Design Principles for Application Integration

Paul Johannesson and Erik Perjons

Department of Information and Systems Sciences,
Stockholm University/Royal Institute of Technology,
Electrum 230, 164 40 Kista, Sweden
{pajo, perjons}@dsv.su.se

Abstract. Application integration is a major trend in information technology today. In this paper we present a number of principles for the design, validation and presentation of process models which align the applications of an organisation to its business processes. The design principles are divided into two groups. The first group consists of guidelines that obtain different views of the models and thereby facilitate for different stakeholders, e. g. business managers, designers and operators, to use common models and process languages. The second group of principles consists of guidelines to check the completeness of the process models. The paper also presents a process description language, BML (Business Model Language), which is tailored for modelling application integration.

1 Background

Three of the major trends in information technology today are the Web, enterprise software packages, and application integration. The Web provides an environment that can link a company's customers, suppliers, partners, and its internal users. Enterprise software packages offer an integrated environment to support business processes across the functional divisions in organisations. Some packages, like enterprise resource planning (ERP), for example SAP R/3 and BaanERP, manage back-office requirements, while other packages provide front-office capabilities, e.g. customer services. Common to Web applications as well as enterprise software packages is the need for application integration. Application integration is required to connect front office systems with back office systems, to transfer business processes to the Web, and to create extended supply chains involving customers, partners, and suppliers. Application integration is also needed for wrapping legacy systems and for migrating to new environments.

The demand for application integration is also fuelled by the move to process orientation in many organisations. Traditionally, organisations have been functionally divided, i.e. companies have been separated into departments such as market, production, and service. However, the functional organisation has been shown to have a number of weaknesses. In particular, it requires a huge administration to handle issues crossing functional borders, and considerable resources are allocated to tasks that do not create value. In order to overcome the problems of a functional organisation, companies have been concentrating on business processes, i. e. the set of related activities that create value for the customers. These processes cross the

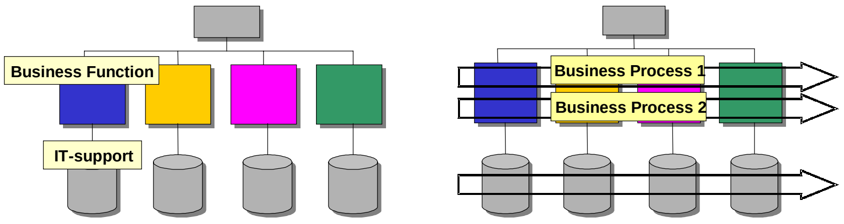

Fig. 0. The traditional function oriented structure, to the left, with the "stovepipe" like relation between business functions and IT systems. To the right the process oriented organisation which requires an integration of the IT systems.

internal borders of an organisation and also sometimes the external borders to other organisations, [6], [22], [27].

Supporting cross-functional processes raises new demands on the IT systems or applications. Traditionally, the applications have been built around departments or functions in the companies. The result has been a "stovepipe" like relation between the functions and the applications, where every function in the company is supported by its own IT-system or applications. This architecture is not satisfactory for process oriented organisations; to support the business processes in full the applications must be integrated, [18], see Fig. 1.

To handle this integration of applications in an efficient way, technologies, tools, and methodologies are required. One main technology is the Process Brokers, also called Process Management Systems, which aim at aligning the applications of an organisation to its business processes. A Process Broker provides an integrated, graphical environment in which all process logic for connecting applications can be encapsulated. The Process Broker enables users to visualise, construct, analyse, simulate and execute processes for application integration. Utilising the Process Broker technology for application integration is a complex design activity. Therefore, it requires adequate methodological support so that well-structured and easily understandable models can be produced. The purpose of this paper is to contribute to such methodological support by introducing a number of principles for the design, validation, and presentation of process models aligning the applications of an organisation to its business processes. The paper is a result of a joint project between the Royal Institute of Technology and Viewlocity, which aims at further developing technology, methods and Viewlocity's modelling language BML for application and process integration, [20], [27].

The remainder of the paper is organised as follows. Section 2 provides a brief overview of different architectures for application integration, in particular the Process Broker architecture. The section also discusses characteristics and problems of application integration. Section 3 describes related research about process modelling languages, and in Section 4 we describe a process modelling language BML (Business Model Language) which is used in the remainder of the paper. A classification of messages and processes, which is the base of the proposed design principles, is described in Section 5. In Section 6, we present our design principles with modelling examples. Finally, in Section 7, we summarise the paper and give suggestions for further research.

2 Architecture and Problems

2.1 Architectures for Application Integration

Integration of applications can be supported by many different architectures. One architecture for integrating applications is the point-to-point solution where every application is directly connected to every other application, see Fig. 2 (left). This solution could work for a small number of applications, but as the number of applications increases, the number of connections quickly becomes overwhelming. The Message Broker architecture reduces this complexity, see Fig. 2 (middle). The main idea is to reduce the number of interfaces by introducing a central Message Broker and thereby make it easier to support the interfaces. If one of the applications changes format, only one connection has to be changed: the one to the Message Broker, [26].

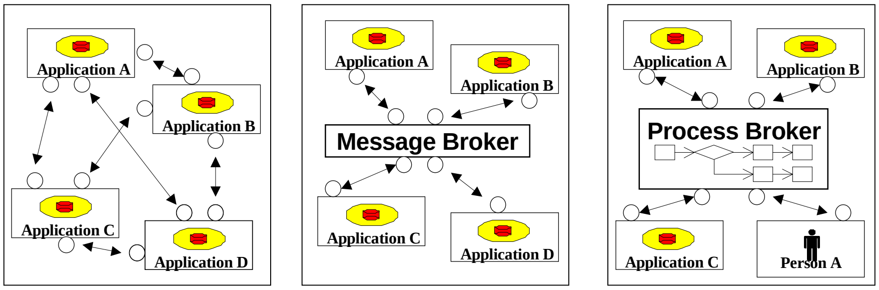

Fig. 1. The point-to-point strategy to integrate applications, to the left. In the middle the Message Broker architecture, which reduces the number of interfaces. To the right the Process Broker, which collect all process logic in the Broker.

The Process Broker, see Fig. 2 (right), is an extension of the Message Broker. In addition to handling format conversions, the Process Broker also encapsulates the process logic for connecting applications. When all process logic resides in one place, it becomes possible to study, analyse, and change the processes using a graphical interface, [5], [15], [27]. This visualisation reduces the complexity and enables different categories of people to take part in the process design.

2.2. Problems in Application Integration

The Process Broker technology requires methodological support so that designers can construct models that align the applications to the business processes. Experiences of a real world modelling study at a European telecommunication company demonstrate several problems when modelling application integration. The telecom company intended to handle an integration of applications by means of a Process Broker, which aimed at facilitating the complex interaction between administrative and technical applications. The problems we noticed can be summarised as follows:

Unstructured and complex models. Application integration often results in highly unstructured and complex models. One reason for this is that exception handling, which describes what to do when applications and human users do not respond in an expected way, makes up a large part of an application integration specification and thereby easily obscures the main business logic. Furthermore, there is often extensive communication between the Process Broker and different applications, which also tends to conceal the essential business logic.

Redundancy. The Process Broker does not maintain control over external applications, which means that these applications can be updated without the Broker being notified. As a consequence, it is often desirable to maintain redundant information that duplicates parts of the information in the external applications. This redundant information enables the Process Broker to maintain a complete, correct, and easily available record of its interactions with customers. However, this duplication of information requires mechanisms for handling possible inconsistencies.

Incomplete models. Since models for application integration tend to become large and complex, there is a risk that designers overlook parts of the models that are needed to maintain completeness and consistency.

Communication among stakeholders. It is possible to distinguish among four kinds of stakeholders: domain experts such as business managers, owners of external applications, business and technical designers, and operators that handle the day-to-day operations. Different stakeholders require different views of the system, while at the same time they need to be able to communicate with each other using common models and languages.

In the rest of the paper, we try to address these problems by proposing a set of design principles for application integration. We also introduce a process language called BML and argue that it facilitates communication between stakeholders. Table 1 summarises how the problems identified are addressed.

Table 1. The table shows in which section of the paper the identified problems are attended.

Problems:	**Section 4: Choice of language: BML**	**Section 5.2: Process Classification**	**Section 6.1: View guidelines**	**Section 6.2: Completeness Guidelines**
Unstructured and complex models		X	X	
Redundancy		X		
Incomplete models				X
Communication among stakeholders	X		X	

3 Related Work

In the beginning of the 90's process orientation became one of the most important trends in management practice as well as research. Authors such as Hammer and Davenport, [6], [14], advocated a radical change from a functional perspective to a process focussed perspective in order to improve customer satisfaction, shorten lead times, increase productivity, and handle technological development. Initially, process orientation achieved most attention in the manufacturing discipline, but in recent years it has also gained prominence in the information systems community, [12].

A number of languages and methodologies for process specification and design have been proposed. Many of these languages are based on Petri nets, [21], e.g. UML activity diagrams. A distinction can be drawn between *activity oriented* and *communication oriented* process languages. An activity oriented process language, e.g. UML activity diagrams or EPC, [23], is intended for handling arbitrary processes including material processes involving physical actions. Therefore, the activity oriented language diagrams usually represent a mix of automated and manual actions. A communication oriented language, on the other hand, focuses on communicative processes describing the interaction between people and systems in terms of sending and receiving messages, which provides an opportunity to support the communication by means of information technology. Communication oriented languages have been heavily influenced by speech act theory, [24]. One of the first systems based on a communication and speech act oriented approach was the Coordinator, developed by Winograd and Flores, [25], which supported the communication process in the work place. The idea of applying a speech act based approach to information systems analysis and design was also employed by the SAMPO (Speech-Act-based office Modelling aPprOach) project in the middle of the eighties, [3]. These ideas were further developed in recent work on Business Action Theory, [13], [17], and in the DEMO (Dynamic Essential Modelling of Organisations) approach, [7], [8]. We believe that a communication oriented approach is particularly suitable for application integration and Process Brokers, as application integration basically consists of the interchange of messages between systems and people.

A technology related to the Process Broker is the Workflow Management System, [16]. The first generation of Workflow systems, during the 80’s and the early 90’s, was supporting communication between people, concentrating on document routing. The next generation of Workflow technologies put the business processes in focus. By also involving automatic actors, the automation of the processes could be facilitated further, [18]. The next step for the Workflow systems should be to implement enterprise wide workflow solutions and provide tools for managing the processes themselves. This process management includes process modelling, process reengineering as well as process implementation and automation, [11], [27]. The Process Broker can be seen as this next step to process management, by providing modelling and simulation capabilities, but the Process Broker also enables rapid modifications of the business processes thanks to its flexible way of handling application integration.

In the marketplace, a new breed of middleware technologies focused on enterprise application integration has emerged. Message Broker vendors provide functionality that enable applications to interoperate with a minimum of custom coding. Some of the major products here are Viewlocity’s AMTrix and SmartSync, [1], IBM’s MQSeries Integrator, [19], and Entire Broker from Software AG, [9]. Several of the Message Broker vendors are adding process modelling and simulation capabilities to their products, thereby moving into the Process Broker market. Some of the major products in this market are: Viewlocity’s SmartSync Model Manager [27], Extricity Software’s AllianceSeries [10], Vitria Technology’s BusinessWare [2], and HP’s Changeengine [15].

4 BML – A Language for Application Integration

To visualise the application integration there is a need for a process description language. This section briefly introduces such a language, BML (Business Model Language), which is developed by Viewlocity, [27]. The language has similarities to SDL (Specification and Description Language), [4], but is more adapted to application integration. BML is a communication oriented process language, see Section 3, which means that it focuses on describing interaction between systems through the sending and receiving of messages. This makes the language suitable for application integration and Process Brokers. Another important advantage of BML is that the language can be used for the business specification and design as well as operation of systems. This means that the same language can be used in different phases of a system's life cycle: in feasibility analysis, in requirements specification, in the design and implementation phases, and even in the operation phase. This enables different categories of stakeholders to use the same language for different purposes. The language can also be used directly as an implementation language and to some extent replace ordinary programming language. Further advantages with BML are its capability to describe and partition the interaction and interfaces between processes that work concurrently. Concurrency is common in application integration, when for example several applications are to be updated in parallel. The possibility of partitioning in BML reduces the complexity of handling large systems, through creating manageable and understandable parts with limited dependencies.

BML can describe the structure as well as the behaviour of a system by using two kinds of graphical diagrams. The structure of the system is visualised by a static diagram, see Fig. 3, which describes the processes in a static mode. The static diagram describes the messages sent between the processes and between the processes and the environment, i. e. the external applications and people.

The dynamic behaviour of a system is described by using process diagrams, see Fig. 4. These diagrams can be seen as templates, visualising the order in which the messages shall be sent and received. For each process diagram there is a number of process instances, that are created during runtime. The process instances execute independently of each other, but can communicate by sending and receiving messages asynchronously. Each instance has an input queue, see down to the left in Fig. 4, where received messages are stored. A process instance can either be waiting in a stable state or perform a transition from one state to another. A transition is initiated when a message in the input queue is chosen and consumed.

Following the example in Fig. 4, the process instance starts in a Start state (circle without a name). Only the messages *m1* from *process a* and *m2* from *process c* can initiate a transition. The message *m1* is first in the queue and is therefore consumed, and the process instance performs a transition to the state *Wait for Event 1*. During the transition a message *m3* is sent to *process c*. Thereafter the message *m9* is first in the queue. Since only message *m5* can initiate a further transition from *Wait for Event 1*, the message *m9* is discarded. The next message in the queue is then *m5*, which can initiate a transition from *Wait for Event 1* to some other Wait for Event state (not specified in the example). During the transition data can be manipulated, decisions can be made, new process instances can be created and messages can be sent to other process instances or to the process instance itself.

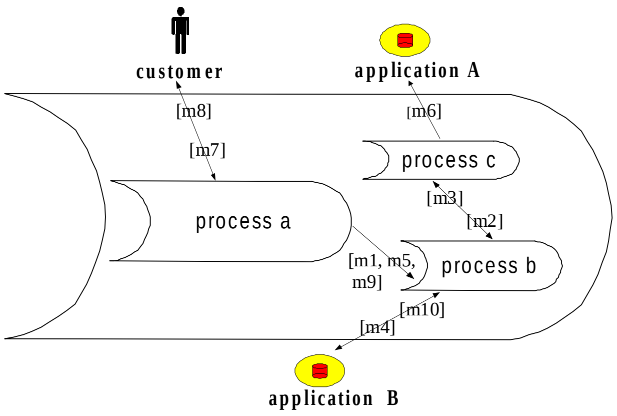

Fig. 2. The static diagram in BML visualises the structure of the processes in a system.

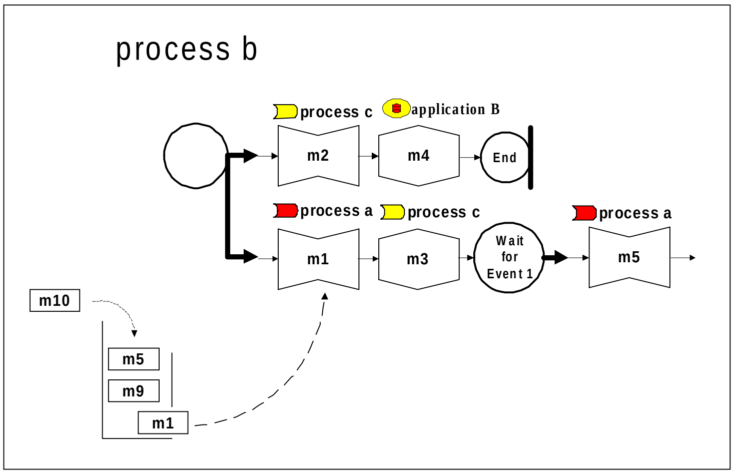

Fig. 3. A process instance with the input queue.

An important feature of BML is also the data model, which is not used in the examples of this paper. Each process diagram has a data model that describes the structure, type and meaning of the data that is handled in the diagram. The data model also describes the structure of the data in the different messages.

The main BML symbols are the following, see also Fig. 5:

Wait for Event and **Start.** The process instance is waiting in the Wait for Event state until a message is received or a timer has expired. A Wait for Event symbol without a name is the starting state.

End. Describes the end of the flow of the process instance.

Receive Message. Describes the consumption of a message from the input queue.

Send Message. Describes the sending of a message.

Automated Business Activity. Describes operations that will be performed on the process instance.

Automated Business Decision. The control flow is dynamically changed depending on different business rules.

Start Timer and **Expire Timer**. In application integration, the notion of time is important and timers occur frequently to obtain delays and supervision. When a timer is started it will be provided with a timeout value. The starting is represented by an hourglass “full of time”, and the timeouts by an hourglass “out of time”.

Application and **Human actor**. Both are symbols of external actors.

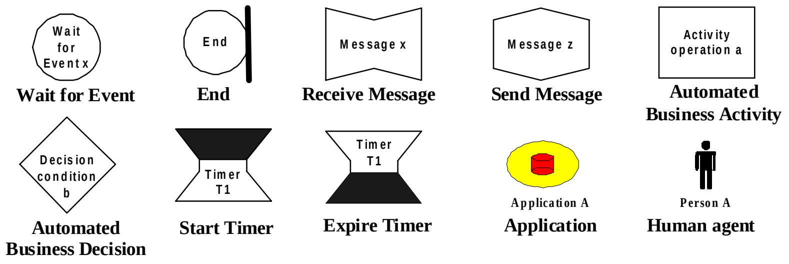

Fig. 4. Symbols used in BML

5 Classifications of Messages and Processes

5.1 A Classification of Messages

In this section, we introduce a classification of messages as a basis for formulating the design principles in Section 6. The classification is based on speech act theory. However, it is our experience that users often find it difficult to classify messages directly according to the basic speech act types. Therefore, we introduce a set of message types which are more closely related to the messages typically found in application integration.

The study of speech acts has been an active research area in analytical philosophy since World War II, and the most influential approach to date is speech act theory as developed by John Searle, [24]. Searle proposes a taxonomy for speech acts consisting of five classes: assertives, commissives, directives, declaratives, and expressives. These are also called the illocutionary points of a speech act. An assertive is a speech act, the purpose of which is to convey information about some state of affairs of the world from one agent, the speaker, to another, the hearer. An example is “It is raining”. A commissive is a speech act, the purpose of which is to commit the speaker to carry out some action or to bring about some state of affairs. An example is “I promise to be at home before nine o’clock”. A directive is a speech act, where the speaker requests the hearer to carry out some action or to bring about some state of affairs. An example is “Please bring me the salt”. A declarative is a speech act, where the speaker brings about some state of affairs by the mere performance of the speech act. An example is “I hereby pronounce you husband and wife”. An expressive is a speech act, the purpose of which is to express the speaker’s attitude about some state of affairs. An example is “I like coffee”.

Based on Searle’s classification of speech acts, we give a list of message types below which frequently occur in application integration. The message types are requests for information and services and the responses to these requests. We also identify messages for reserving, booking, and releasing resources. The difference between reserving and booking is that reserving is a preliminary stage to booking. A reservation could either be followed by a booking or a cancelation of the reserved resource. The distinction is important if the system automatically should cancel reserved resources that not have been booked after a certain time. For example, a person can reserve several telephone numbers for a certain time, so that he or she can choose to book one or more numbers from them. The reserved numbers which are not booked after a time limit are automatically released so that other persons can reserve or book the numbers.

The message types are the following:

Information request. An information request is a directive speech act in which the sender asks the receiver for a piece of information. *Example: What is the telephone number to the help desk?*

Service request. A service request is a directive speech act in which the sender asks the receiver to carry out a service. Typical examples of services are to deliver a product, get an authorisation, and booking a resource. In contrast to an information request, a service request does not ask for information about a state of affairs – it requires a state of affairs to be changed. *Example: Provide me with a new telephone subscription.*

Reservation request. A reservation request is a special service request in which the sender asks the receiver to reserve a resource for a period of time, meaning that the resource cannot be reserved or booked by anyone else during this period of time. *Example: Reserve five different telephone numbers (which the customer can choose from).*

Booking request. A booking request is a special service request in which the sender asks the receiver to make a resource available for the sender. *Example: Book the telephone number that the customer has chosen.*

Information confirmation. An information confirmation is an assertive speech act in which the sender, in response to an information request, provides the receiver with a piece of information. *Example: The telephone number to the help desk is 07-70 70 70.*

Service confirmation. A service confirmation is an assertive speech act in which the sender, in response to a service request, informs the receiver that the required service has been carried out. *Example: You have been provided with a new telephone subscription.*

Reservation confirmation. A reservation confirmation is a special service confirmation in which the sender, in response to a reservation request, informs the receiver that the required reservation has been made. *Example: The telephone number is reserved (until the customer has chosen to book the number or a certain time limit has passed).*

Booking confirmation. A booking confirmation is a special service confirmation in which the sender, in response to a booking request, informs the receiver that the required booking has been made. *Example: The telephone number is booked.*

Service promise. A service promise is a commissive speech act in which the sender, in response to a service request, commits itself to carry out the required service. *Example: The delivery department promises to send the ordered telephone.*

Notification. A notification is an assertive speech act in which the sender informs the receiver about the changes of some state of affairs. *Example: The customer has started to use the subscription.*

Cancel reservation. A cancel reservation is a directive speech act in which the sender asks the receiver to cancel a previous reservation. *Example: Release a reserved number.*

Cancel booking. A cancel booking is a directive speech act in which the sender asks the receiver to cancel a previous booking. *Example: Release a booked number.*

5.2 A Classification of Processes

In this section, we introduce a classification of processes. The purpose is to support the designer in building well-structured and easily understandable application integration models. The classification identifies types of processes which are largely independent of each other and which can be combined with clear and simple interfaces. This makes it possible to partition a system of processes into manageable and understandable parts.

A starting point of the classification is the customer, an actor for whom a process is to create value. By emphasising the customer, we can distinguish between customer oriented processes that directly interact with the customer, processes that support the customer oriented processes in various ways, and processes that manage more technical and informational aspects.

The classification identifies the following types of processes:

Customer process. A customer process focuses on the interaction with the customer. A customer process may contain messages to or from a customer or another process types, but not to and from external actors, i.e. applications or people (except the customer). The purpose of a customer process is to show the business logic from the customer’s point of view.

Interface process. An interface process handles the interaction with the external applications or people (except the customer). An interface process may contain messages to and from all other types of processes as well as to and from external applications and people. An interface process interacts with exactly one external application or person. The purpose of the interface processes is to insulate the interfaces of external applications from the main business logic. For example, when the format of messages sent from an external application changes, only the data model in the interface process has to be modified while the customer processes can be left untouched. There are two subtypes of interface processes:

Request process. A request process handles information or service requests from other processes.

Release process. A release process handles cancel reservations or cancel bookings from other processes.

Synchronisation process. A synchronisation process synchronises a number of interface processes. It may contain messages to and from different types of processes, but not from external applications and people. The purpose of a synchronisation process is to encapsulate a piece of business logic – typically a synchronisation process takes care of a request from a customer process by invoking and synchronising a number of interface processes.

Maintenance process. A maintenance process handles the internal storage of information that duplicates the information in external applications, see Redundancy in Section 2.2. There are two subtypes of maintenance processes:

Update process. An update process takes care of a notification from a customer or synchronisation process and stores the information carried by the notification.

Consistency process. A consistency process is a process that checks whether there is any inconsistency between internally stored information and information in external applications. A consistency process also takes appropriate action when an inconsistency is detected.

A typical structure of a model using the process classification is shown in Fig. 11. The customer process contains the main business logic with respect to the customer. It interacts with synchronisation processes and interface processes in order to take care of the customer's requests. It also sends notifications about the customer interaction to maintenance processes, not shown in Fig. 11. We believe that this structure supports flexibility, stability, and understandability by separating main business logic from more technical and informational aspects. By using the design principles of Section 6, designer will automatically arrive at an application integration model with the proposed structure.

6 Design Principles

In this section, we introduce a number of design principles in the form of guidelines for the design, validation, and presentation of application integration models. These guidelines are divided into two groups. The first group consists of guidelines to obtain different views of process models, while the second group consists of guidelines to check the completeness of process diagrams.

The main idea behind the guidelines in the first group, the view guidelines, is to obtain a series of views of processes starting with a customer oriented view on the business level. This first view means that the Process Broker can be seen as a mediator between the customer, i.e. the one for whom value is to be created, and a set of applications and people, see Fig. 6. Note that the customer does not communicate directly with the applications and other people, but only through the Process Broker.

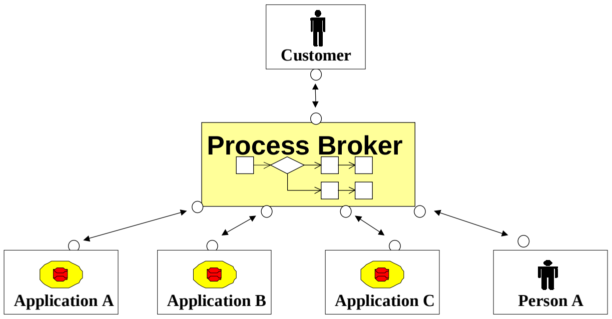

Fig. 5. The customer does not communicate directly with the applications and other people, but only through the Process Broker.

The succeeding views add more and more details moving from a business perspective to a more technical perspective. Each view is an extension of the previous one, either through adding subprocesses or through introducing new components into the existing diagrams. Note that the Process Broker contains all the modelled processes in the example, i. e. the Process Broker is the internal system, while the external system is the one represented by applications and people.

The purpose of the guidelines in the second group, the completeness guidelines, is to support the designer in creating processes that include complete discourse structures and not only fragments. In particular, the completeness guidelines can be used to ensure that requests are always handled in an appropriate way, and that outstanding bookings and reservations are taken care of in cases of exception.

6.1 View Guidelines

In this section, we present a number of views supporting a top-down approach. Each view is illustrated by means of a telephony case, in which a customer wants to order a subscription.

View 1. Customer interaction. This view models the interactions between the Process Broker and the customer, i.e. the messages exchanged by the customer and the Broker as well as the flow of control. In this view, there is only one process diagram.

The first tasks of the designer in this view are to clarify how the process is initiated, what messages the customer sends to the Process Broker, and what messages the Process Broker sends to the customer. Based on this information, the designer constructs a static diagram, see Fig. 7. The corresponding process diagram is shown in Fig. 8. The *Order subscription process* is initiated when it receives a message called *Order initiated* from the customer. Furthermore, the customer asks for a number of telephone number suggestions, *Request number proposals*. The answer from the Process Broker, is the message *Number proposals*. The customer can now choose one of the numbers or ask for further telephone number suggestions if he or she is not satisfied by those suggested. The receiving message from the customer, *Customer response*, is therefore evaluated in a decision point, *Nr chosen*. If the customer has not chosen a number the process instance follows the "false" path back and the customer can ask for more numbers. If the customer has chosen a number the order is taken care of. The Process Broker will then inform the customer about the status of the order, *Reporting of the order's state*. Finally, if the order is approved of, the customer will get further information about the subscription, *Information delivery*.

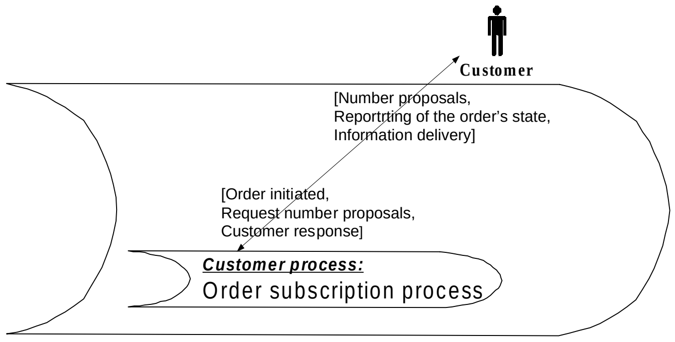

Fig. 6. The static diagram in view 1

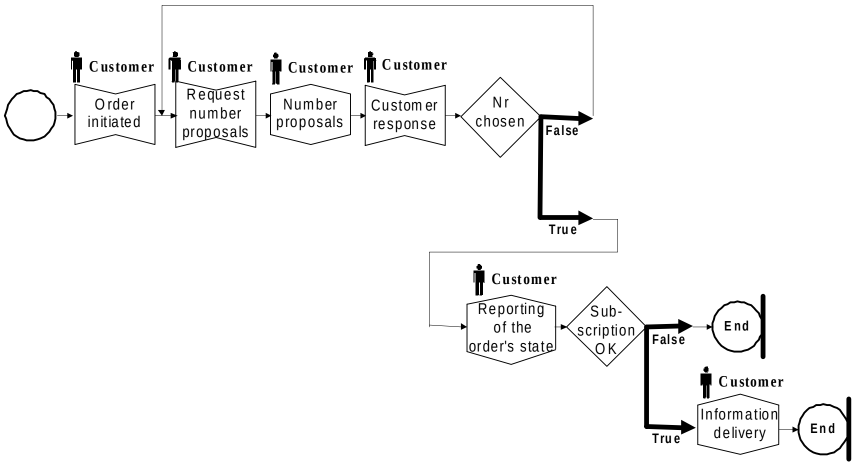

Fig. 7. The Order subscription process (view 1).

View 2. Broker requests. This view is an extension of view 1, which describes how the Process Broker produces the messages it sends to the customer. For each Send Message symbol from the Broker in view 1, a pair of one Send Message symbol and one Receive Message symbol is added.

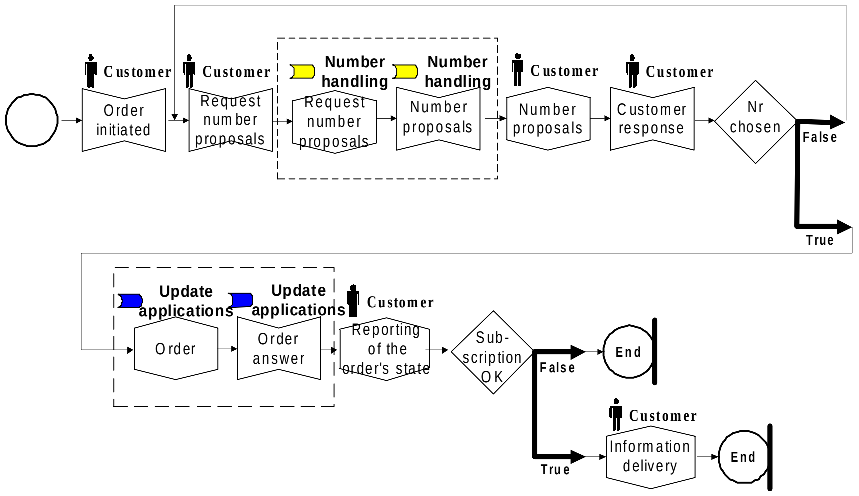

Fig. 8. The Order subscription process (view 2).

The first thing the designer must do when creating this view is to determine how the messages sent by the Process Broker are to be produced, which means that Send Message symbols in view 1 should be identified and analysed. Before the Send Message symbol *Number proposals* in Fig. 9, one Send Message symbol, *Request number proposals*, and one Receive Message symbol, *Number proposals*, are added, see symbols surrounded by the upper dotted box in Fig. 9. (Note that the dotted boxes are not part of the BML notation; they are used in the examples to help the reader identify the extensions for every new view introduced.) The Send Message and Receive Message symbols in the upper dotted box represent the messages sent to and received from a new subprocess, *Number handling process*. In view 2, the applications to be integrated are still not visible. They will become visible in the next view, where the introduced subprocesses are modelled.

View 3. External system interaction. Each subprocess introduced in view 2 is specified here. Only information and service requests and the responses to these are included in this view.

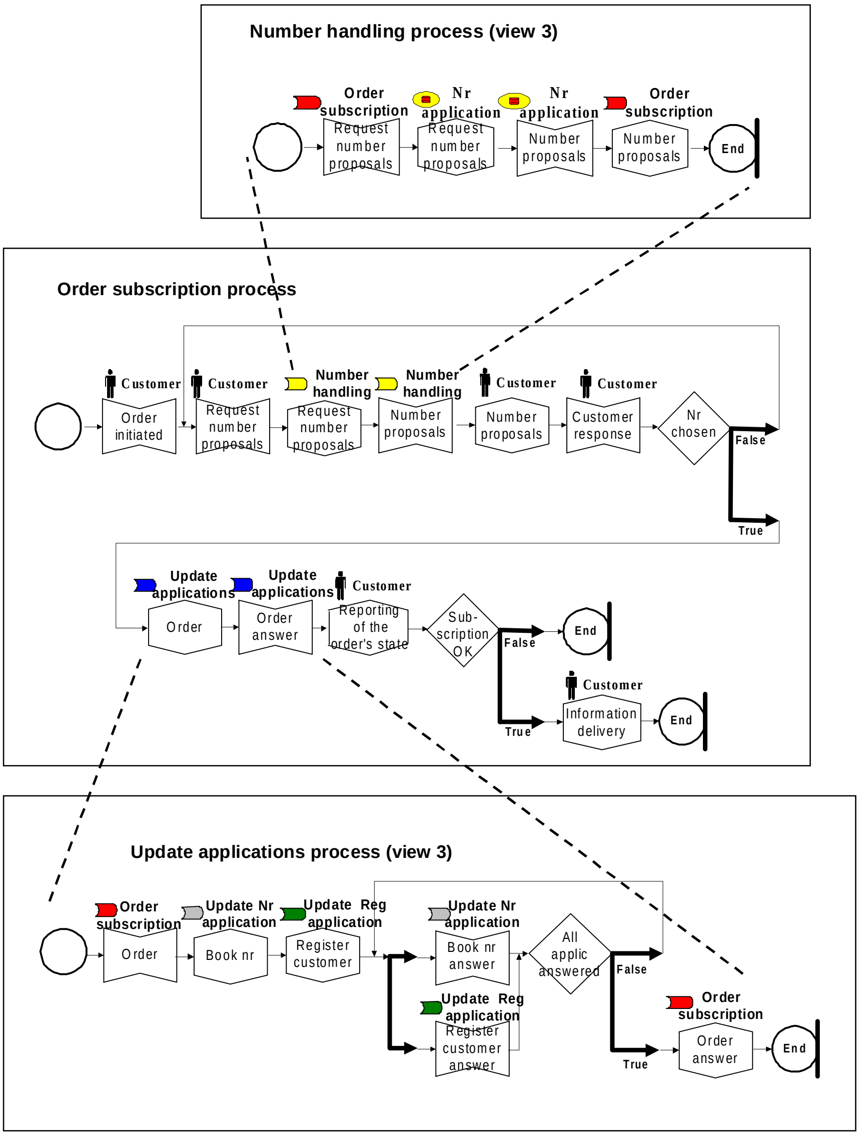

Fig. 9. Some of the process diagrams in view 3.

This view models the interaction with the applications to be integrated. In the simplest case, a subprocess is an interface process, i.e. it communicates with exactly one application, see Section 5.2. An example of an interface process is *Number handling process* at the top in Fig. 10, which makes a call to one application, *Nr application*, which returns an answer, *Number proposals*. The *Number handling process* finally forwards the answer to the process that invoked the subprocess, i.e. *Order subscription process*.

In some cases, it is convenient to introduce two or more levels of subprocesses. If the subprocess to be specified requires interaction with several applications, the designer should first construct a synchronisation process, see Section 5.2. The synchronisation process *Update applications process*, at the bottom in Fig. 10, invokes its own subprocesses and synchronises these. These subprocesses, *Update Nr application process* and *Update Reg application process*, and the relation to the synchronisation process, *Update application process*, are shown in the static diagram in Fig. 11. Each of the invoked subprocesses is an interface process and look like the interface process *Number handling process* in Fig. 10.

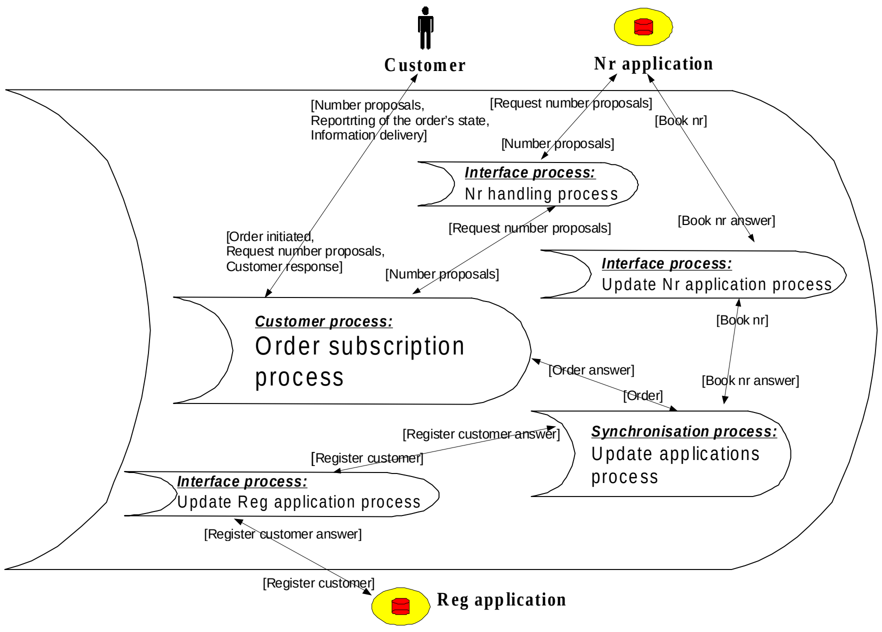
Customer
Nr application
[Number proposals, Reportrting of the order's state, Information delivery]
[Request number proposals]
[Book nr]
[Number proposals]
Interface process: Nr handling process
[Request number proposals]
[Book nr answer]
[Order initiated, Request number proposals, Customer response]
[Number proposals]
Interface process: Update Nr application process
[Book nr]
Customer process: Order subscription process
[Order answer]
[Order]
[Book nr answer]
[Register customer answer]
[Register customer]
Synchronisation process: Update applications process
Interface process: Update Reg application process
[Register customer answer]
[Register customer]
Reg application

Fig. 10. The static diagram in view 3.

View 4. Exception handling. This view specifies the exception handling. For each Receive Message symbol, whose sender is an external application or a human actor, a Start timer and an Expire timer are added as well as the behaviour when the exception is raised, see dotted boxes a) and c) in Fig. 12.

Views 1 – 3 specify only the normal course of events. In view 4, the designer specifies how to handle exceptions, i.e. situations where an actor has not replied to a request within a pre-specified time limit. This means that the designer first has to extend all interface process diagrams by adding Start timers and Expire timers, as well as the behaviour after a timer has expired. Furthermore, the designer may have to extend the process diagrams at a higher level, i.e. the synchronisation or the customer processes, in order to describe how to handle error messages from the interface processes. An example of this is shown in Fig. 12, describing the interface process *Number handling process* which returns a message, *Number Proposals="No answer in time"*, to the *Order subscription process* when the timer has expired. The *Order subscription process* then has to handle the "No answer in time" message. Note that the timer symbols can only be found in the interface processes and sometimes in the customer process. The latter situation occurs if the designer wants the system to handle situations where the customer does not answer in time.

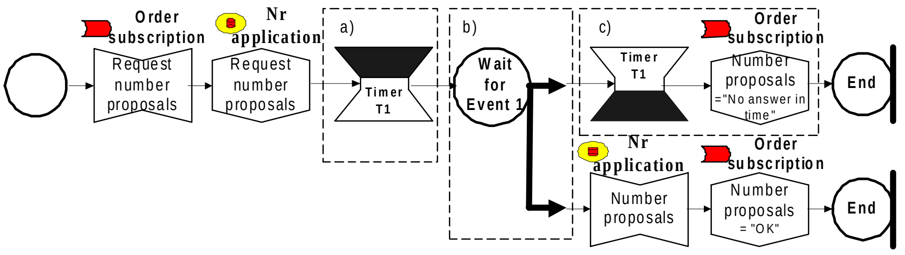

Fig. 11. The Number handling process (view 4).

In this view Wait for Event state symbols must also be added before every Receive Message symbol, see dotted box b) in Fig. 12. Note that before the first Receive Message symbol, *Request number proposal*, there is already a state symbol, the Start symbol. The Wait for Event and Start symbols are described in section 4.

View 5. Resource releasing. This view adds all messages of the type Cancel reservation and Cancel booking. When a process cannot be completed as intended, it becomes necessary to release the resources that have been reserved or booked during the process. This releasing of resources is handled in view 5 by extending the process diagrams accordingly by introducing certain release interface processes, see Section 5.2. In Fig. 13 the messages *Delete booking* and *Delete registration* are sent to the release interface processes *Delete Nr application process* and *Delete Reg application process*, see dotted box.

View 6. Notifications. This view adds all messages of the type Notification.

There are two main types of situations where notifications are required. First, when an exception has occurred, it is customary to inform an operator about this event so that he or she may take appropriate action. Secondly, a notification may be sent to a maintenance process, see Section 5.2, which redundantly stores information about essential states of affairs.

6.2 Completeness Guidelines

The completeness guidelines below exploit the fact that messages typically occur in certain dialogue structures. A very simple dialogue structure consists of a pair: a question followed by an answer. Another well-known dialogues structure is the conversation for basic action, introduced by Winograd and Flores in [25], which consists of the four steps: request, negotiation, fulfilment, and acknowledgement.

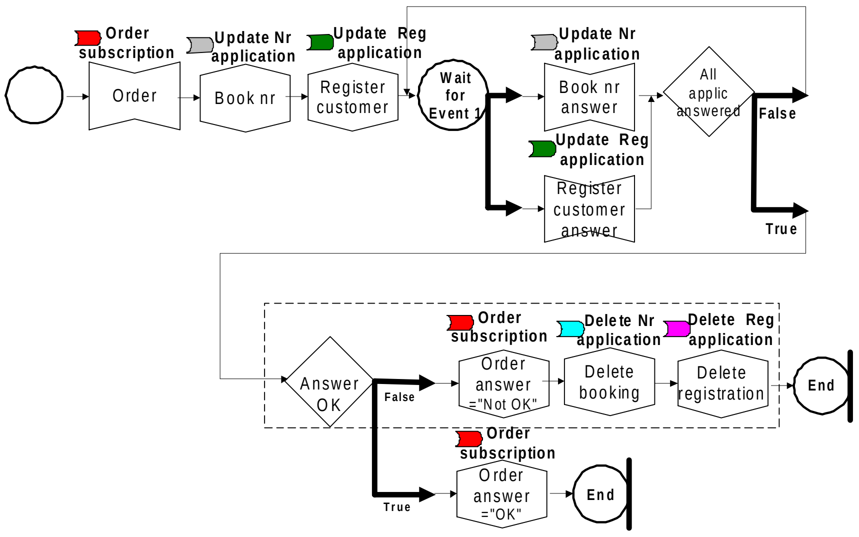

Fig. 12. The Update application process (view 5).

The following guidelines are a preliminary result of our research. Further research will produce additional guidelines.

1. In a process diagram, every information request and service request should be followed by a corresponding information confirmation or service confirmation, respectively. This pair of request and confirmation is optionally followed by a notification.
2. In a process diagram, every reservation request should be followed by a corresponding booking request by the same actor.
3. In a process diagram, every reservation request should be followed by a corresponding cancel reservation. A process instance typically takes this path when some intermediate request has not been satisfied.
4. In a process diagram, every booking request should be followed by a corresponding cancel booking. A process instance typically takes this path when some intermediate request has not been satisfied.

7 Concluding Remarks and Further Research

In this paper we have addressed methodological support for modelling the aligning of application integration to business processes. The main contribution is the guidelines introduced above which can be used in several ways. First, they can be used in design.

A designer could utilise the view guidelines by first constructing a process diagram according to view 1 and then gradually refine it until a set of diagrams in view 6 is obtained. Furthermore, the designer can use the completeness guidelines to guide the design of each individual process diagram. Secondly, the guidelines can be a support for validation and verification. By checking the completeness guidelines, a designer can ensure that essential parts of discourse structures and exception handling are not omitted. Thirdly, the guidelines can be used for presentation purposes. Business oriented users can choose to see only the top view or views, while technical designers and implementers can proceed to lower views. Even for the latter category of stakeholders, the layered views can help to understand a system by allowing to focus on an essential business perspective first and thereafter to proceed to a technical perspective. Different categories of users, for example customers, business and technical designers, have the possibility to suggest input on the right level in the modeling process. Business designers probably want to concentrate on the important parts of the business processes to clarify how they want the main business logic to work.

We intend to follow up the work presented in this paper by further research which goes in several directions. One is to compare designers following the suggested view guidelines with designers who are not following them. There is also a possibility to let different kinds of stakeholders design the models. Such an empirical study could give input to refined or additional design principles. Another direction is to find further dialog structures to produce additional completeness guidelines for validation of the models. To obtain a more complete methodology there is also a need for guidelines helping to design the data models, which describe the structure, type and meaning of the data beeing handled in the process.

Acknowledgements

This work was performed as part of the NUTEK (Swedish National Board for Industrial and Technical Development) sponsored project Process Broker [20]. The authors are grateful to Jan-Owe Halldén, Mikael Nilsson and Christer Wåhlander at Viewlocity for their valuable suggestions and knowledge. We also tank our colleagues at the Royal Institute of Technology, especially Birger Andersson, S.J. Paheerathan**,** Prasad Jayaweera, Nasrin Shakeri and Benkt Wangler for commenting earlier versions of this paper.

References

1. AMTrix. Viewlocity. URL: http://www.viewlocity.com/solutions/, 1999-11-25
2. Atwood, R.: Bringing Process Automation to Bear on the Task of Business Integration. Vitria Technology (1999). URL: http://www.vitria.com/products/whitepapers/ seyboldwp.html, 1999-11-25
3. Auramäki, E., Lehtinen, E., Lyytinen, K.: A Speech Act Based Office Modelling Approach. In: ACM Transactions on Office Information systems. Vol. 6, no. 2 (1988) 126-152
4. Belina, F., Hogrefe, D., Amardeo, S.: SDL with Applications from Protocol Specification. Carl Hanser Verlag and Prentice Hall International, UK (1991)

5. Butterworth, P.: Automating the Business Processes of Mission-Critical Distributed Applications. Forté Software (1997). URL: http://www.forte.com/product/downloads.html, 1999-10-04
6. Davenport, T.: Process Innovation: Reengineering work through information technology. Business School Press, Boston (1993)
7. Dietz, J.: Modelling Communication in Organizations. In: Riet R. v. d. (ed): Linguistic Instruments in Knowledge Engineering. Elsevier Science Publishers (1992) 131 - 142
8. Dietz, J.: Business Modeling for Business Redesign. In: proceedings of the 27th Hawaii International Conference on System Sciences. IEE Computer Society Press (1994) 723-732
9. Entire Broker. Software AG. URL: http://www.softwareag.com/corporat/solutions/applintegr/default.htm#b1, 1999-11-25
10. Extricity AllianceSeries. Extricity Software. URL: http://www.extricity.com/products/alli_series_over.html, 2000-02-21
11. Georgakopoulos, D., Hornick, M.: An Overview of Workflow management: From Process Modeling to Workflow Automation Infrastructure. In: Distributed and Parallel Databases, 3 (1995) 119-153
12. Green, P., Rosemann, M.: An Ontological Analysis of Integrated Process Modelling. In: proceedings of the 11th International Conference, CaiSE'99. Springer-Verlag, Heidelberg (1999) 225-240
13. Goldkuhl, G.: Generic Business Frameworks and Action Modelling. In: proceedings of conference Language/Action Perspective'96. Springer-Verlag (1996)
14. Hammer, M., Champy, J.: Reengineering the Corporation. A manifesto for Business revolution. New York (1993)
15. HP Changengine Overview. Hewlett Packard Company (1998). URL: http://www.ice.hp.com/ cyc/af/00/101-0110.dir/aovm.pdf, 1999-10-04
16. Jablonski, S., Bussler, C.: Workflow Management. Thomson, London (1996)
17. Lind, M., Goldkuhl, G.: Reconstruction of different Business Processes – A Theory and Method Driven Analysis. In: proceedings of Conference on Language/Action Perspective '97. Veldhoven (1997).
18. Makey, P. (ed): Workflow: Integrating the Enterprise. Butler Group report. Hessle (1996)
19. MQSeries Integrator, IBM. URL: http://www-4.ibm.com/software/ts/mqseries/, 1999-11-25
20. Process Broker architecture for system integration, Homepage of the Process Broker Project (1999). URL: http://www.dsv.su.se/~pajo/arrange/index.html, 1999-11-25
21. Reisig, W.: Petri Nets: an introduction. Springer-Verlag, Berlin (1985)
22. Riempp, G.: Wide Area Workflow Management: Creating Partnership for the 21st Century. Springer-Verlag (1998)
23. Sheer, A.: ARIS-Business Process Modelling. Springer-Verlag, Berlin (1998)
24. Searle, J.R.: Speech Acts – An Essay in the Philosophy of Language. Cambridge University Press (1969)
25. Winograd, T., Flores, F.: Understanding Computers and Cognition: A New Foundation for Design. Ablex, Norwood, N.J. (1986)
26. Yeamans, L.: Enterprise Application Integration. NSAG Inc. URL: http://www.messageq.com/EAI_survival.html, 1999-10-04
27. Wåhlander, C., Nilsson, M., Skoog, A.: Introduction to Business Model Language & SmartSync Model Manager, Copyright Viewlocity (1998)