

Business Modelling for Component Systems with UML

Sandy Tyndale-Biscoe
sandy@open-it.co.uk

Oliver Sims
oliver@open-it.co.uk

Bryan Wood
bryan@open-it.co.uk

Chris Sluman
chris@open-it.co.uk

Open-IT Limited

Abstract

The EC funded COMBINE Project has the objective of dramatically improving software development productivity by providing a holistic approach to component-based development of Enterprise systems. It has a single, cohesive UML based modelling environment for all development stages.

Business modelling is key to user satisfaction, business flexibility and survival. Today, some IT services are so crucial to a business that, without them, things don't just happen less well, they don't happen at all, with disastrous consequences. Business models must do more than provide the starting point for system models – they must become an integral part of a system model, from which code is generated.

“Traditional” business modelling is generally done using modelling concepts and tools that are not optimised to linking to a system model. The COMBINE Project has overcome this problem in two ways – identifying a coherent minimum set of modelling concepts and discovering a way of representing these in UML. This had to be done in a way that allowed models to be equally comprehensible to business experts and system modellers.

The paper describes the COMBINE Team's approach to these problems, covering the following topics:

- *structure of a business model.*
- *a minimum set of modelling concepts.*
- *use of UML “out of the box”.*
- *derivation of a business component model.*

In particular the paper also shows how a business model can be structured to provide the mapping between real world business concepts and software artifacts, which is essential if the full benefits of component based development are to be realised, as it leads to fewer transformations, more robust design and better resilience to business change.

1. Introduction – the COMBINE Project

The COMBINE Project is a research programme, partially funded by the European Commission, that has the objective of dramatically improving software development productivity by providing a holistic approach to

component-based development, integration and evolution of Enterprise systems.

The programme is being conducted jointly by the following eight European based companies: Adaptive Ltd (UK), INESC (Portugal), Iona (Italy), Open-IT Ltd (UK), SINTEF (Norway), Softeam (France), The Open Group (Belgium) and WesternGeco (Norway).

The key product of the programme will be a working “Component Centre”, a development facility for components and component-based systems that provides shorter time to market, greater user satisfaction through more efficient use of development resources and re-use of both application and technical infrastructure artifacts. The Component Centre uses a single, cohesive modelling environment, firmly based on UML and the concepts of the OMG's Model Driven Architecture^{TM1} (MDA), to cover all life cycle stages, from Business Modelling to deployment of working software components. The demonstration facility being developed by the COMBINE Project is based on an industrial strength repository and execution environment, and supported by a state of the art modelling tool.

This paper describes the approach to Business Modelling embodied in the Component Centre and its integration with system (or component) modelling, covering the following topics:

- The business modelling challenge.
- The broad structure of a COMBINE Business Model.
- The modelling concepts used and a conceptual model for them
- How “out of the box” UML can be used to represent the concepts.
- Examples from the COMBINE Programme's work.
- How the “final” step in business modelling is the derivation of a business component model that identifies the coarse grain components to be developed.

Finally, the paper shows how a such a business model can be mapped to a corresponding system model in which there is a clear relationship between real world concepts of importance to the business and the software artifacts (components, and other types) to be developed in the IT system or component. This is essential if the full benefits

¹ Model Driven Architecture is a registered trademark of the Object Management Group Inc.

of component-based development are to be realised, in terms of minimising transformations between models over the life cycle, more robust design and greater resilience to business change.

2. The Business Modelling Challenge

Adequate and correct business modelling, linked to system design, is key not only to user satisfaction with an IT system, but also to business flexibility and, indeed, survival. Increasingly, businesses are totally reliant on IT for fundamental business processes, such as ordering products, manufacturing them and delivering them. Today, some IT services are so crucial to a business that, if they fail, things don't just happen less well, they don't happen at all, and this can be disastrous for the business concerned. The systems that provide such services may be termed "business critical". For them, a business model must do more than provide the starting point for system models – it must be an integral part of that system model from which code is generated, and it must be the ultimate reference to be consulted before any changes to the IT artifacts can be implemented.

It should be noted at this point that this paper is largely about the implications of business modelling for *design* of business critical systems; business models also have undoubted, but different, implications for the *operation* of such systems, particularly where they concern the procedures to be followed by operational and help desk staff. Such issues are outside the scope of this paper.

Such a business model is analogous to the job description for a human member of a business, that defines his or her position in the organisation. It defines responsibilities and functions, and it can act as a vehicle, when things do not go as planned, for identifying cause and, where necessary, blame. No-one would consider neglecting such a document for a key member of staff, yet for some reason it is often not considered necessary where a piece of IT has an equally vital role.

Whilst a job description cannot, of itself, *ensure* that a person behaves as specified, an IT system is supposed to be deterministic, and its job description, i.e. the Business Model that defines it, can and should be used as part of the set of specifications that determine its behaviour.

3. Structure of a COMBINE Business Model

A COMBINE Component Centre produces "Products", which are either whole application systems or specific components. In a COMBINE Component Centre, a Business Model is expressed the part played by a Product being developed in the Centre, in the context of the

business that will fund its development (or purchase or re-use) and use it.

The Business Model is intended to represent goals, business processes, roles, actors, resources and provided services. The scope of the model may be any part of the world defined as interesting for a company, organisation or others, and which has some impact on the required behaviour or other characteristic of the Product. The scope of a business model might be broad or narrow, e.g. describing the entire business of a company or describing the immediate environment and context of a system under consideration. Thus, the business model is recursive in the sense that some business models might detail aspects of another more broadly scoped/ higher-level business model.

A COMBINE business model comprises seven work products. Three of these are relatively informal:

- Context statement
- Vision statement
- Risk analysis

The other four are a set of linked rigorous models expressible in UML:

- Goal model
- Business Process and Role model
- Business Resource model
- Business Component model

Details of each of these work products are given below.

3.1 The Context Statement

The purpose of the context statement is to define the scope of a business model and to position it in its context. This may be done by identifying relationships to some other business models or by developing and getting agreement to a more informal representation of the target business being modelled in its environment. This informal representation takes the form of a domain picture aiming to give an overall understanding of the domain. It focuses on describing stakeholders and their relationships and identifies stakeholders concerns. It typically covers key value-chains and information flows. The emphasis is on the process of building it, agreeing it with subject matter experts and thereby understanding the context. Thus, it may not be maintained in subsequent phases of the development of the Product.

3.2 Vision for Change

This work product is a vision of what to improve, the motivation (i.e. what is wrong with the current situation), a description or indication of what the improvements might be and a gap analysis (a clear understanding of difference between the current and desired situations).

3.3 Risk Analysis

This work product identifies the risks related to a development project for the proposed Product. All types of risk should be considered, including commercial risk (failure to give the intended return on investment), business risk (impact on the business of failure of the Product, either before or after deployment), programme risk (failure to deliver on time), development risk (Product development is more difficult or costly than expected) and support risk (high cost of user support or maintenance because of product fragility). In addition, risks relating to staff implications (users, support staff, management) must also be considered - including operating procedures, redundancy, retraining, morale, re-deployment, new management structures, etc.

3.4 Goal Model

This work product describes the business goals that will be met by implementing and then using the Product. The goal model describes a loose hierarchy of goals of the business within the particular area of concern, starting with the goals of a stakeholder in developing or buying the Product, and leading to the detailed business goals met by the Product or its users when using it.

3.5 Business Process and Roles model

The Business Process and Roles model (hereinafter called the Business Process model) defines the business processes of the domain which are relevant to the Product, and which will enable the goals to be met, and the roles of the resources that perform those processes. This will be at a number of levels of detail, from a high level description of the business processes down to detailed activity specifications for the services that an IT component will provide. It includes a full definition of the roles in the business, including those fulfilled by the system or component to be developed.

In building a Business Process model, each business process is defined in terms of its steps, and each step performed by an actor at a higher level of detail is then treated as a process performed by a community (of actors) and further analysed at a lower level of detail. Eventually, “leaf” steps will be defined. The time to stop analysing is when, by analysing the processes, the “system” role in them is defined, or when there are no more questions about the business to be answered.

3.6 The Business Resource model

The business resource model identifies and defines the main things of interest of the domain that are relevant to

the Product. It defines both those resources that fulfil business roles (the “actors”) and those that are the subject of interactions between actors in the business processes (known as the “artifacts”), and expresses the business significant relationships between them. It includes the constraints on the fulfillment of roles imposed as a consequence of physical structure and organization of the business being modelled.

3.7 The Business Component model

The business component model is a view on the rest of the business model that identifies the coarse-grained components that will make up the product being developed. This is the model that provides the minimal transformation with the system models, which will refine the components identified here.

4. Modelling Concepts

“Traditional” business modelling is generally done using a wide variety of modelling concepts and tools, very few of which are optimised to providing a direct link to a system model. Furthermore, few business modelling tools use the primary system modelling language, UML. Thus the COMBINE Project was faced with two problems – identifying the coherent minimum set of modelling concepts needed for a component model, and discovering a way of representing these concepts in UML, so that derivation of a system model (which has to use UML) from the corresponding business model can be done as simply as possible. In addition, this has to be done in a way that results in models that can be discussed in a comprehensible way with both business experts and system modellers. This section describes the minimum set of concepts that were found to be necessary. It is followed by a description of the use of UML for representing these concepts, an example of preparing a model and a discussion of the process of identifying software components in the Business Model.

Business models are produced for a variety of purposes, ranging from pure “Business Process (Re-)Engineering”, as practiced by Management Consultants, through to the kind of directly implementable model that forms part of the specification of an IT service. Whilst there is some consensus on the kind of things that have to be modelled in a business model, there is a huge variety of concepts that are used to prepare such models, and little general agreement on the meaning of each of these concepts and their relationships with other business modelling concepts.

This situation is unacceptable for a rigorous, deterministic model of an IT component’s behaviour and its position in the organisation that owns and uses it. The

COMBINE Project therefore identified a minimum subset of concepts necessary to specify the following:

- an IT component's required behaviour within the context of the business that it supports,
- the constraints on that behaviour that arise from the business organisation's aims and culture, and those of the larger, legal, framework within which it operates.

Having identified this minimum set, it was necessary to define them rigorously and express the relationships between them, in a manner that ensured clarity of models and minimised (and, preferably, eliminated) duplication.

The set of concepts that was selected is based on those of the enterprise language of RM-ODP (see [1]), adapted to map well to commonly used business modelling terms. These are described below, and illustrated as a conceptual model (or, for those inclined to use the term, metamodel) in Figure 1.

The business is described in terms of the goals of the business, the things that happen in the business, the roles in the business needed to make those things happen, and the resources fulfilling roles. The key concepts used are described below.

A **Community** is a collection of resources formed to meet a set of goals. Thus, a community has a set of members, which are resources, an objective, which is defined by one or more goals, and behaviour, which achieves the goals..

A **Goal** is some desired state of affairs or aim of the business being modelled (or of some part of it). A goal is described in terms of a loose goal hierarchy, where the achievement of each goal can be considered in terms of achievement of a set of sub-goals. The sub-goals related to a goal can be combined with predicates to express conditions that govern success of their parent goal.

The goals of a community are achieved by its **Enabling Behaviour**. This may be expressed in two ways:

- as a business process;
- as a behavioural policy.

A **Business Process** is a behaviour of the business which may be modelled as consisting of a defined start point, a set of steps, and one or more defined end points. Business processes may generate events and may be affected by events. A number of roles are associated with each business process, which describe the behaviour of the resources that take some part in the business process.

A **Step** is defined as something which happens in the business. A step is performed by a resource that is an actor in that process (it may be a composite resource, but at this level it is considered as a single one); a step may concern one or more resources that are artifacts in that business process. A step can itself be modelled as business process executed by a community modelling a resource, and thus considered at a greater level of detail.

A special kind of step is an **Event**, which is generated by a business process and may affect other business

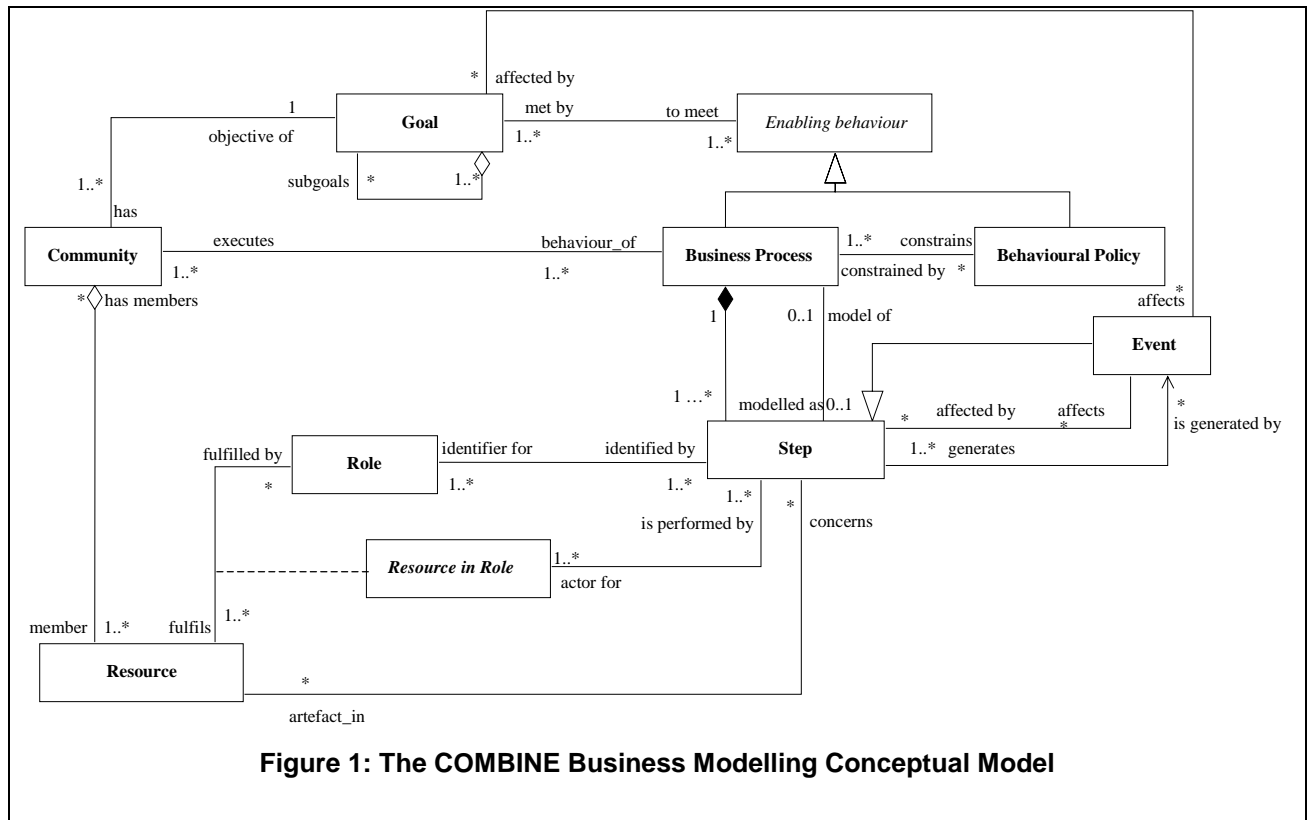


Figure 1: The COMBINE Business Modelling Conceptual Model

processes. An event may have an effect on a business goal.

A **Behavioural Policy** is a behaviour of the business (something which happens) which, for any reason, it is not appropriate to model as consisting of a defined start point, a set of steps, and a number of defined end points. It is, nonetheless a behaviour which enables the achievement of a goal. For example, “Ensure Customer Satisfaction” may be a Goal that is enabled by “Answer all calls within 30 seconds”. It would probably not be appropriate to represent this behaviour as a business process. In general, these behaviours are constraints on one or more behaviours which are expressed as business processes. Thus the model takes this form: We do *this* (business process), in *this* manner (behavioural policy).

A **Role** is an identifier for a behaviour. A role has a well-defined responsibility, defined through the set of steps in one or more business processes in which the role is involved. The behaviour of a role is defined in terms of its related steps. Roles are related to the resources that fulfil them.

A **Resource** represents a thing of interest to the Business that plays some part in the business, either as an *actor for* or as an *artifact in* some behaviour. An identified business actor, which may be one or more human beings or machines or groups of the two, is responsible for each step in a business process. Work products are examples of artifact resources, which typically are input to and output from business activities.

An essential facility for business modelling is the ability, in a single model, to represent things and their behaviour at more than one level of detail. Generally, business modelling techniques offer facilities to do this, but only for actual things or for actual behaviour – not in combination. What is frequently needed is a means to take some piece of (composite) behaviour, and the (also composite) thing that exhibits it, and decompose that combination. In the COMBINE Business Modelling technique this is done using the derived concept **Resource in Role**, which represents both a behaviour (steps) and the thing that exhibits that behaviour (resource). An instance of this concept can then be considered as a community with goals, enabling behaviour and resources.

5. Using UML

Having a good conceptual model for business modelling is one thing; representing the concepts concerned using UML (the UML metamodel, its model management facilities, and, where appropriate, the UML notation) is quite a different matter. UML offers an extremely rich set of concepts, but they are optimised to specifying the computational and engineering aspects of a

system, and particularly towards the generation (or at least documentation) of code. They are certainly not optimised towards modelling the generally rather more abstract concepts of a business.

The key features of the approach to using UML are:

- use of a UML Profile defining stereotypes of UML metaclasses to represent the business modelling concepts described above, and
- structuring and model management practices that enable a business model to be prepared and used.

5.1 A UML Profile for Business Modelling

The COMBINE Business Modelling Profile, in its initial form, uses stereotypes, and icons where appropriate, to model the concepts described in Section 4 above. The Context Statement, the Vision Statement and the Risk Analysis are all represented as attached documents, with UML packages linked to them. It would be simple to stereotype these to represent the particular type of document concerned, but this has not been seen as necessary.

Table 1: Summary of Stereotypes

Business Modelling Concept	Base Metaclass	Icon
Community	Package	
Goal	Class	
Business Process	Class	
Behavioural Policy	Class	None yet
Step	ActionState	
Process <i>achieves</i> Goal	Association	None
Resource in Role	ActivityGraph:: Partition	
Resource	Class	
Resource as Artifact	ObjectFlowState	

The Goal model is contained in its own package in which goals and sub-goals are modelled as classes stereotyped as «Goal», with normal aggregation associations to represent relationships between goals and sub-goals.

Community is key concept of both the Business Process and Role model, and the Resource model, which collectively represent the things that happen in a business and how the resources of the business are organised to make those things happen. Whilst the concept of Community does indeed have semantics (a collection of objects formed to meet an objective – mapping well to the

natural concept of Organisation), these semantics are almost completely expressed in the business process model and the resource model that are associated with Community. Thus the primary purpose of the concept of Community is to collect other model elements (representing business processes, roles and resources) in some context, and the UML metaclass Package, stereotyped as «Community», is used to represent it.

Within the specification of a Community, the main model elements are the business processes that achieve the Community's objective and the resources that have the behaviour necessary to perform those business processes. These are modelled with classes, stereotyped as «Business

In the second case, where the need is to rigorously represent the sequence of steps in the process and the possible paths through it, an ActivityGraph must be used. This will represent each Resource in Role as a Partition stereotyped as «ResourceInRole». Within each partition the Steps of the Process that the role performs will appear as ActionStates stereotyped as «Step», and the Resources involved as artifacts in the step (i.e. information flows associated with the step) will appear as ObjectFlows, stereotyped as «Resource».

This ActivityGraph representation of the details of a business process has intrinsic limitations imposed by the UML1.4 metamodel (which may be resolved in UML2.0),

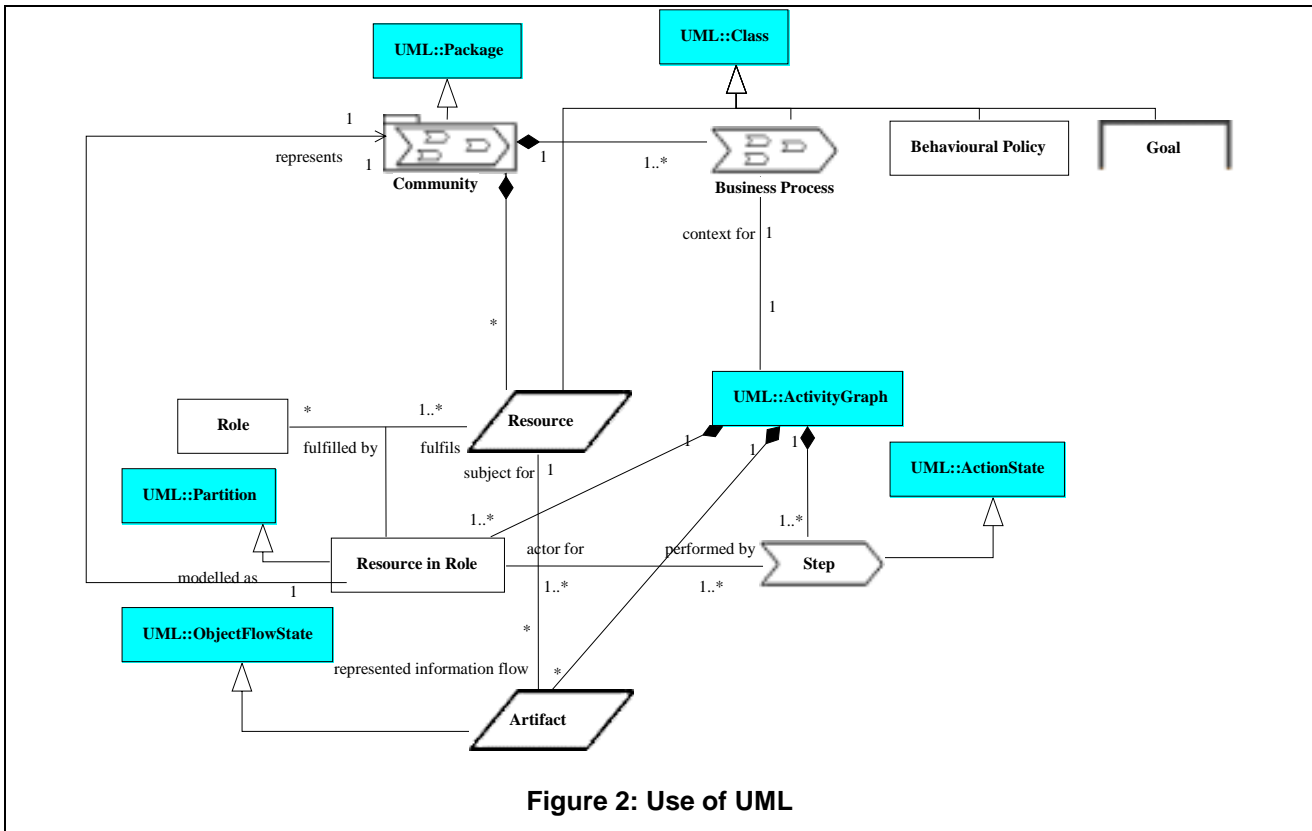


Figure 2: Use of UML

Process» and «Resource» respectively.

Business processes may be detailed in one or both of two ways. In the first case, where the intention is merely to represent the overall structure of a business process and the resources involved, then it may be detailed as a number of sub-processes, each modelled with a class stereotyped as «Business Process», with aggregation associations used to model the structure, and regular associations, stereotyped as «performs» or «uses», to represent the relationships between the business processes and the resources that either perform them or are involved in their execution. A class diagram is then used to represent this structure.

but it is necessary if the logic of the process is to be modelled. It is also highly desirable if the steps in the process are to be properly decomposed. This is because the key element to be represented at greater detail is the concept Resource in Role – not just the Resource, note, nor just the Role. This can only be modelled practically by a Partition.

The approach used to achieve this decomposition is one of the crucial innovations in this business modelling method. The technique is to create a new package that maps to the partition representing a Resource in Role, stereotype it as «Community» and create a link in the model between the two. Detailing of the behaviour then

proceeds as before, with a contained process model that represents lower level business processes (which are steps in the higher level model) and a contained resource model that represents the resources that are local to the community concerned.

5.2 Structuring the Model

Use of UML for doing this sort of modelling is not simple, and, without some fairly strict structuring rules, a model produced in this fashion would not mean much except to the modellers who generated it, and, after a while, not much to them.

As already stated, COMBINE Business models are structured around the concept of Community, with, inside the container that models each community, model elements representing business processes, roles and resources relevant to the community concerned. Within this overall structure, the actual structure used will vary from project to project, depending somewhat on modelling choices, but mostly on two things: the needs for presentation and the route by which the model was generated.

Figure 3 shows an example of a typical generic model structure. The package “Project” contains the entire Platform Independent Model (see MDA), of which the Business model forms a part. Two other parts of the PIM are shown, the “User Requirements Model”, which uses a use case based approach to model user interaction dialogues, and the “Architecture Model” which models the detailed component structure. There may be other elements of the PIM as well, but all these are outside the scope of this paper.

Within the business model we see eight packages. The first three are containers for the documents that provide the Context Statement, the Vision Statement and the Risk Analysis.

The goal model is contained within a single package labelled, helpfully, “Goal Model”. This package will have dependency links to packages containing the Business Process model so that the «achieves» associations between business processes and goal can be represented.

The next package is a «Community» that is a container for the top level process and resource models, which identify, and provide the first level of detail of, the processes visible at the top level, and the resources that are actors or artifacts in those processes. There may be an activity graph that represents the sequence of flow between these processes.

The next package is a container for all the second level communities, each of which represents some top level resource and its role in the top level processes. Thus, suppose we identified Processes 1 and 2, in which Resources A and B participate, then at the second level we

might, (provided we were interested in decomposing both processes and both resources) have communities for all four of those combinations.

Similarly, there are packages for all third level communities, and further packages for decompositions at a greater level, to the extent necessary to define the “system” role in the business, or to answer all questions about the business.

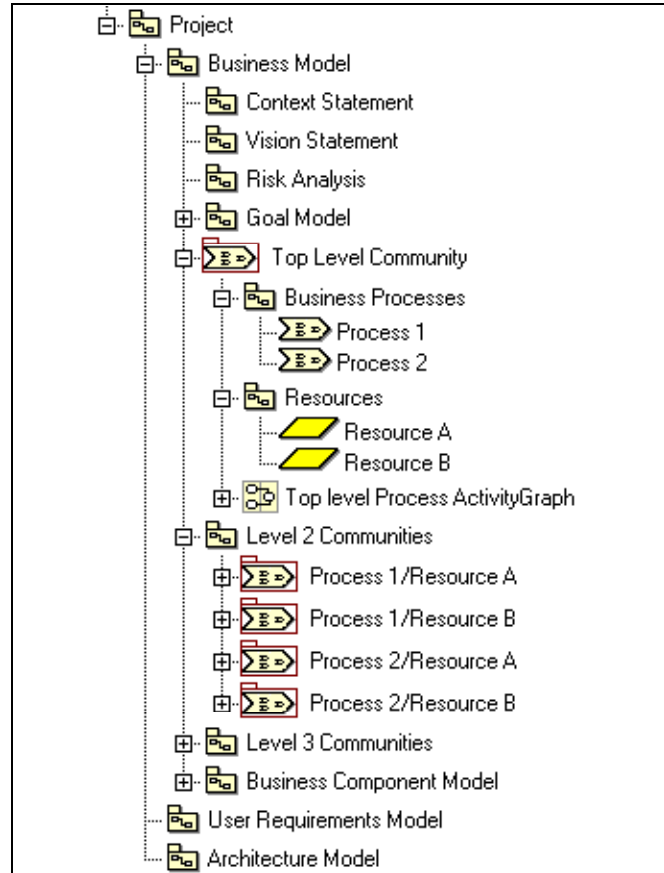


Figure 3: Typical Model Structure

6. Example – from the COMBINE Test Case

To validate the techniques being developed by the COMBINE Programme we have a test case that is a live application of the user partner, WesternGeco, that has taken a business model through to actual component development and implementation. This section shows some examples of the business model that was produced.

WesternGeco’s business is the conduct of geological surveys in search of oil, primarily at sea, using survey ships. The test case concerns the development of an IT component known as the “Survey Booking Tool”, which will assist in the management of surveying resources. The

overall model structure is shown in Figure 4. It will be seen that it matches to suggested structure described in Section 5.2 above. In this case the top level community is called “Marine Operations”, and there are a number of second level communities, chief of which at this point in the model’s development is the “Marine acquisition community”.

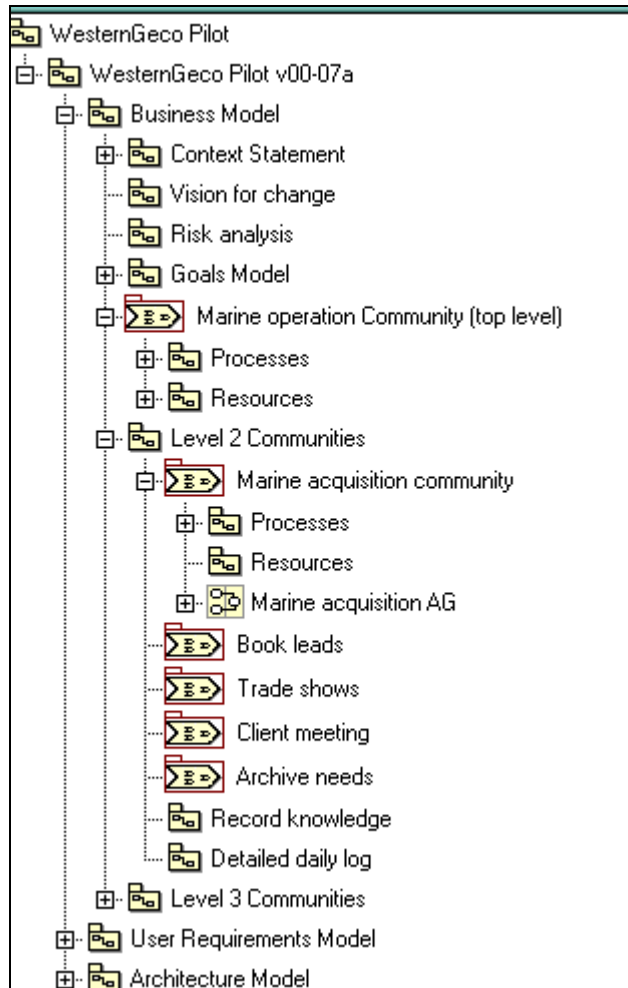


Figure 4: WesternGeco Model – Top Level Structure

6.1 Goal Model

Figure 5 shows an extract from the modelling tool’s explorer pane listing all the goals, identified through discussions with the business stakeholders. Figure 6 shows, for one particular goal, “Win Profitable Bids”, the goal hierarchy within which it fits, and the enabling processes that are required for it to be achieved. This latter element provides the link forward to, and traceability back from, the Process model.

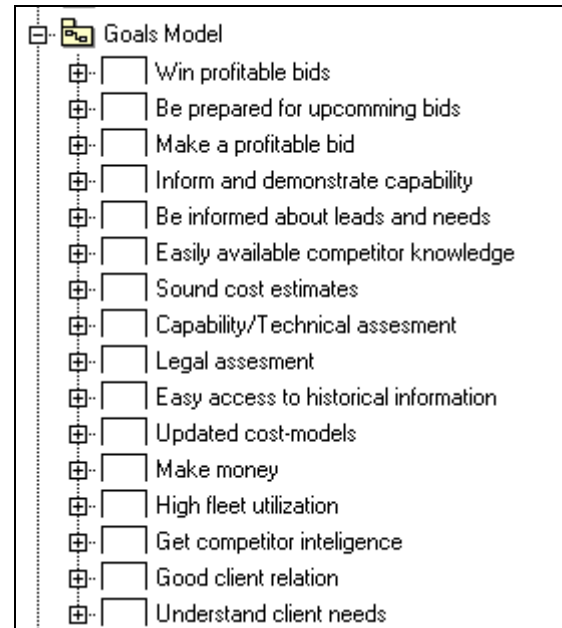


Figure 5: Full list of Goals

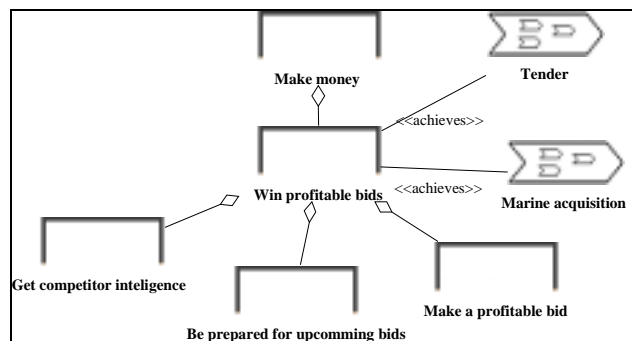


Figure 6: Example Goal Hierarchy and associated Processes

6.2 Process and Role Model

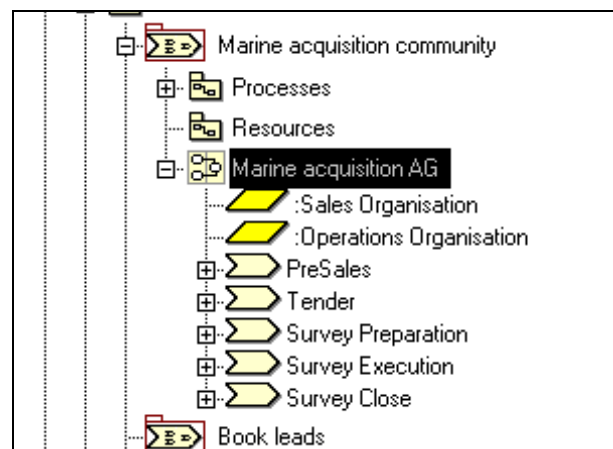


Figure 7: Marine Acquisition Process

Figure 7 and Figure 8 show the explorer pane and a graphical representation respectively from the Process model at the second level, illustrating the breakdown of the Marine Acquisition Process, one of the enabling processes for the goal “Win Profitable Bids”.

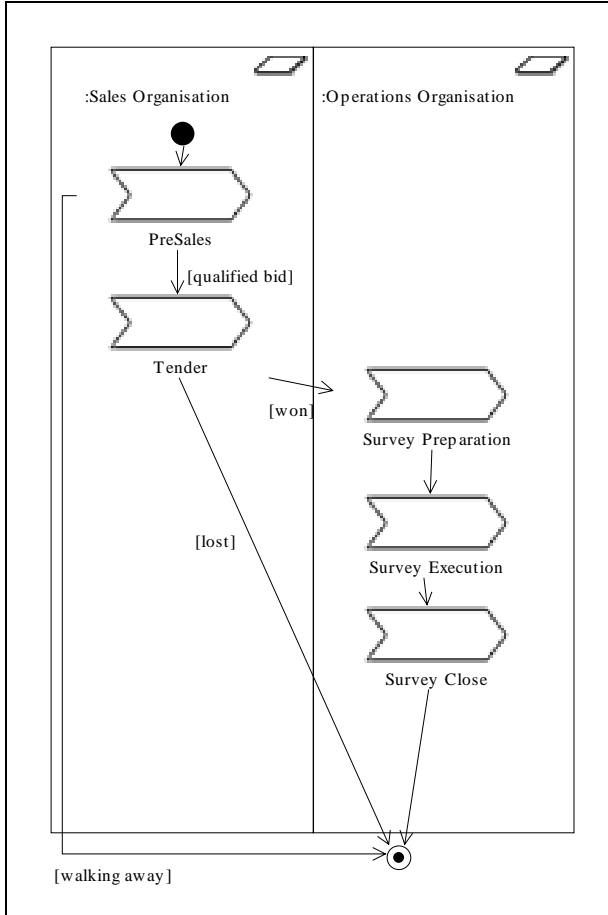


Figure 8: Marine Acquisition Process Activity Graph

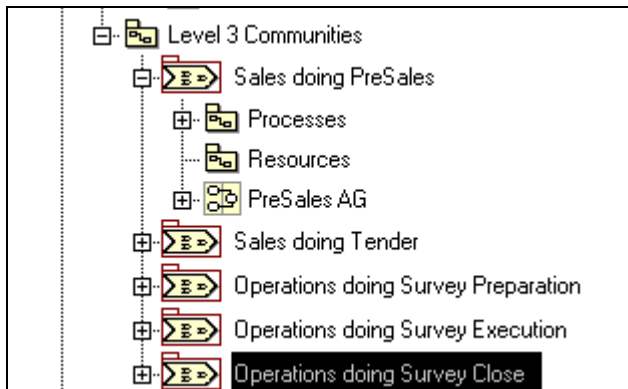


Figure 9: Level 3 breakdown

Figure 9 shows a further breakdown, showing how the steps identified in the Marine Acquisition Process, PreSales, Tender, Survey Preparation, Survey Execution and Survey Close, are each detailed as a separate community. Note that it is assumed that all these processes involve the Survey Booking Tool; in other words, the Survey Booking Tool will be a member of each of these communities.

6.3 Resource Model

Figure 10 shows an extract from the high level resource model. Note that it contains resources that may be both actors and artifacts.

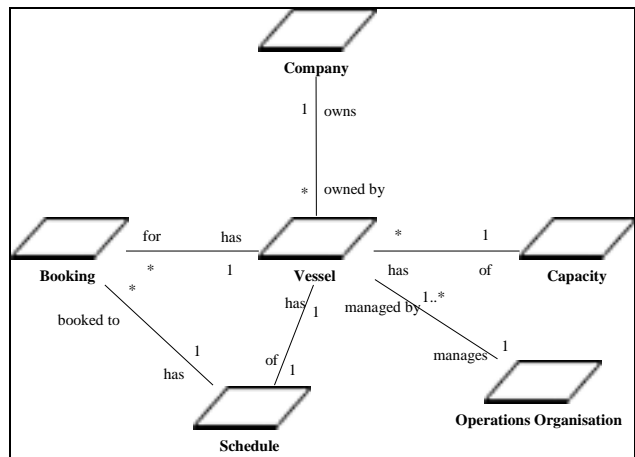


Figure 10: Extract from Resource Model

7. Component Identification

Business is carried out through processes performed by actor resources in defined roles, using artifact resources. The UML model defining desired business behaviour can structure the roles of the IT system in many ways. If there is a way of structuring the model such that designing the computer system (for whose production the business modelling effort was funded) is made easier, then such a way is clearly superior.

The key is to identify the Business Components (“components of the business”) within the business model. These components are candidates for software components in subsequent software design work. The resultant structure therefore requires no transformation when input to a Component-Based Development (CBD) software design process, and so little or no information is lost in the process.

(Note that the approach presented here is oriented towards a particular component metamodel; however we believe the general principles to be valid for several approaches to component-based development.)

The objective is to identify candidates for process and entity components that will be programmed by IT staff.

First we give an extremely high-level overview of CBD. Then we address identification of Business Processes and Business Entities. Finally we show how the “application”—a very-large-grained software component, is identified.

7.1 Component-Based Development

CBD seeks to represent “chunks” of the business function as “chunks” of software that fulfil that function. Such a software “chunk” is represented in the Business Model as a Business Component. A Business Component is not only a business-oriented part of the business, but it is also a candidate for implementation as an IT component that encapsulates a piece of business function, provides interfaces to other components, requires interfaces of other components, and is developed as autonomously as possible.

Software components have a great deal in common with software classes in Object-Oriented Programming, but instead of being used to build a program, they are themselves a specific kind of program that is built using some programming language. Since they are autonomous (but not independent), and are intended to represent “chunks of business function”, CBD aims to identify the component types very early in the development lifecycle. Thus each software component “program” implements a specific business function type, either process or entity. It is this that makes software components quite different from OOP classes and instances, and makes CBD quite different from more traditional forms of development.

Business Components can be entity components or process components

7.2 Entity Business Components

An entity business component (also called just “entity component” in this paper) encapsulates a group of business artifact resources. The first step in identifying entity components in the Business Resource Model is to consider each Artifact Resource, and to ask whether it is “real” and “independent”.

- **Real:** means that the concept is both used and well understood by subject matter experts (SMEs). It is not abstract. A “real” concept is a type whose instances are actually used in the business, as opposed to an abstraction of that type. For example, in a manufacturing business, “customer,” “address,” and “invoice line item” would probably be real, while “legal entity,” “location,” and “collection member” might not. That is, an SME would assert that while a “customer” is a common

everyday concrete thing, a “legal entity” is not (although there might be agreement that, hypothetically, it would be a good super-type of “customer”). (For a full discussion of this, see under Trading Partners on p.463 of [2].)

- **Independent:** An independent concept is one that can be talked about by SMEs without first saying to what it belongs. That is, its scope is implied and understood, and it is probably that of the business or of a major division within the business. For example, in a manufacturing business, a “customer” probably does not have to be qualified, so it’s independent. “Address” or “Balance”, on the other hand, would have to be qualified to be meaningful, for example, “customer address” or “supplier address.” Again, this is context-dependent; for example, often addresses are kept separate from the entities they relate to (especially in distribution-oriented organizations). Often such organizations identify an “Address List” or “Address Book” business component

It is important to avoid confusing user interface constructs with the core enterprise business concepts. For example (and still thinking of a manufacturing business) order-entry people might well like the facility to pick up an order line and drag it to another order to avoid re-entering data. At first sight, this might suggest that an order line is real and independent. However, such a requirement is actually about a desired user interaction within the “user model”. This has a single user as its context, and has nothing to do with the business model, which has the enterprise as its context. This thus the two models are not the same, although there may well be striking similarities.

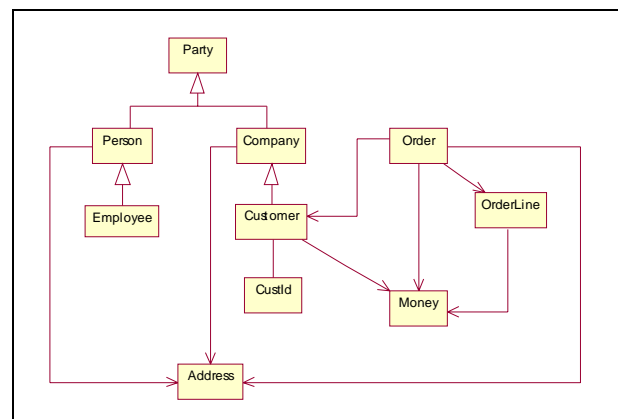


Figure 11: Sample Artifact Resource Model

Continuing with the example, a business model will normally see an order line as being within an Order entity component, an hence would be real but dependent.

Looking at the Business Resource Model in this way, certain types can be identified as real and independent. These are called “focal” types. Other types are called “auxiliary” types. (These terms follow UML 1.4 semantics, except that they are applied in the business model.) For example, consider the following fragment of a (simplified) business artifact resource model.

Suppose that Employee, Customer, and Order are identified as real and independent. These are “focal types”. Others are “auxiliary types”. Now follow the relationship and specialization lines from each, stopping only when you come to another focal type. This results in a set of auxiliary types, plus the focal type, and is said to comprise a “focal group”. If an auxiliary type appears in more than one focal group (e.g. Party, Address, Money, and OrderLine), then perform a normalization exercise. The result is shown in Figure 12.

Now, separating these out, we arrive at the desired view of the entity components, as shown in Figure 13.

As can be seen, a focal group is the set of types that make up the entity component. An entity component typically takes its name from the focal type, which may then be re-named (for example to “CustomerFocus”).

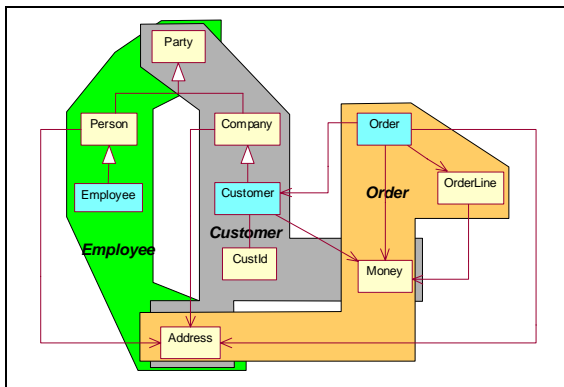


Figure 12: Focal groups

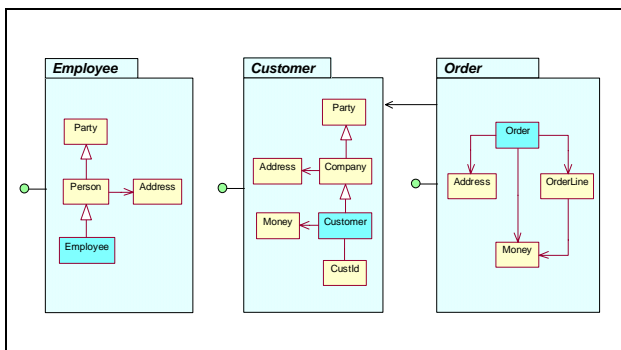


Figure 13: Entity Components

7.3 Process Business Components

Implementation of process business components (also known as just “process components” in this paper) can be through a number of different vehicles, including service-oriented programmed technical components (e.g. as realised by EJBs), workflow specifications, and B2B collaboration specifications. The question is, can we structure the business model so as to identify candidates for these different implementation vehicles, all of which typically exist and complement each other in real systems? The answer is yes. However, since space prevents full exposition here, we have chosen to focus mainly on the service-oriented programmed component. So back to the business model.

A number of the processes defined in the Business Process Model will have steps performed by (a role of) the IT system. These can be categorized as being either “atomic” or “compound”:

- Atomic: the process is one where either all steps are completed, or the state of the business should be left unchanged. That is, the business is not interested in intermediate states, and would prefer not to have such states recorded anywhere.
- Compound: the process has a discrete series of steps, each of which may or may not itself be atomic, and each separated by a time lag that is either necessary and/or desired.

There are two important constraints on such categorisation. First, business atomicity must observe real-world constraints. For example, a business may desire payment by a customer to its bank account as soon as an invoice is issued. However, many customers will insist on payment terms that impose some delay in payment.

Second, user interaction design considerations must be absent from the process. Although good user interaction design is of great importance, it does not play a part in the Business Process and Role Model. For example, a process might state:

<i>Person Role</i>	<i>System Role</i>
Enter Order Header information	Record the Order Header
Enter Order Detail information	Record the Order Details

If there is no business need for a delay between *Person Role*’s steps, then these steps are merely reflecting some user interaction sequence. The two steps can be validly compressed into a single step, where the System Role simply “records the Order”.

Processes performed wholly or partly by the IT System are candidates for implementation as process components. Following the categorisation just presented, these can be defined as being:

- “atomic” process components, which are candidates for implementation using some programming language and which will run in a transactional system environment providing ACIDity.
- “compound” process components, which are choreographed by some multi-transactional implementation such as workflow or B2B, or by the user driving the process on his or her workstation through multi-transactional long-running processes defined in the user model.

This paper presents a heuristic for identifying the former category, which forms the heart of a service-oriented IT system. As mentioned previously, discussion of the later category must, due to space constraints, be deemed outside the scope of this paper.

The identification heuristic for atomic business process components is the only part of the component identification process that requires some minor transformation. However, it is a transformation entirely on business terms, and in itself has nothing to do with specific IT implementation—other than that the whole reason for doing it is to make implementation represent better the business structures. The heuristic is as follows.

First, we examine the atomic process components identified as described above. If each is described in verb-noun form, it will be found that many of the artifact resources identified by the nouns are entity components such as Supplier, Contract, Invoice, and Product (as discussed in the previous section).

Next, we group the processes by entity component. It is likely that each group will then describe the CRUD lifecycle of the entity component. Many groups will involve more than one entity component. For example, the Sales Order group could include process components that not only direct the life cycle of a Sales Order in some way, but will also update Inventory, and also Customer Balance. However, the focus of the process will be the Sales Order.

Each such group can be seen as a process component, which makes use of other process and/or entity components. For example, suppose the following atomic processes were among those defined in the Business Process Model:

- Change Customer
- Place Order
- Employee Leaves
- Create Customer
- Update Order
- Cancel Order
- Hire New Employee

Following the grouping described above, the following candidate process components would be identified:

- CustomerProcess:
 - Change
 - Create
 - ...
- OrderProcess:
 - Create (place order)
 - Update
 - Delete (cancel order)
 - ...
- EmployeeProcess:
 - Create (hire new)
 - Delete (employee leaves)
 - ...

The various verbs represent candidate operations in the component’s interfaces.

In this way, it is possible to have the Business Model provide, in business terms, candidates for process components that not only provide the basis for a service-oriented enterprise system (excluding user interface elements), but also exhibit best practice modularisation along high cohesion low coupling principals.

7.4 Identifying Application Components

The identification and grouping heuristic described in the previous section will result in a small number of “top-level” atomic process business components. Each will tend to map to a specific organisation within the business (reflecting the fact that the business model is being built to address the needs of one or more of those organisations). In addition, each top-level process will be at the top of a component composition hierarchy (assuming use of the mediator pattern to minimise dependencies, which is a desirable feature in CBD). The other components will be lower-level process components and the entity components that they use. Figure 14 shows a simplified example, where “Order Process” composes one other atomic process components and three entity components.

This composition is effectively an “application”. The COMBINE architectural metamodel has the concept of an “Application Component” which, aside from the user interaction specifications that are also included, maps to the pattern illustrated by Figure 14. Thus the Business Model can also define business-oriented assemblies that not only make business sense to the sponsoring organisations, but also enable identification of very-large-grained application components.

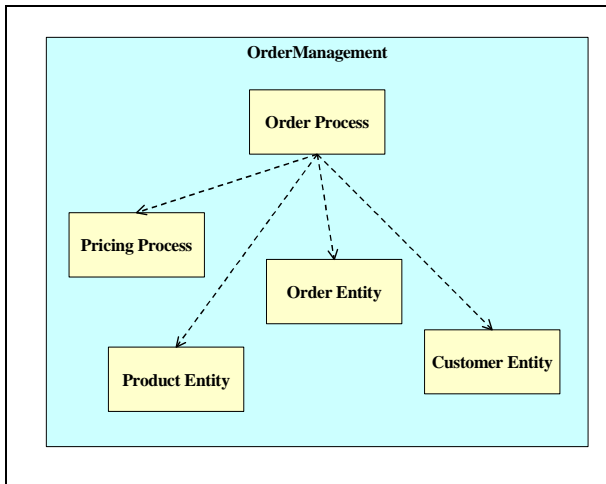


Figure 14: Application Component

8. Conclusions

This paper summarises the demonstration by the COMBINE Programme of the benefits and feasibility of making business models an integral part of system models for component based development, and using UML1.4, effectively “out of the box” to do so. We have identified a sufficient and internally consistent set of modelling concepts to be used, and related them to UML. We have shown how such models can be used as an essential step in the high level design that identifies the IT components to be developed, and demonstrated that they can be used as part of the specification for these components, to assist with re-use, and to provide the all-important link between the operation of the business and the design of the IT elements that make that operation happen.

The validity of identifying components in the business model is based on three assumptions:

- It is desirable that the structure of the business should be reflected as tightly as possible in the structure of the system.

- Maintenance of the business structure in terms of process, entity, and business-oriented assemblies of both leads, *ceteris paribus*, to minimal transformations, and hence to intrinsically higher quality outcomes
- Identification of business components is not a technology thing, it is a business thing, and hence should be done in the business model, where other business concepts are identified.

Component technology provides the technical means for delivering the “components of the system” that are as close as possible in structure and function to the “components of the business”. Good CBD processes and architectural concepts make up the glue that links business modelling and technical component delivery into the run-time.

9. Acknowledgements

The authors wish to acknowledge the work done by other COMBINE Partners, particularly those in SINTEF and INESC, that has contributed significantly to the ideas and methods described in this paper.

10. References

- [1] ISO/IEC & ITU-T: Open Distributed Processing – Enterprise Language. ITU-T Rec. X.911 | ISO/IEC 15414 (under development)
- [2] Peter Herzum & Oliver Sims: Business Component Factory, Wiley Computer Publishing, 1999, ISBN 0-471-32760-3
- [3] Oliver Sims: Business Objects, McGraw-Hill 1994, ISBN 0-07-707957-4