the services-listbox of this class - a convenient way to accomplish reusability. The OMD will then establish a reference to this service.

In order to foster system integrity it is not possible to type in a class name directly to characterize an attribute, a superclass, or a parameter. Instead it is required that the dictionary that contains all class names is updated first. Then the name can be pasted to the corresponding field. The OMD controls a number of integrity constraints. It prevents the user from deleting elements which are referenced by other elements, from assigning superclasses or classes of attributes in an inconsistent way, etc. The graphical representation of the class hierarchy can be used as an additional browser. The icons are mouse sensitive and cause the focus to shift to the selected class.

The main focus of the WFD is on business processes - both for analysis (that is mainly the organizational view) and design (organizational and information system view). On the highest level of abstraction the user can draw a business process picking from predefined icons within a specialized editor. When the procedure is described on this level (see fig. 7), the user can 'zoom' into it by selecting an icon - either a subtask or a document state. Within the example shown in fig. 7 the subtask 'Verification of substantial matter' is selected. Within the window titled 'Activity' it can be characterized by assigning execution times, an organizational unit, an organizational position, and possible exceptions. Furthermore the classes (like 'InsuredPerson', 'Policy', etc.), forms, and files which are needed as well as the involved roles can be listed in this window.

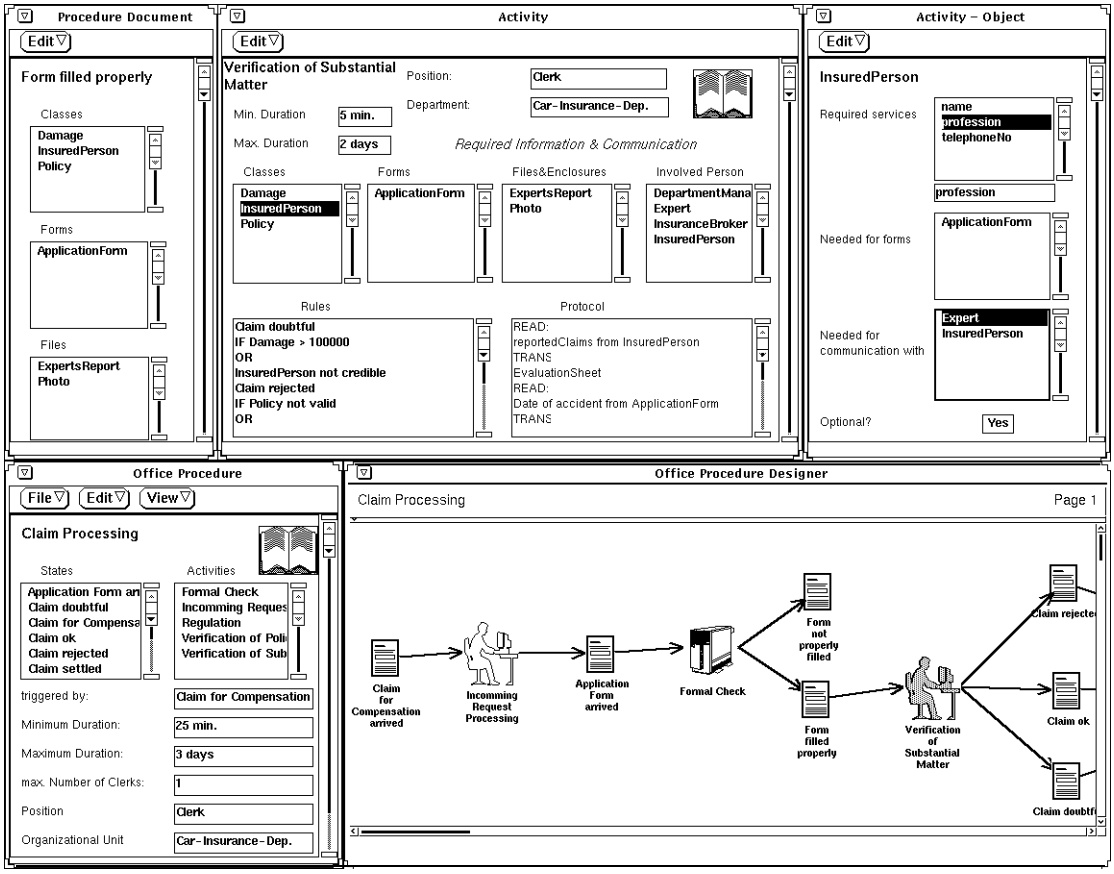

Figure 7. User-interface of the Workflow Designer

Selecting an item causes a window with an appropriate template to pop to the front. Within the example shown in fig. 7 this is the window in top right position. It allows to pick the required services from the selected class. For each service - or the object it delivers respectively - it can be specified what it is needed for: either for a form or for communicating with an involved person. The example shows that the service 'profession' is needed for the 'ApplicationForm' as well as for communicating with the 'InsuredPerson' and the 'Expert'.

Other templates exist to specify the relevant information within forms (fields together with an informal description of their purpose) or files (physical location, subject of interest within the file) and as well as the communication with involved persons (channel, subject). These specifications are used to generate a protocol which is shown in the right bottom corner of the 'Activity' window in fig. 7. Another text-widget within this window presents a template that can be filled to describe decision rules relevant for the focussed activity. A selected document state is specified in a similar way. First the classes, forms, and files which are involved in this state are assigned to it. Then these items have to be characterized in a more detailed way. The state of an object (which is represented by the name of its class) has to be defined by naming a boolean service that checks this state. That requires one to enhance the class with this service by switching to the OMD. For instance: An object of class 'Policy' is required to be in state 'valid'. That requires a service like 'isValid' to be specified for this class. For every form it has to be defined which fields have to be filled in. A file is characterized by its physical location and optionally the means of transportation to get it to the clerk. In order to support the user and to avoid inconsistencies the classes, forms and files assigned to a document state are pasted to the subtask that is triggered by this state. The WFD can generate a prototypical user-interface for every activity - provided the default views have been assigned to the corresponding classes in the object model (see above). The widgets placed within such an interface may be rearranged interactively. In order to use the simulation features built into the WFD it is necessary to first assign probabilities (using percentage values) to the states produced by every subtask. Furthermore it is required to type in the workload as number of procedures in a time period. After that the simulation can be started. Animation is accomplished by dynamically marking the subtasks which are active. If the simulation reveals any chances to improve the organization of the procedure the net can be rearranged interactively. Fig. 8 shows a snapshot of a simulation where the subtask 'Verification of substantial matter' has been expanded since it was found to be a bottleneck.

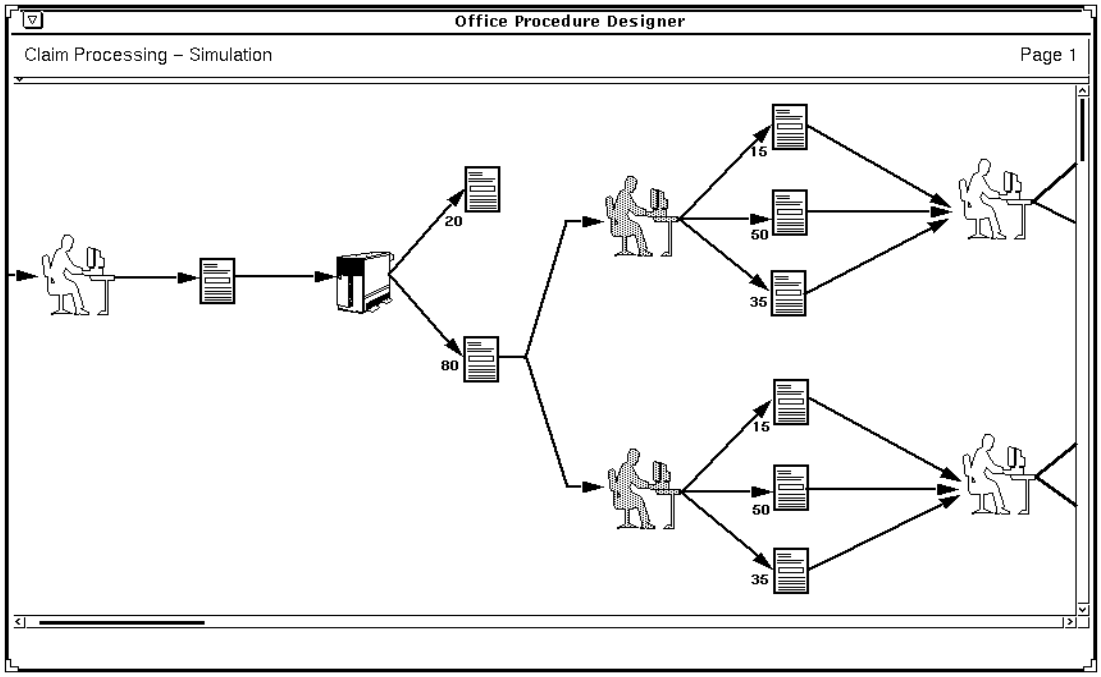

Figure 8. Snapshot of an animated simulation

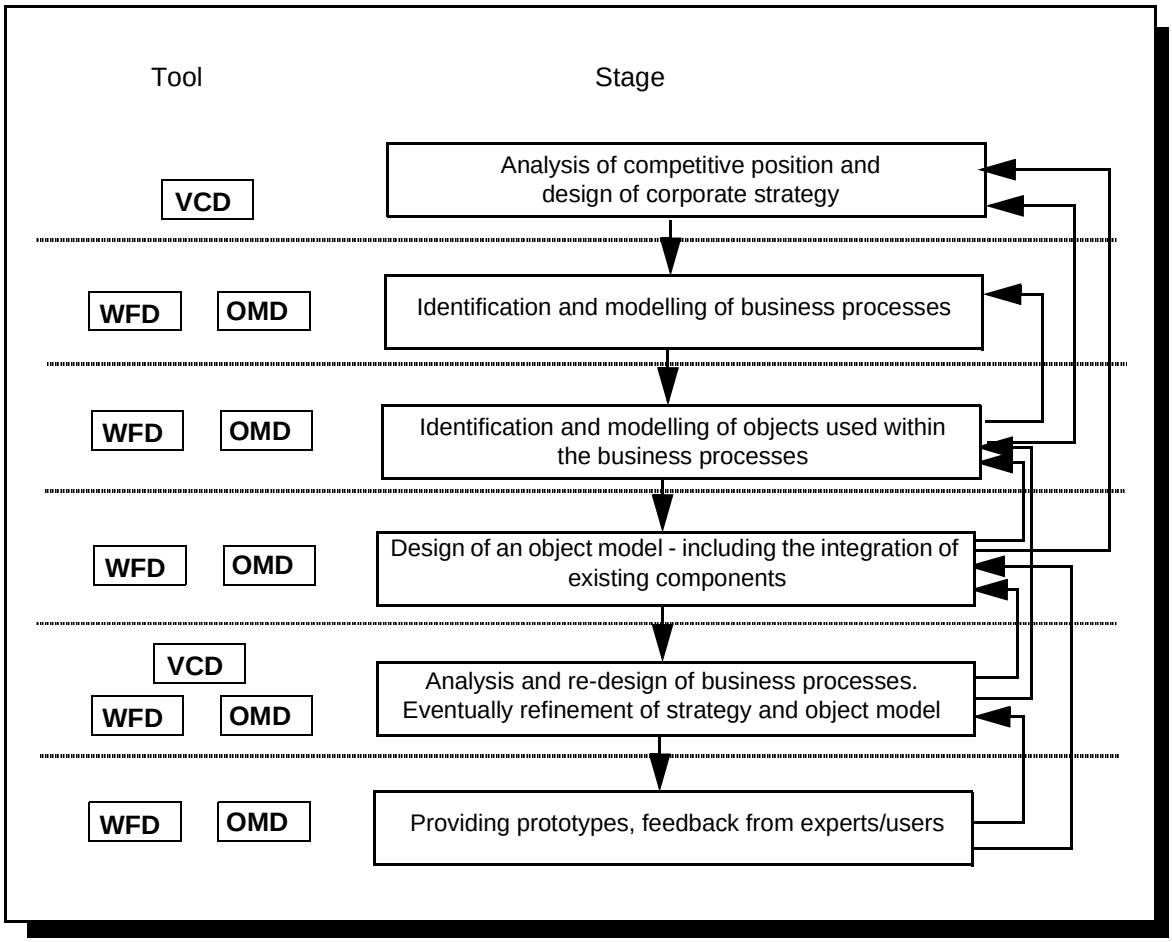

Figure 9. The usage of MEMO Center during important development stages (this is a simplified model!) and selected feedbacks between stages.

The WFD also allows for generating a report on detected media frictions and to graphically visualize communication relationships. Fig. 9 shows which tools are primarily used in the various stages of enterprise modelling.

4 Conclusions

Different from most other methodologies for object-oriented analysis and design MEMO not only focuses on the information system in itself but also on relevant strategic and organizational issues. Thereby it fosters a smooth integration of an information system with business processes and corporate strategies. Our experience with modelling office domains within insurance companies indicates that the proposed representations offer illustrative abstractions of an enterprise. This is especially the case for the representation of business processes. Both system analysts and domain experts intuitively understood the graphical notation. Thereby it proved to be a valuable medium for starting knowledge acquisition or object modelling respectively. We found that asking about strategies and processes is not only a prerequisite for organizational change. It furthermore helps to identify objects and specify their semantics. MEMO does - of course - not guarantee to optimize the organization of work. It can help however to reduce complexity by providing an illustrative representation of important aspects, by detecting certain types of organizational misconception, and - last but not least - promoting a sophisticated object-oriented architecture of a company's information system.

In the current version MEMO Center is restricted to one Smalltalk image. Therefore the prototypical workflow systems only simulate distribution. The objects used within the prototypes are either implemented directly in Smalltalk or refer to C-functions which are linked with the virtual machine. Currently we are working on extensions that will allow

for partial generation of distributed workflow management systems. For this purpose we use HP's "Distributed Smalltalk". It includes a CORBA implementation according to the OMG guidelines. MEMO Center will then allow for integrating any existing component that offers an IDL-interface. In order to allow for an automatic transformation of an object model designed according to MEMO into the OMG object model MEMO Center will be enhanced with means to distinguish between inheritance and subtyping. A text book with a comprehensive documentation of the methodology and the environment will be published by the end of 1994.

References

Booch, G.: Object-oriented design with applications. Redwood 1990

Coad, P.; Yourdon, E.: Object Oriented Design. Englewood Cliffs, NJ 1991

Davenport, T.; Short, J.E.: The New Industrial Engineering: Information Technology and Business Process Redesign. In: Sloan Management Review, Summer 1990, pp. 11-27

Dennis, A.R.; Hayes, G.S.; Daniels, R.M.: Re-Engineering Business Process Modeling. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 27th Hawaii International Conference on System Sciences, Vol. IV. Los Alamitos, Ca. 1994, pp. 245-253

ESPRIT Consortium AMICE: CIM-OSA AD 1.0 Architecture Description. Brussels 1991

Frank, U.; Klein, S.: Three integrated tools for designing and prototyping object-oriented enterprise models. GMD Research Report No. 689, Sankt Augustin 1992

Frank, U.: A Comparison of two Outstanding Methodologies for Object-Oriented Design. GMD Research Report No. 779, Sankt Augustin 1993

Frank, U.: An Object-Oriented Methodology for Analyzing, Designing and Prototyping Office Procedures. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 27th Hawaii International Conference on System Sciences, Vol. IV. Los Alamitos, Ca. 1994, pp. 663-672

Hammer, M.; Champy, J.: Reengineering the Corporation. New York 1993

Hassey, D.E.: Glossary of Management Techniques. In: International Review of Strategic Management. Vol. 3, 1992, pp. 47-75

Hong, S.; Goor, G.: A Formal Approach to the Comparison of Object-Oriented Analysis and Design Methodologies. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 26th International Hawaii International Conferenc on System Sciences, Vol. III., Los Alamitos 1993, pp. 689-698

Jacobson, I.; Christerson, M; Jonsson, P; Overgaard, G.: Object-Oriented Engineering. A Use Case Driven Approach. Reading, Mass. 1992

Katz, R.L.: Business/enterprise modelling. In: IBM Systems Journal, Vol. 29, No. 4, 1990, pp. 509-525

Keen, P.G.W.: Shaping the Future: Business Design through Information Technology. Cambridge 1991

Meyer, B.: Object-Oriented Software Construction. Prentice Hall 1988

Monarchi, D.E.; Puhr, G.: A Research Typology for Object-Oriented Analysis and Design. In: Communications of the ACM, Vol. 35, No. 9, 1992, pp. 35-47

Nierstrasz, O.; Dami, L.; De Mey, V.; Stadelmann, M.; Tsichritzis, D.; Vitek, J.: Visual Scripting: Towards Interactive Construction of Object-Oriented. In: Tsichritzis, D. C. (Hg.): Object Management. Geneva 1990, pp. 315-331

Peters, L.; Schultz, R.: The Application of Petri-Nets in Object-Oriented Enterprise Simulations. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 26th International Hawaii International Conferenc on System Sciences. Vol. III, Los Alamitos 1993, pp. 390-398

Porter, M.E.: Competitive Advantage. New York 1985

Porter, M.E.; Millar, V.E.: How Information Gives You Competitive Advantage. In: Harvard Business Review, July/August 1985, pp. 149-160

Rumbaugh et.al.: Object-Oriented Modelling and Design. Hemel-Hempstead 1990

Scott Morton, M.S.: Strategy formulation methodologies. Cambridge, Mass. 1986

Sowa, J.F.; Zachman, J.A.: Extending and formalizing the framework for information systems architecture. In: IBM Systems Journal, Vol. 31, No. 3, 1992, pp. 590-616

Talwar, R.: Business Re-engineering - a Strategy-driven approach. In: Long Range Planning, Vol. 26, No. 6, 1993, pp. 22-40

Wiseman, C.: Strategy and Computers: Information Systems as Competitive Weapons. Homewood, Ill. 1985

Zachman, J.A.: A framework for information systems architecture. In: IBM Systems Journal, Vol. 26, No. 3, 1987, pp. 277-293

MEMO: A Tool Supported Methodology for Analyzing and (Re-) Designing Business Information Systems

Ulrich Frank
Institut für Wirtschaftsinformatik, Universität Koblenz
Rheinau 1, 56075 Koblenz, Germany
Email: ulrich.frank@informatik.uni-koblenz.de

Abstract

The paper presents a conceptual framework as well as a design environment to develop object-oriented enterprise models. It helps to coordinate the design of a business information system with the modelling of corporate strategies and organizational (re-) design. For this purpose the framework introduces concepts for illustratively modelling and integrating three main views on the enterprise. The views focus on corporate strategy, business (process) organization, and information system design. Thereby the methodology contributes to overcome the communication barriers that commonly exist between the different views. The integration of the various stages of system development is promoted by linking concepts of different views. Since business information systems are usually not built from scratch the methodology provides means for integrating existing software or data structures. The development environment that is based on the framework allows the user to navigate through the views of an enterprise model on various levels of detail. It maintains a model's integrity and allows for simulation and fast prototyping.

1 The Need for Models of the Enterprise

Designing, implementing and maintaining business information systems faces a number of challenges. On the software level it is - among others - desirable to support integration on a high level of semantics, to foster integrity and to allow for convenient reuse of existing artifacts. Penetrating a company with information technology allows for or may require new ways to organize the business. During the last years different aspects of this problem have been addressed by a variety of approaches. They are related to notions like "paperless office", "lean management" or "business (process) re-engineering" ([Hammer 93], [Talwar 93]). Reorganizing a firm or parts of it should be compatible with its long term goals. On the other hand the range of strategic options is more or less influenced by the available information technology - no matter whether you regard it as a "strategic weapon" ([Porter/Millar 85], [Wiseman 85]) or as a constraint.

The various approaches to deal with those different challenges usually have one characteristic in common: they are based on models - either of the whole enterprise or of parts of it. It has been well accepted for long that conceptual models are crucial for designing and implementing integrated information systems. While those models used to be data-oriented, object-oriented design methodologies are currently gaining more and more attention. Methodologies to support the analyst with business re-engineering are usually based on models as well. They are mainly focused on business processes ([Davenport 90], [Dennis 94]). Furthermore there is a wide range of models to help with analyzing and shaping a firm's strategy. They usually stress a more abstract view with highly aggregated data (for an overview see [Hassey 92] and [Scott Morton 86]). [Keen 91] explicitly promotes the use of information technology to develop corporate strategies. All these models have been introduced to reduce the complexity of strategic planning in order to help the analyst to concentrate on the essentials - and to communicate them to others who should be involved.

It is no surprise that strategic, organizational and information system models are usually based on

Published in: Ege, R.; Singh, M.; Meyer, B. (Eds.): Technology of Object-Oriented Languages ans Systems. Englewood Cliffs 1994, S. 367-380

different concepts. However, treating these different aspects independently bares the risk of redundant work and friction - a well known phenomenon for long. In order to allow for a more synergetic approach a number of authors ([Zachman 87], [ESPRIT 91], [Katz 90], [Sowa 92], [Peters 93]) have suggested enriched modelling frameworks - often named "enterprise modelling" (a term which is however not used in a unique way). Such methodologies differentiate between a number of views on the enterprise and intend to capture the relationships between these views. Studying them however shows that they either remain on a rather abstract level (for instance: [Sowa 92], [Zachman 87]) or lack sophisticated concepts - both from a managerial and a software engineering point of view (they are usually rather data- than object-oriented).

This paper presents a methodology as well as a set of integrated tools for developing object-oriented enterprise models that cover the main aspects of analyzing, designing, implementing and maintaining business information systems. It puts special emphasis on the following aspects:

- Different ways to conceptualize an enterprise in an illustrative way, called perspectives or views, are supported.
- An object-oriented design methodology that is specially suited for modelling business information systems.
- Concepts to support the integration of existing components, like applications or data structures, are provided.
- A systematic approach to analyze a firm's competitive position and to generate strategic options is included.
- There is support for (re-) designing information intensive business processes.
- An enterprise model can be very complex. However, for economic reasons there may be the need to be less ambitious - both in extent and detail. Therefore the methodology can be adapted to the constraints of a specific project.
- The development environment provides various ways of browsing through an enterprise model and maintaining its integrity. It also allows for fast prototyping and simulation.

2 Multi Purpose Enterprise Modelling

Inspired by the vision of highly integrated business information systems and additionally motivated by the various problems with existing systems a group of researchers at GMD started in 1990 to develop scenarios of how to deal with this complex challenge. Very soon it became obvious that it was a key issue to design multi-view models of the enterprise - mainly inspired by the framework presented in [Zachman 87]. We decided on three main views. The *strategic view* was to model the enterprise in a way that fits the perception common to senior executives. It should allow for illustratively describing a firm's competitive position and its strategic options. The organizational view was to be focussed on a company's organizational structure and the way its tasks/processes are performed. The *information system view* should provide the basic modelling constructs (in other words: the meta model for modelling) together with a framework to analyze existing systems and (re-) design and maintain them.

The methodology that has been developed in the following years - called *Multi Purpose Enterprise Modelling* (MEMO) - provides the analyst with various concepts to describe each view. The concepts are presented on different levels of detail and precision: a conceptual framework, guidelines to develop scenarios (mainly to describe tasks/processes), heuristics that help to identify important aspects, structured questionnaires to guide interviewing, and templates to gather formal aspects.

There are three dimensions to structure each of the three main views. *Stage* serves to describe a particular model's position within the continuum between analysis, design and maintenance. Each view is described in terms of the required resources, the operations or processes, the results which are produced, and the relevant features of external systems. This dimension is called *focus*. Usually it is desirable to model a view on a high level of abstraction. However, for certain types of analysis you may need to consider the state of instances. Within this dimension, called *aggregation*, MEMO allows the analyst to use concepts (like classes), particular instances and so called prototypical instances which represent the average state of a relevant set of instances (like the salary of the "average clerk").

2.1 The Strategic View

A methodology to guide analysis and design of corporate strategies should be suited to represent the required concepts in a way that is familiar to those who are commonly dealing with strategic planning - like senior executives. It should be based on generalized assumptions and should also allow to be con-

figured/specialized to a particular firm's needs. Among the wide range of methodologies (see [Scott Morton 86]) we found Porter's *value chain approach* to be most appropriate. Due to the complexity and contingency of the domain the methodology does not offer a precise guideline for developing strategies. However, Porter's approach provides the analyst with familiar concepts which support him to develop a solution of his problem in a systematic way. The value chain concept has gained a high degree of acceptance with many companies and consultants.

On the top level Porter models an enterprise as a system of "activities" which form a "value chain". "The value chain disaggregates a firm into its strategically relevant activities in order to understand the behavior of costs and the existing and potential sources of differentiation. A firm gains competitive advantage by performing these strategically important activities more cheaply or better than its competitors." ([Porter 85], p. 33) *Primary activities* are directly involved in the process to produce the products or services that are offered to a firm's customers. They include inbound logistics, operations, outbound logistics, marketing and sales and service. *Support activities* (firm infrastructure, human resource management, technology development and procurement) serve to support primary activities. An activity is an abstraction of actual business processes. Therefore it does not have to directly correspond to a specific process (for a detailed description see [Porter 85]).

In order to adapt Porter's approach to the MEMO framework we applied the three dimensions stage, focus and abstraction to it. There are two outstanding stages - with an arbitrary number of intermediate stages: the current competitive position and the position aimed at with the future strategy. Resources - like various types of capital or human resource - are used to perform activities. They are described on a high level of aggregation with emphasis on cost and quality issues. The outcome which is produced by an activity is modelled on a similar level - with aggregated figures for quality and price. Processes are basically described as a chain of activities with each activity characterized by the resources it consumes and its outcome. This results in a value chain being modelled as a graph of activities, resources, and outcomes with special emphasis on the interrelationships.

The relevant external world consists of more or less detailed value chains of competitors, suppliers and customers. In order to support the shaping of a firm's value chain the analyst is provided with heuristics like "How can the activity be performed differently or even eliminated?" or "How can a group of linked value activities be reordered or regrouped?" ([Porter 85], p. 110).

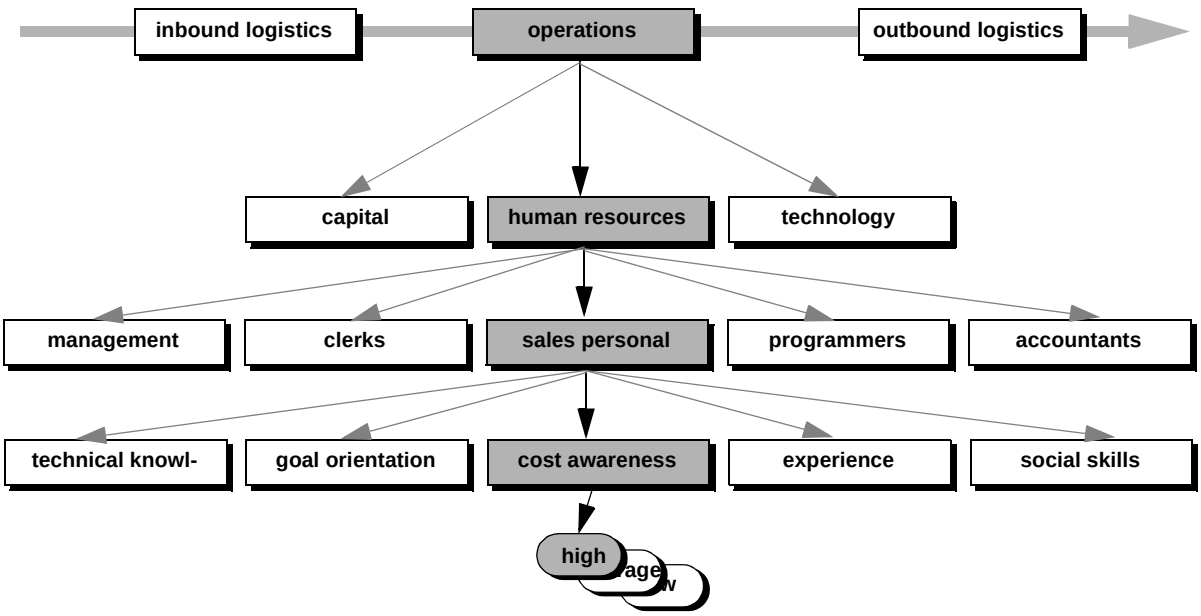

Figure 1. An example for the analysis of resources within the strategic view

Additionally Porter offers a top down approach that suggests to start with one out of a list of “generic strategies” and stepwise specialize it into a new value chain.

It is obvious that both analyzing and designing a value chain require special attention to the interrelationships, which Porter calls “linkages” ([Porter 85], p. 50): "Managing linkages thus is a more complex organizational task than managing value activities in themselves." While the methodology certainly does not allow for automatically generating a firm’s value chain, it can well be mapped to a specialized browser (see 3). The concepts used to describe the strategic view are represented in an object-oriented way according to the meta model characterized in 2.3.

2.2 The Organizational View

On the organizational level an enterprise model comprises a model of the organizational structure and models of business processes - both for analyzing the given state and for designing future states.

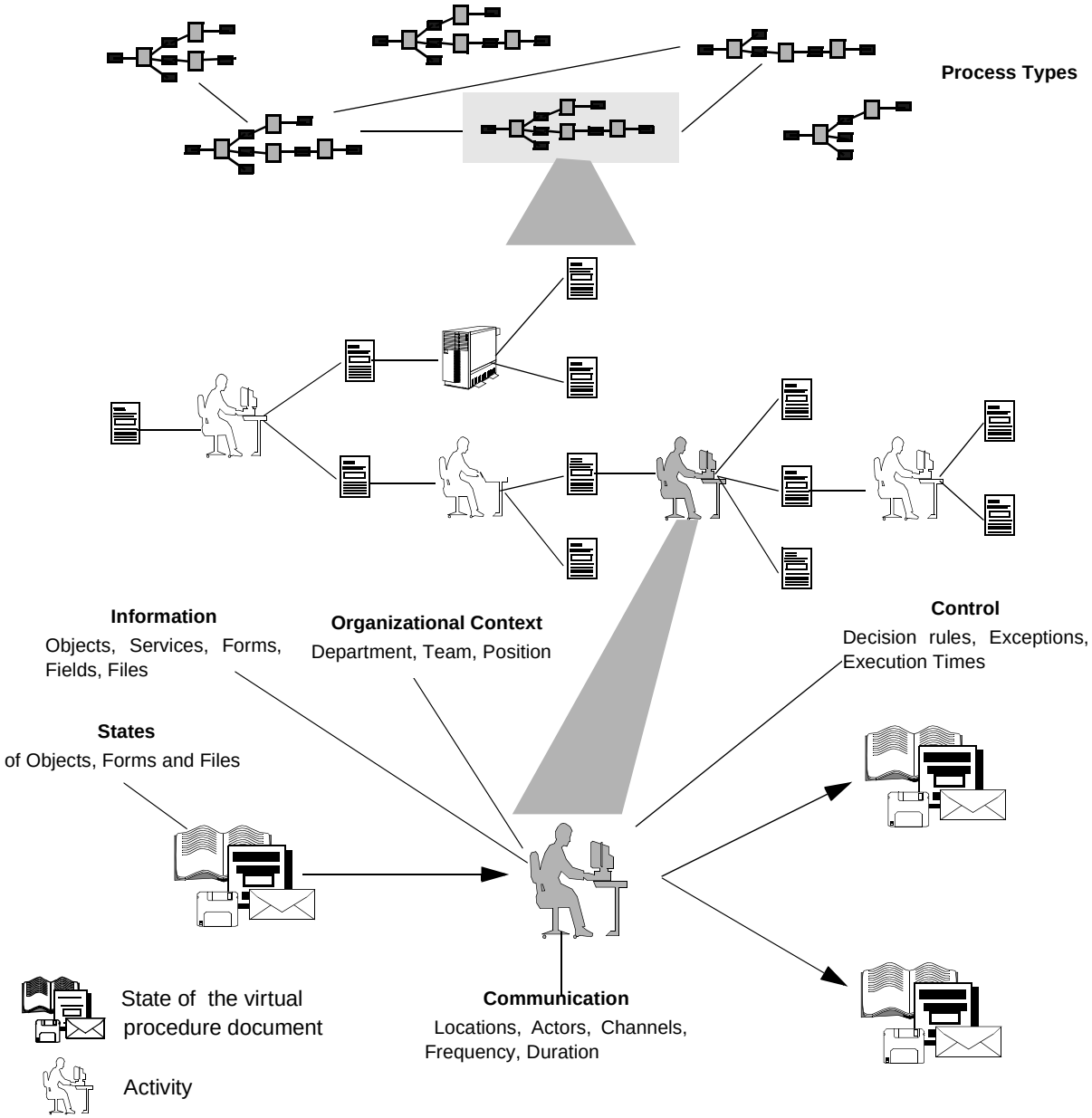

Figure 2. Conceptualization of business processes

The organizational structure is represented by organizational units (such as divisions, departments, groups, positions etc.) and their relationships ("reports to", "is part of" etc.). In addition to modelling a particular structure MEMO encourages the analyst to identify general rules for the division of labor and its coordination. Organizational resources are usually modelled as prototypical instances. Examples for resources on this level are people, buildings, furniture, machinery etc. Computer hardware may also be considered as resource within the organizational view. It is described by referring to the information system view (see below). Features which may be described within prototypical instances include various types of cost, availability, capacity etc. The number of resources and the detail of their description is subject to individual configuration. The external world is represented by relevant roles (like lawyer, consultant, customer, etc.) and services.

A business process is modelled as an ordered graph of subprocesses, where a subprocess in itself can be decomposed into other subprocesses. MEMO's emphasis is on office procedures. Therefore the "material" that is operated on is information. Information is grouped into three categories: objects which reside on the computer based information system, forms, and files. Information that is located in the information system is described by referring to the object model that is part of the information system view (see below). Forms have a formal structure, that is they contain fields, have well defined states (like 'complete', 'incomplete', 'consistent'), and a set of constraints defines the permissible operations. Their content may be changed within a subprocess. The term file is used to summarize documented information that is read-only - like office files, letters or journals. The information a business process operates on is gathered in a "virtual procedure document" that extends the traditional notion of a document in two respects: It may contain references to information that is located somewhere else. Furthermore it can be duplicated and thereby processed in parallel. Each subprocess is triggered by a certain state of the virtual procedure document and produces one or more new states.

Each process is assigned an *organizational context* by referring to the organizational units which are responsible for its execution. By default the organizational context is propagated to the subprocesses where it may be overwritten. Furthermore each process can be characterized by *business rules* like: "All subprocesses should be managed by the same person." The subprocesses can be described in a very detailed way - depending on the effort that is to be spent for analysis. Among the more important features are: minimum and maximum execution time, decision rules, required resources (for instance: copy machine, printer), exceptions, people (roles) needed to communicate with, communication media etc. In order to support the re-design of business processes MEMO provides the analyst with means to analyze existing processes and design heuristics. The model of a business process allows the analyst to detect media frictions (for instance: information that originally resides in the information system and is then being transferred to a form), to identify bottlenecks, or to draw communication nets. By adding statistical data on workload, capacity and probabilities of the subprocesses' possible outcome the model can be utilized to perform simulations (see fig. 8).

By focussing on processes the organizational model serves different purposes: It helps to redesign the business (if necessary), it helps to define the communication technology required to improve efficiency, and - last but not least - it supports to identify the objects/classes required to perform the relevant tasks. As with the strategic view the concepts of the organizational view are described with the object-oriented meta model (see below) in mind. Therefore they can be represented as an object model which fosters their mapping on a tool.

2.3 The Information System View

The information system view includes both a model of the existing system and the system that is to be designed. They are separated into an object-oriented conceptual model and a model of information system resources (like hardware, networks, operating systems). An object-oriented conceptual model includes an object model and other models to express dynamic aspects. Among the still increasing number of object-oriented design methodologies (in a survey we did last year we found more than forty approaches) we felt most inspired by the ones proposed by [Booch 90], [Rumbaugh et al. 90], and [Jacobson et al. 92]. However, none of them was satisfactory for our purpose. While Rumbaugh et al.' methodology suffers from being somewhat superficial and not consequently object-oriented, Booch's approach seems to be overloaded by details of various programming languages - which,

in our opinion, should not be part of a general methodology for the design of *conceptual* models. Jacobson et al. are primarily focussing on analysis and put less emphasis on software engineering issues occurring during design. (for a comparison of important methodologies see [Frank 93], [Hong 93], [Monarchi 92]). Furthermore we were not satisfied with the way dynamic aspects are modelled within these approaches. State transition diagrams are often suggested to capture dynamic aspects. At first sight such diagrams seem to be appropriate for modelling automated business processes, since they allow the analyst to describe events and corresponding state changes. They are however restricted to events and state transitions which may occur during the lifetime of objects of one class. Within an office procedure however one usually needs objects of more than one class.

2.3.1 Conceptualizing Object Models

While from a (re-)using programmer´s point of view it is sufficient to describe an object solely by the services it provides analysis and design require a more detailed view. Within an object model one defines classes and associations between classes or between objects respectively. Like in most other methodologies the outstanding features of a class are attributes and services. An attribute is regarded as an object that is encapsulated within an object. We do not allow attributes - like [Coad 91] - to only hold references to external objects that have an existence of there own in the object space. An attributes semantics is primarily defined by its class. Furthermore a cardinality (using min-max notation) may be assigned. Assigning a default value allows for generating appropriate initialization operations.

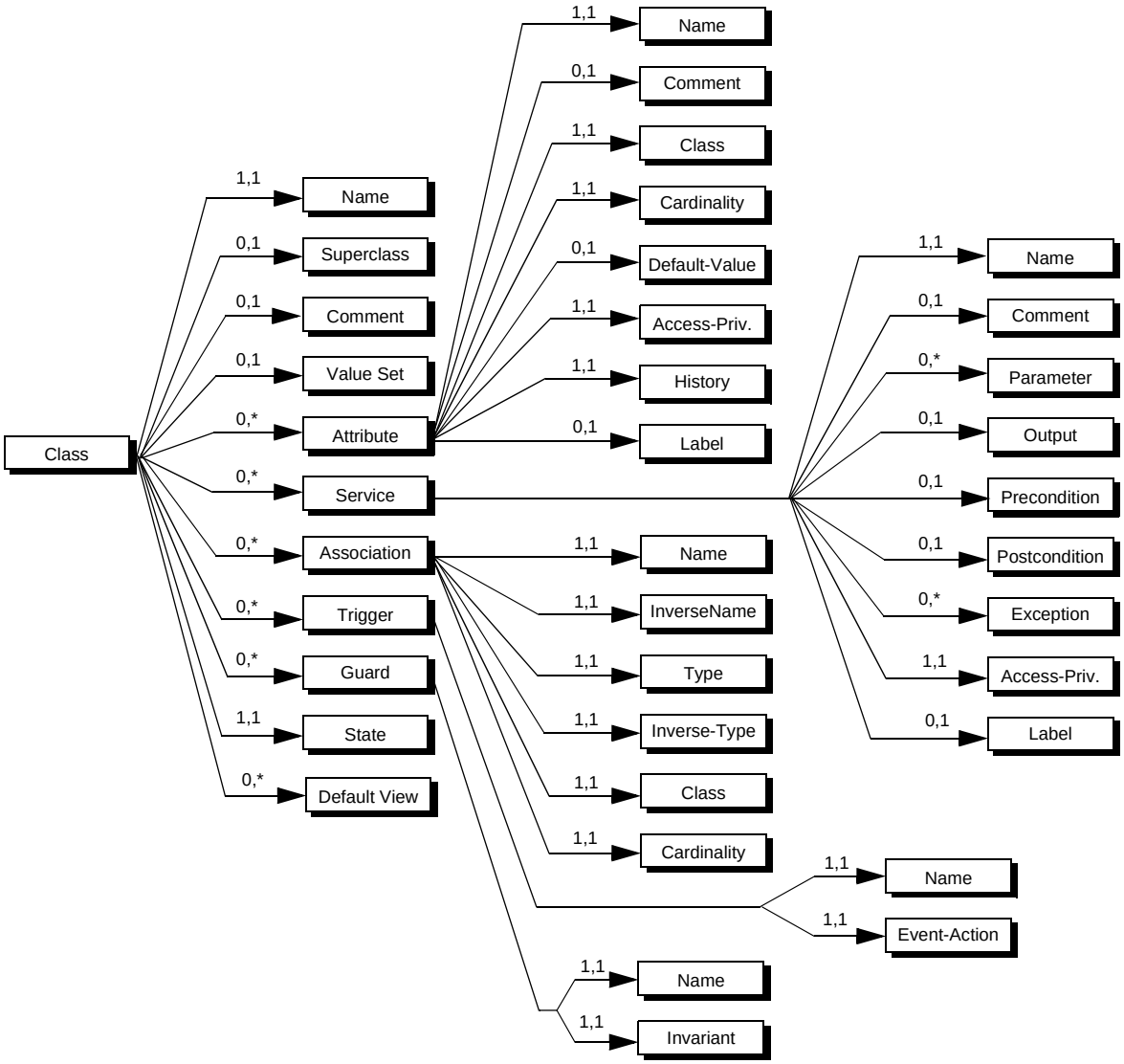

Figure 3. Part of the meta model for conceptualizing object models

Each attribute is also characterized by a history-flag. Setting it to true means that every update should be recorded somehow.

In order to allow for generating prototypical user-interfaces it is possible to assign a default-view to each class. A default view is either a widget or a collection of widgets. One can also define a label that is to be presented with the default view. Additionally features like size and font may be specified. This approach is a first attempt to deal with the complexity of user interaction. It cannot be completely satisfactory: the way a value of a certain class is presented to the user often is not unique but varies with the context of interaction.

In order to specify a service the designer may describe a list of input-parameters (which can be empty) where each parameter is characterized by its class and its name. If a service returns a result it has

A postcondition has to be fulfilled after the service has terminated. Similar to a precondition it can be defined by referring to an object state or to a state of the object that is returned by the service. Each service can be assigned a set of exceptions (like media errors) which should be named in a unified way for a whole object model. Thereby exception handling can be defined for all involved objects in the same way.

During the life time of an object there may be certain events and rules which go beyond the scope of a single service. For this purpose we introduce triggers and guards. A trigger can be generally defined as a tupel consisting of an event and an action. The event is specified by a condition that in turn is defined by referring to attribute states or states of objects which are returned by a service. The action is defined by the service that has to be executed when the event occurs.

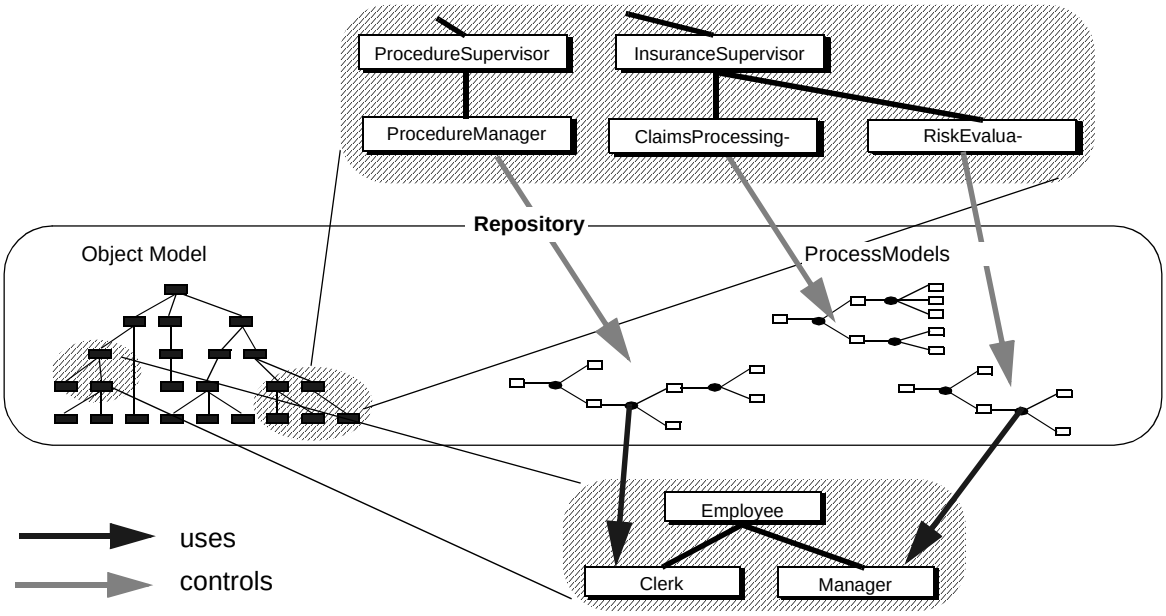

Figure 4. Integration of object model and office procedure models within an enterprise-wide repository

to be exactly one object. So it is sufficient to define the class of this object. It is important to note that such an object may be a composed object (like an array, a container etc.) that contains many other objects. A precondition in general specifies conditions "under which a routine will function properly" ([Meyer 88], p. 114). In our model it can be defined by referring to object or parameter states. For instance: For a service that requires an object of class ’Person’ as a parameter it may be necessary that the service ’sex’ delivers the state ’male’.

To give an example for a trigger: Whenever an object of class ’customerAccount’ has a balance less than x, the object should execute a service that is suited to notify somebody who is managing the account. A guard is a condition that has to be fulfilled during the lifetime of any object of a particular class (similar to what [Meyer 88], p. 124) calls "class invariant"). For instance: The value of attribute ’retailPrice’ within objects of class ’Product’ should never be lower than the value of attribute ’wholesalePrice’.

Every class may have exactly one superclass. Although there are a number of arguments in favor of multiple inheritance we restricted our model to single inheritance. We found that in most cases single inheritance is satisfactory while multiple inheritance increases the complexity of an object model and thereby makes it more difficult to maintain it in a consistent way. On the instance level MEMO distinguishes between two types of associations: interaction and aggregation. However, for a conceptual object model to be illustrative it is desirable to allow for a more detailed differentiation. For this reason each association has to be assigned a domain level identifier. Such identifiers do not include any semantics, they only improve readability of the model and allow for enhanced retrieval capabilities.

In order to allow for smoothly integrating existing elements - like applications or data structures - MEMO suggests that the modeller regards them as objects. Application classes are subclasses of the class "Application" that provides attributes and services to manage information like installation date, version, cost etc. An application class is usually modelled by the set of services it offers to the outside (which is rather poor for the average legacy application) and details about the communication protocol (for instance: OS-call, RPC, UDP or CORBA). Existing data structures, such as relation types of an RDBMS or a document file of a word processor, are represented as dummy classes: They do not encapsulate any attributes, instead they offer services that allow to access an existing element. For instance: class 'WordDocument' represents documents produced by a certain word processor. In order to integrate them with a more sophisticated model one can use an association with the reserved name "corresponds to". For instance: "RelTypeCustomer corresponds to Customer" or "WordDocument corresponds to CompoundDocument". Some existing applications use data that might be shared with other objects. Whenever such a potential source of redundancy is discovered it can be expressed in the model using an association with the reserved name: "should use". For instance: "AccountingSystem should use Customer". To avoid confusion about the implementation state of a model it is important to assign one of four predefined states to each class: "not implemented", "implemented", "dummy", "encapsulated". Fig. 3 gives an overview of the concepts proposed for designing object models. For a more comprehensive documentation see [Frank 92], [Frank 94].

2.3.2 Modelling dynamic aspects

A methodology for modelling dynamic behavior should allow for conveniently expressing temporal and control (in other words: dynamic) aspects. It should also help to avoid inconsistencies, like non terminating cycles, tasks which cannot be reached by any chance, deadlocks, etc. Unlike the object-oriented methodologies mentioned above Peters and Schultz [Peters 93] propose a modelling technique that allows for including objects of more than one class. They use Petri nets where each transition represents a state transition of an object of a particular class. Different transitions within a net may represent objects of different classes. Furthermore they allow - different from traditional concepts - transitions to have an execution time larger than zero. Mapping transitions to operations of an object of a certain class is attractive from a software-engineering point of view since it supports the idea of building procedures by 'glueing' objects together (as it is proposed in [Nierstrasz 90]). Nevertheless such an approach has its deficiencies, too. It is important for a model to allow for direct correspondence to familiar conceptualizations of the relevant domain. Business processes are not necessarily structured in a way that there is always only one object operated on within a subtask.

The approach we have chosen is similar to the one suggested by Peters and Schultz in that we also use semantically enriched Petri nets and allow transitions to have an execution time larger than zero. Our methodology however allows to explicitly assign objects of many different classes to one transition. The information system model of a process is very similar to the model of a business process within the organizational level - and in fact may be deducted from it. The main difference: On the information system level a process is described only by its automated parts. That implies tighter constraints on the specification of the information being processed: only information that is represented in the object model can be regarded.

In order to provide the model with prototyping capabilities it is necessary to enhance it with information about a suitable user-interface. This information can be deducted from those services assigned to a subtask. The widgets needed to interact with the services can be looked up in the object model (see above). Since every suitable class in the object model should have a default view assigned to it, a prototypical user-interface for an activity can be generated.

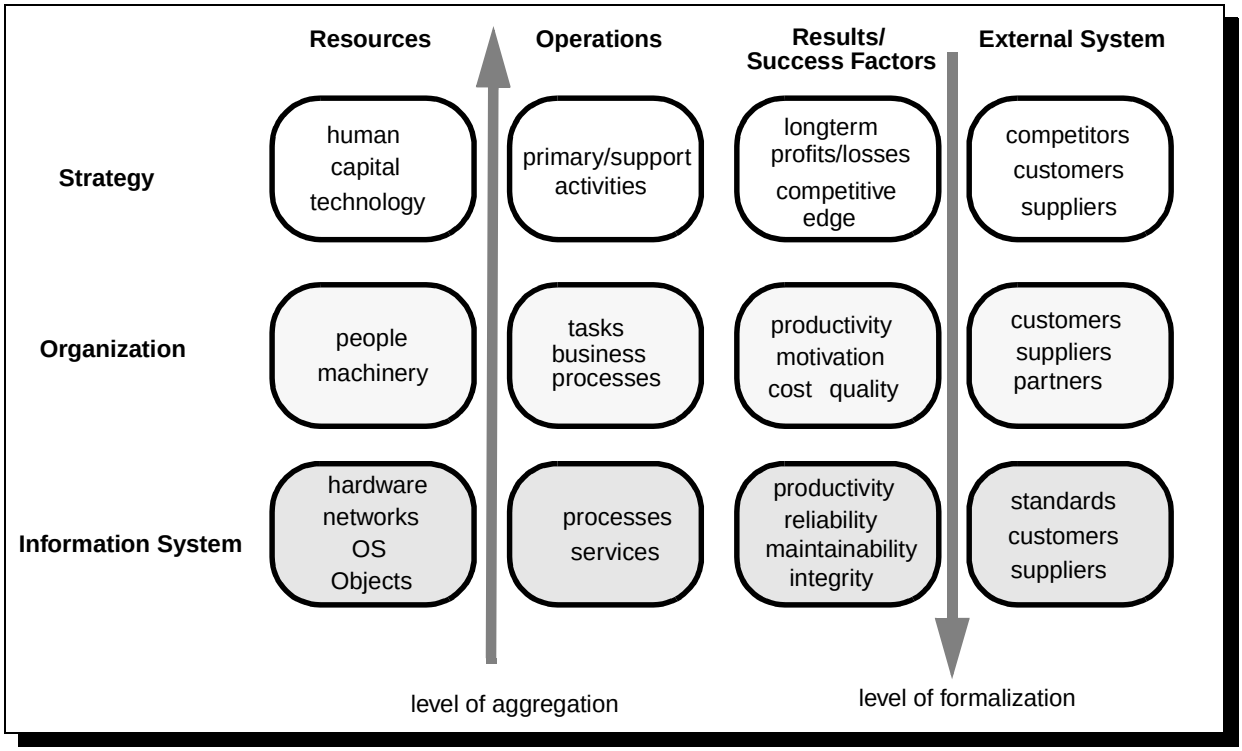

Figure 5. Selected concepts of different views and foci

Procedure models are integrated with an object model in two ways. First they refer to the objects they use, second the management of an office procedure is done by objects that are specified within the object model themselves. Fig. 4 shows how an object model and office procedure models could be represented within an enterprise-wide repository.

2.4 Integrating the Views

While it might be an intriguing vision to deduct the model of the information system from the organizational model and the organizational model from the strategic model, we did not even try to accomplish such a level of integration. This is mainly for two reasons. First: there is hardly general knowledge of how to deduct the organizational concept best suited to accomplish a strategic goal (which is also the case for the relationship between organizational model and information system model). Second: usually a strategy cannot be developed independently from organizational options which in turn are influenced by information technology as well. For those reasons MEMO allows one to explicitly link concepts on different levels. Direct tight coupling is the case, if a concept of a certain view directly corresponds to a concept of another view. For instance: The concept "Customer" within the organizational view corresponds to the class "Customer" within the information system view - although the representations may vary in detail.

The other extreme would be indirect weak coupling. For instance: The goal "customer satisfaction" on the strategic level could be first linked to a model of the firm's order processing on the organizational level. From there one could establish a link to objects within the information system view that provide services compatible with certain EDI-standards.

3 MEMO-Center: The Design Environment

Designing enterprise models according to the proposed methodology can hardly be accomplished without appropriate tools: A complex model requires support for browsing and searching. Because of multiple integrity constraints maintenance that is solely done manually jeopardizes a model's consistency to an unacceptable extent. Finally it is impossible to do without tools when prototyping is to be accomplished. For these reasons we developed an environment based on the

MEMO framework. It was implemented using Smalltalk-80 within the Objectworks® 4.0 environment on Sun4-workstations. High productivity could be achieved by using additional class libraries (see [Frank 92] for more details). The current version runs under Objectworks® 4.1 and VisualWorks® - on all platforms the appropriate Parc Place Smalltalk machine is available for.

The environment consists of three main tools: The *Value Chain Designer* (VCD) serves to design and analyze models of a firm's value chain. The *Object Model Designer* (OMD) supports the specification of object models. It provides various features to search for certain elements of an object model and to browse through the model. The *Workflow Designer* (WFD) allows for conveniently modelling business processes. It provides functions for simulation, fast prototyping and organizational analysis. The three tools are tightly integrated. The integration on the conceptual level has already been characterized above. System integration is accomplished by locating the tools within one Smalltalk image. The concepts managed by the tools can be annotated by using an integrated hypertext system.

The VCD provides a browser through the different elements of a value chain. The description of prototypical instances (like "human resource") is supported by predefined value sets (like "high", "average", "low") which can be modified interactively. This is the case, too, for relationships between activities. They can be characterized using an extensible list of identifiers (like "supports", "supported by", "contains", etc.). In addition to a textual representation relationships can be visualized in a graphical way. Items managed within the VCD can be associated to related items within the other tools. For instance: "activity uses business process". The VCD controls referential integrity of a value chain. Furthermore it provides dialogs to guide with analyzing and re-designing a value chain.

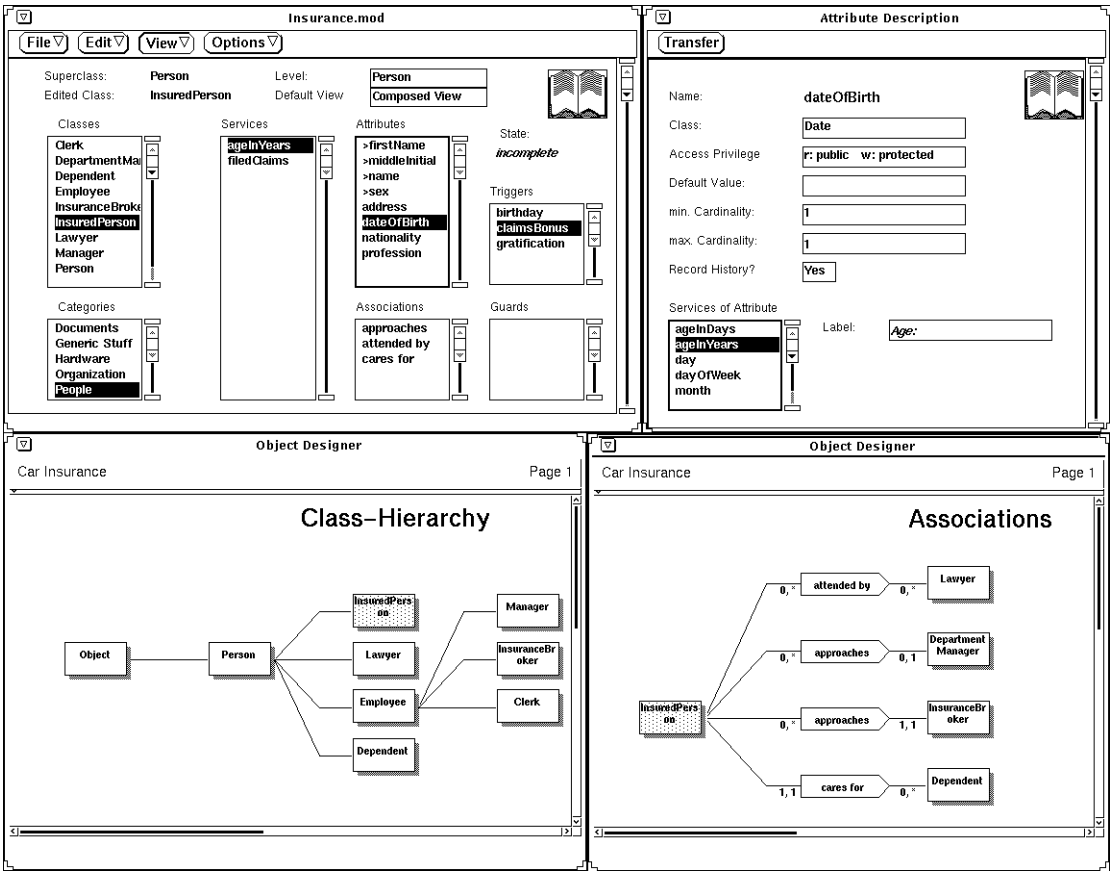

Figure 6. User-interface of the Object Model Designer

According to the meta model described above the OMD offers a number of different levels of abstraction. On the highest level class names are grouped into categories. On the next level a class can be assigned a list of attribute names, service names, guard names, and trigger names. Selecting an attribute, service, etc. causes the corresponding window to pop to the front. It allows for a more detailed description. Fig. 6 shows the window that contains the template for specifying an attribute - 'dateOfBirth' in the given example. Its class is 'Date'. The services which are provided by this class are shown in a listbox. If one of these services is needed for the class that is currently selected (which is 'InsuredPerson' in our example) it can be pasted to the services-listbox of this class - a convenient way to accomplish reusability. The OMD will then establish a reference to this service.

In order to foster system integrity it is not possible to type in a class name directly to characterize an attribute, a superclass, or a parameter. Instead it is required that the dictionary that contains all class names is updated first. Then the name can be pasted to the corresponding field. The OMD controls a number of integrity constraints. It prevents the user from deleting elements which are referenced by other elements, from assigning superclasses or classes of attributes in an inconsistent way, etc. The graphical representation of the class hierarchy can be used as an additional browser. The icons are mouse sensitive and cause the focus to shift to the selected class.

The main focus of the WFD is on business processes - both for analysis (that is mainly the organizational view) and design (organizational and information system view). On the highest level of abstraction the user can draw a business process picking from predefined icons within a specialized editor. When the procedure is described on this level (see fig. 7), the user can 'zoom' into it by selecting an icon - either a subtask or a document state. Within the example shown in fig. 7 the subtask 'Verification of substantial matter' is selected. Within the window titled 'Activity' it can be characterized by assigning execution times, an organizational unit, an organizational position, and possible exceptions. Furthermore the classes (like 'InsuredPerson', 'Policy', etc.), forms, and files which are needed as well as the involved roles can be listed in this window.

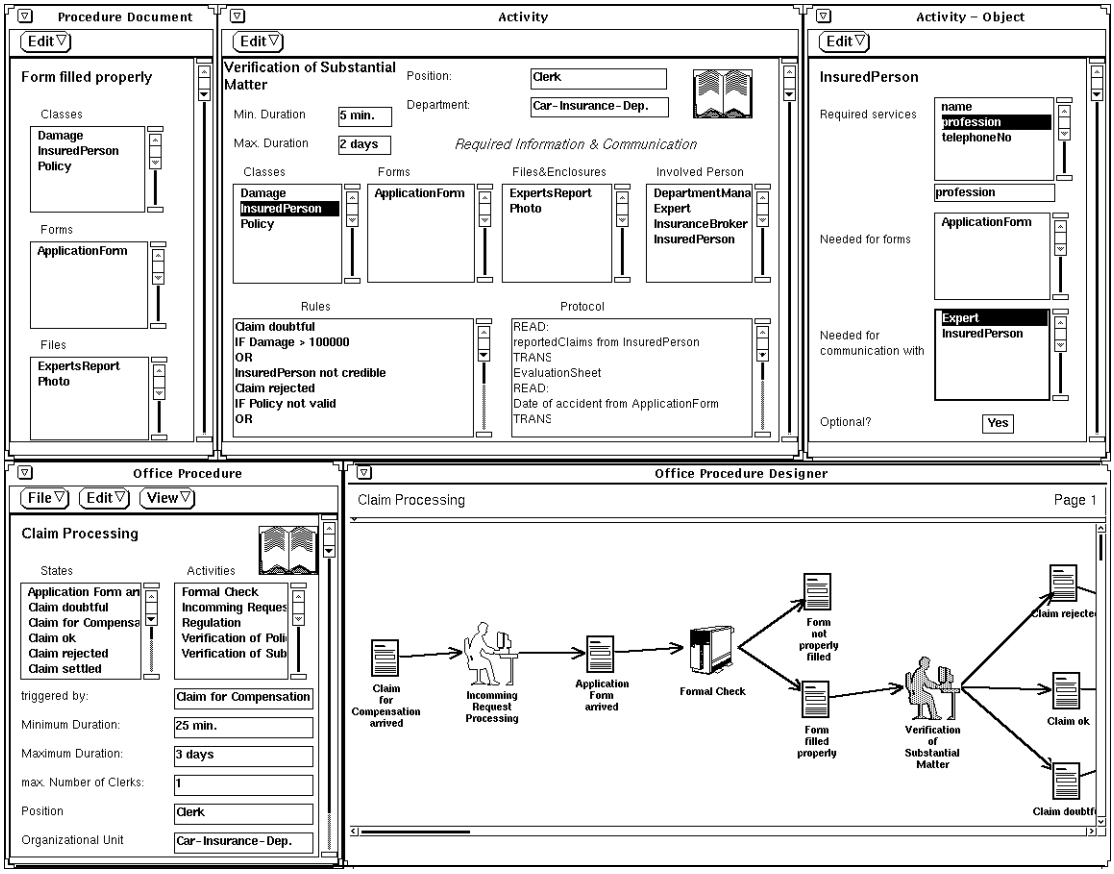

Figure 7. User-interface of the Workflow Designer

Selecting an item causes a window with an appropriate template to pop to the front. Within the example shown in fig. 7 this is the window in top right position. It allows to pick the required services from the selected class. For each service - or the object it delivers respectively - it can be specified what it is needed for: either for a form or for communicating with an involved person. The example shows that the service 'profession' is needed for the 'ApplicationForm' as well as for communicating with the 'InsuredPerson' and the 'Expert'.

Other templates exist to specify the relevant information within forms (fields together with an informal description of their purpose) or files (physical location, subject of interest within the file) and as well as the communication with involved persons (channel, subject). These specifications are used to generate a protocol which is shown in the right bottom corner of the 'Activity' window in fig. 7. Another text-widget within this window presents a template that can be filled to describe decision rules relevant for the focussed activity. A selected document state is specified in a similar way. First the classes, forms, and files which are involved in this state are assigned to it. Then these items have to be characterized in a more detailed way. The state of an object (which is represented by the name of its class) has to be defined by naming a boolean service that checks this state. That requires one to enhance the class with this service by switching to the OMD. For instance: An object of class 'Policy' is required to be in state 'valid'. That requires a service like 'isValid' to be specified for this class. For every form it has to be defined which fields have to be filled in. A file is characterized by its physical location and optionally the means of transportation to get it to the clerk. In order to support the user and to avoid inconsistencies the classes, forms and files assigned to a document state are pasted to the subtask that is triggered by this state. The WFD can generate a prototypical user-interface for every activity - provided the default views have been assigned to the corresponding classes in the object model (see above). The widgets placed within such an interface may be rearranged interactively. In order to use the simulation features built into the WFD it is necessary to first assign probabilities (using percentage values) to the states produced by every subtask. Furthermore it is required to type in the workload as number of procedures in a time period. After that the simulation can be started. Animation is accomplished by dynamically marking the subtasks which are active. If the simulation reveals any chances to improve the organization of the procedure the net can be rearranged interactively. Fig. 8 shows a snapshot of a simulation where the subtask 'Verification of substantial matter' has been expanded since it was found to be a bottleneck.

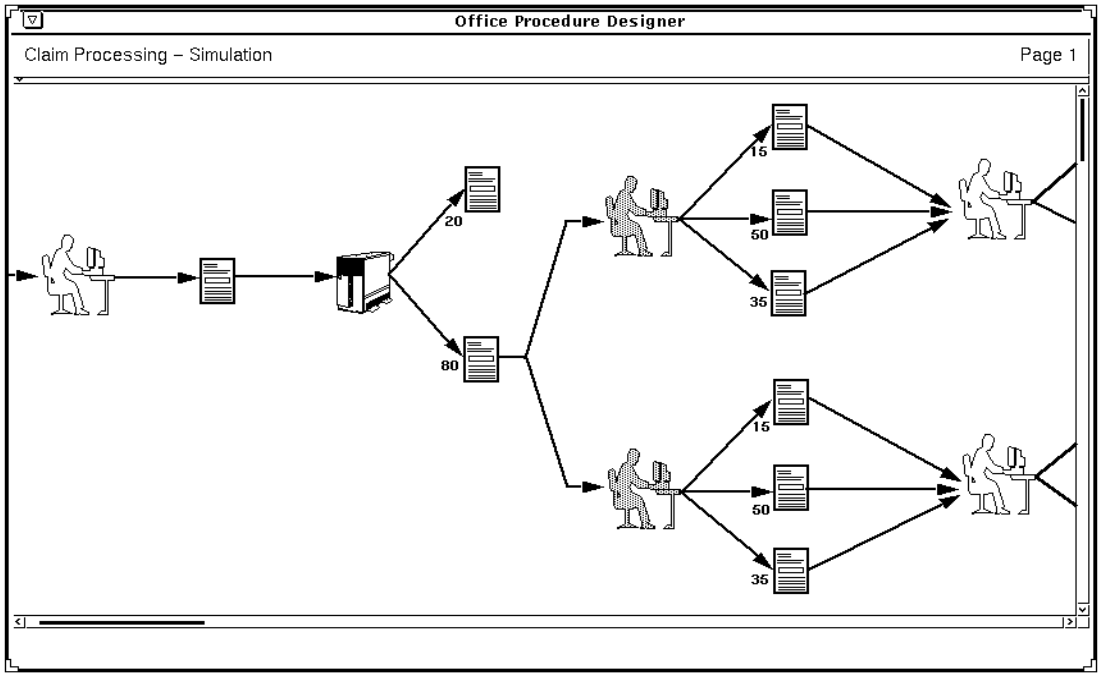

Figure 8. Snapshot of an animated simulation

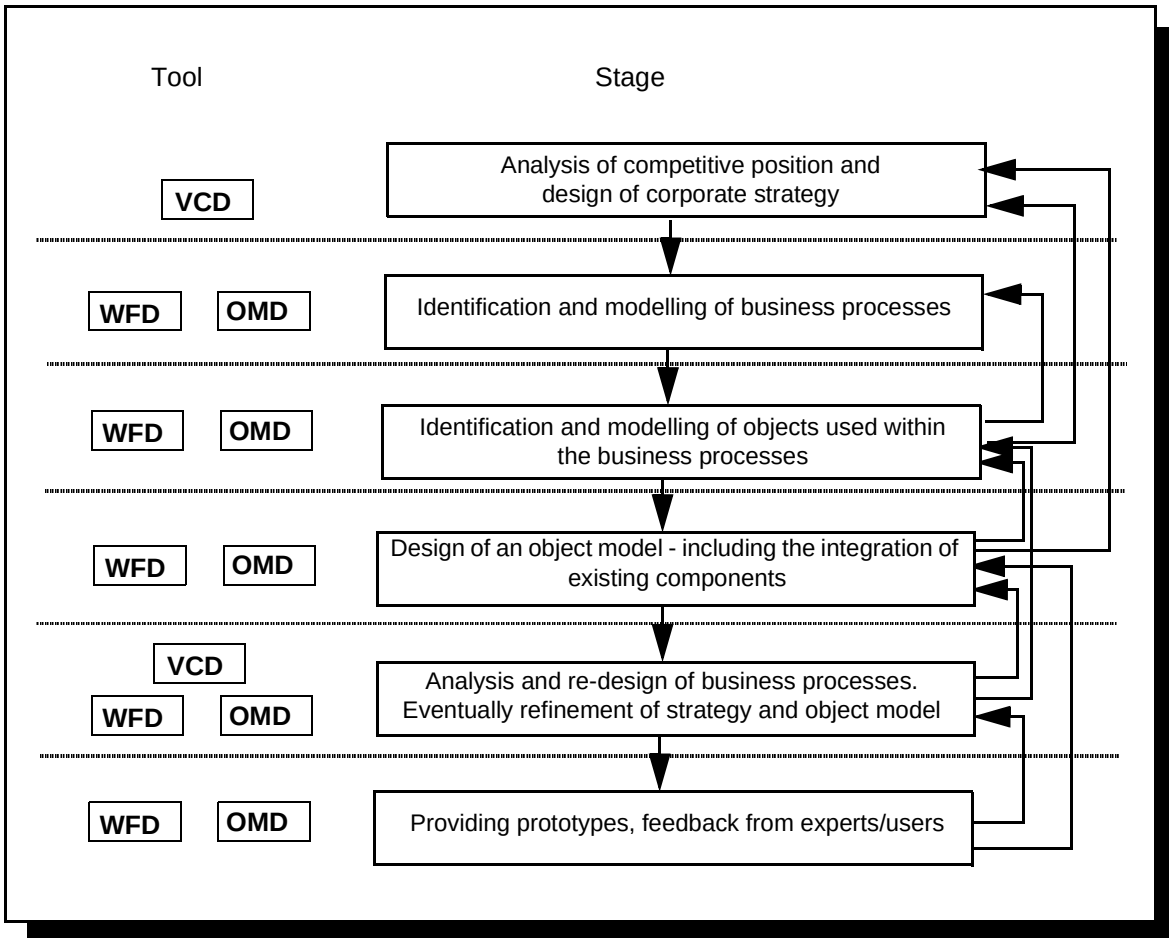

Figure 9. The usage of MEMO Center during important development stages (this is a simplified model!) and selected feedbacks between stages.

The WFD also allows for generating a report on detected media frictions and to graphically visualize communication relationships. Fig. 9 shows which tools are primarily used in the various stages of enterprise modelling.

4 Conclusions

Different from most other methodologies for object-oriented analysis and design MEMO not only focuses on the information system in itself but also on relevant strategic and organizational issues. Thereby it fosters a smooth integration of an information system with business processes and corporate strategies. Our experience with modelling office domains within insurance companies indicates that the proposed representations offer illustrative abstractions of an enterprise. This is especially the case for the representation of business processes. Both system analysts and domain experts intuitively understood the graphical notation. Thereby it proved to be a valuable medium for starting knowledge acquisition or object modelling respectively. We found that asking about strategies and processes is not only a prerequisite for organizational change. It furthermore helps to identify objects and specify their semantics. MEMO does - of course - not guarantee to optimize the organization of work. It can help however to reduce complexity by providing an illustrative representation of important aspects, by detecting certain types of organizational misconception, and - last but not least - promoting a sophisticated object-oriented architecture of a company's information system.

In the current version MEMO Center is restricted to one Smalltalk image. Therefore the prototypical workflow systems only simulate distribution. The objects used within the prototypes are either implemented directly in Smalltalk or refer to C-functions which are linked with the virtual machine. Currently we are working on extensions that will allow

for partial generation of distributed workflow management systems. For this purpose we use HP's "Distributed Smalltalk". It includes a CORBA implementation according to the OMG guidelines. MEMO Center will then allow for integrating any existing component that offers an IDL-interface. In order to allow for an automatic transformation of an object model designed according to MEMO into the OMG object model MEMO Center will be enhanced with means to distinguish between inheritance and subtyping. A text book with a comprehensive documentation of the methodology and the environment will be published by the end of 1994.

References

Booch, G.: Object-oriented design with applications. Redwood 1990

Coad, P.; Yourdon, E.: Object Oriented Design. Englewood Cliffs, NJ 1991

Davenport, T.; Short, J.E.: The New Industrial Engineering: Information Technology and Business Process Redesign. In: Sloan Management Review, Summer 1990, pp. 11-27

Dennis, A.R.; Hayes, G.S.; Daniels, R.M.: Re-Engineering Business Process Modeling. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 27th Hawaii International Conference on System Sciences, Vol. IV. Los Alamitos, Ca. 1994, pp. 245-253

ESPRIT Consortium AMICE: CIM-OSA AD 1.0 Architecture Description. Brussels 1991

Frank, U.; Klein, S.: Three integrated tools for designing and prototyping object-oriented enterprise models. GMD Research Report No. 689, Sankt Augustin 1992

Frank, U.: A Comparison of two Outstanding Methodologies for Object-Oriented Design. GMD Research Report No. 779, Sankt Augustin 1993

Frank, U.: An Object-Oriented Methodology for Analyzing, Designing and Prototyping Office Procedures. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 27th Hawaii International Conference on System Sciences, Vol. IV. Los Alamitos, Ca. 1994, pp. 663-672

Hammer, M.; Champy, J.: Reengineering the Corporation. New York 1993

Hassey, D.E.: Glossary of Management Techniques. In: International Review of Strategic Management. Vol. 3, 1992, pp. 47-75

Hong, S.; Goor, G.: A Formal Approach to the Comparison of Object-Oriented Analysis and Design Methodologies. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 26th International Hawaii International Conferenc on System Sciences, Vol. III., Los Alamitos 1993, pp. 689-698

Jacobson, I.; Christerson, M; Jonsson, P; Overgaard, G.: Object-Oriented Engineering. A Use Case Driven Approach. Reading, Mass. 1992

Katz, R.L.: Business/enterprise modelling. In: IBM Systems Journal, Vol. 29, No. 4, 1990, pp. 509-525

Keen, P.G.W.: Shaping the Future: Business Design through Information Technology. Cambridge 1991

Meyer, B.: Object-Oriented Software Construction. Prentice Hall 1988

Monarchi, D.E.; Puhr, G.: A Research Typology for Object-Oriented Analysis and Design. In: Communications of the ACM, Vol. 35, No. 9, 1992, pp. 35-47

Nierstrasz, O.; Dami, L.; De Mey, V.; Stadelmann, M.; Tsichritzis, D.; Vitek, J.: Visual Scripting: Towards Interactive Construction of Object-Oriented. In: Tsichritzis, D. C. (Hg.): Object Management. Geneva 1990, pp. 315-331

Peters, L.; Schultz, R.: The Application of Petri-Nets in Object-Oriented Enterprise Simulations. In: Nunamaker, J.F.; Sprague, R.H. (Ed.): Proceedings of the 26th International Hawaii International Conferenc on System Sciences. Vol. III, Los Alamitos 1993, pp. 390-398

Porter, M.E.: Competitive Advantage. New York 1985

Porter, M.E.; Millar, V.E.: How Information Gives You Competitive Advantage. In: Harvard Business Review, July/August 1985, pp. 149-160

Rumbaugh et.al.: Object-Oriented Modelling and Design. Hemel-Hempstead 1990

Scott Morton, M.S.: Strategy formulation methodologies. Cambridge, Mass. 1986

Sowa, J.F.; Zachman, J.A.: Extending and formalizing the framework for information systems architecture. In: IBM Systems Journal, Vol. 31, No. 3, 1992, pp. 590-616

Talwar, R.: Business Re-engineering - a Strategy-driven approach. In: Long Range Planning, Vol. 26, No. 6, 1993, pp. 22-40

Wiseman, C.: Strategy and Computers: Information Systems as Competitive Weapons. Homewood, Ill. 1985

Zachman, J.A.: A framework for information systems architecture. In: IBM Systems Journal, Vol. 26, No. 3, 1987, pp. 277-293