Beyond Use Cases

Graham McLeod
University of Cape Town, Pvt Bag Rondebosch, 7700, South Africa
mcleod@iafrica.com, tel +27 21 650 4028, cell +27 82 578 1834

Abstract

UML is widely accepted as a standard. It is competent in system level modeling and static structure analysis but lacks constructs for enterprise and business process modeling: areas which are vital for a competent technique: especially given the trend to use object technology to support enterprise engineering and implement value chains transcending organizational boundaries.

The author previously extended the Martin/Odell OOA/D methods to incorporate stakeholder, enterprise, value chain and business modeling techniques. In 1998, equivalent extensions were proposed to UML to cope with enterprise and business process modeling. The approach has since been refined through experience and has proven effective in supporting sustainable business engineering. This paper reports the results of this experience and proposes the replacement of Use Cases and other dynamic models within UML with process models which transition seamlessly from stakeholder to design level. The suggested notation represents a Use Case and Activity Diagram superset.

Need for Competent Process Modeling

Organizations face increasing pressure to deliver goods and services with higher quality, more cheaply and more quickly. They must innovate and bring new offerings to market at a rapid rate. These challenges require processes which are quicker, more effective and cheaper - i.e. consume less resources. [Taylor, 1994]. The transition to e-commerce has necessitated a rethinking of fundamental business models [Seybold & Marshak, 1998]. Many organizations are concentrating on core competencies, outsourcing a variety of other functions. These challenges mean that competent techniques are vital to assist analysts in enterprise and business process modeling, and in the evaluation of competing alternatives. It is also imperative to identify opportunities for leveraging technology in support of more effective processes, and then to transition the revised business process models into system requirements and designs with minimal effort and delay while preserving maximum fidelity.

Object Methods Appear Desirable

Object technology offers many potential benefits, including richer modeling, increased reliability, easier integration and reduced costs and implementation times due to reuse [Taylor, 1994]. Organizations thus seek to use object oriented analysis, design techniques and technology to meet the challenges mentioned previously. When choosing methods, they are very likely to adopt the Unified Modeling Language (UML) sanctioned by the OMG [OMG, 1999]. This occurs for a variety of reasons, including the following: UML is “standard”; it is widely taught and publicised; it is rich; it is well supported by a variety of CASE tools; it promises to provide all the necessary techniques in one integrated package. All of these are essentially true, with the possible exception of the last reason, with which we will take issue.

UML Coverage

The UML specifies a notation for analysis and design models. It does not prescribe a method or process for how these should be used. UML has powerful techniques for expressing many facets during the analysis and design stages of a project including:

Representation of	UML Artefact
Domain knowledge required to support the application	Static Structure/Class Diagram
Interaction of a user with a system to achieve a single goal	Use Case
Sequence of collaboration between objects	Sequence Diagram
Responsibilities for dynamic behaviour within the system	Collaboration Diagram
Changes in state of an object over time	State Diagram
Detailed sequence and dependency of activities	Activity Diagram
Specific implementation choices	Component and Deployment Diagrams

UML is thus richly endowed with techniques to express the results of systems-level modeling and design choices. There is little, however, to express high level views of the organization or processes and to assist a business engineer in choosing among competing alternative processes or business models. [Jacobson, Ericsson and Jacobson, 94] have described the use of use cases in support of business process engineering, but the approach still lacks the semantics to facilitate the above fully.

Shortcomings of UML

UML may be *too* rich. An analysis of UML 1.1 by [Xavier Castellani, 1998] revealed a total of 233 discrete concepts. This can lead to difficulties in teaching it, learning it and applying it in projects. We have observed in our work in industry, that many projects enter an extended phase of use case modeling but flounder when attempting to translate these into detail requirements and rigorous specifications necessary for sound design.

Other authors, including [Anthony Simons, 1999; Brian Henderson-Sellers, 1999] have pointed out that there are many inconsistencies and gaps in the semantics of the UML.

In our opinion, there are some fundamental problems in using use cases for enterprise and business process modeling in support of enterprise engineering:

- A system centric view is encouraged too early
- There is no adequate means to show what is computerised or manual in a complete business process
- There is no sense of flow, dependencies and probability of pursuing various paths
- There is no accounting for costs and resource consumption
- There is inadequate definition of inputs and outputs
- There is no mapping to organizational responsibilities, or geographic locations
- There is no means to express timing or synchronisation
- It is difficult to show parallelism explicitly
- There is no verification of the domain model
- Without consulting narratives, it is difficult to determine which steps are optional and which comprise the “normal case”

Requirements for a Competent Approach

We developed a list of criteria [McLeod, 1999] for a competent approach. The technique should:

- Use same notation for current and future models to allow direct comparison of alternatives

- Show complete business process including manual, partially computerised and fully automated processes
- Indicate how process is initiated
- Identify people and organisational groups involved
- Show sequence and dependencies of activities based upon state of business objects affected
- Allow synchronous, asynchronous and parallel activities
- Validate and link to business object domain model (class diagram)
- Show possible outcomes which can trigger other steps
- Allow steps to be other embedded processes
- Be instrumented to allow analysis and improvement

In addition, there were a number of desirable features, including:

- Models should be usable with users, analysts and designers (and in JAD sessions)
- Familiar techniques with wide support in industry to ensure CASE tool availability
- Object orientation
- Easy transition to rigorous design- level models
- Ability to show location, responsibility
- Ability to link resources, specify constraints, calculate costs and timing

Towards a Solution

From 1992-1997, the author and colleagues evolved an approach based upon the Martin/Odell OOA/D techniques and notation, but incorporating many innovations from the work of [Ulrich Frank, 1994] at the *Geselschaft fur Matematik und Datenverarbeitung* (GMD); [David Taylor, 1994] and various Business Reengineering authors [Hammer, 1990, Porter, 1985, Jacobson *et al*, 1994]. This provided a mapping from stakeholder models, via value chain identification, to business process models, and finally event models. Rather than being discrete, these models evolved using a common notation, enhanced at various levels of detail to add more fidelity and rigour as analysis proceeds.

When UML was adopted as a standard, we sought to express our models in a UML-friendly notation. In 1998, we presented a notation which extemded the (then new) activity diagrams within UML to accomplish our purposes [McLeod, 1998]. We chose activity diagrams as being the closest to the techniques we had used previously. This is not surprising, since the UML activity diagrams draw heavily on the Odell event modeling technique.

Recent Work - CASE and Repository Support, Tighter Linking to Other Models

Since the 1998 paper, we have used the revised techniques in teaching, consulting and projects. A supporting CASE tool was built using Visio™ as a graphical engine, but with a customised repository to support the necessary properties. This necessitated a formal analysis of the meta-model to support the process models. The theoretical base of the techniques was enhanced and the fuller notation presented at the Tools Europe workshop in June 1999 [McLeod, 1999].

Context Provided by Architectures

Over the past three years, we have evolved a comprehensive set of architectural frameworks which cover business, information, application and technology infrastructure facets. An element of the business architecture is business processes. We consulted these models and integrated some further high level properties required by this strategic and architecture view into our business process models. The refinements since the 1998 paper, together with this, more strategic perspective, provide a notation and associated properties that is rich enough to support process modeling from a very high level (stakeholder view), through business engineering and process improvement, all the way down to system activity models which are at least as rigorous as a UML sequence diagram. A major benefit is the seamless transition from high level to detailed views without loss of fidelity.

The process architecture is shown below:

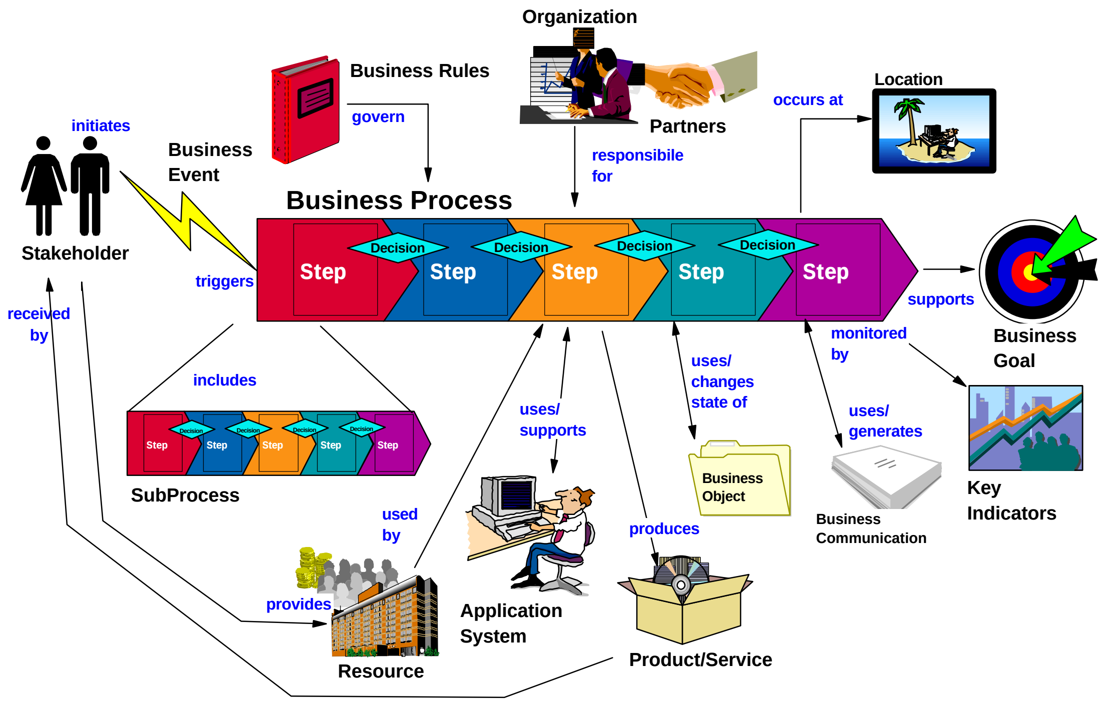


Figure.1 - Inspired Frameworks© Process Architecture

In this picture, we can see the following elements:

- The **business process** is seen as a sequence of steps and decisions. It may include sub-processes as steps
- The process is triggered by a **stakeholder** (which is anyone who interacts with the enterprise) initiating a business event. Examples would be requesting a new account, ordering a product, requesting payment for goods provided..
- The business process is governed by **rules and policies** which the enterprise may establish, or which may be mandated by law
- Various parties (internal **organizational units** and **external partners**) are responsible for the performance of particular steps
- Steps in the process will occur at a variety of **locations** (physical or organizational)
- The process should support one or more expressed **business goals**
- The process effectiveness can be measured by one or more **key indicators**
- Various **business communications** may be produced as a result of the process e.g. Statements, contracts, letters etc.
- The process may reference or update the state of various **business objects**, such as customer, product and agent information
- The process will usually produce a **product**, a **service**, or combinations of these
- Parts of the process are likely to use or be supported by **application systems**
- The process is likely to consume **resources**, such as staff time, money, raw materials, etc.

- We would like to know the time that the process may take (current/desired as well as minimum/average and maximum), the costs of performing the process and the frequency of occurrence per unit time. These values may only be available later from more detailed modeling

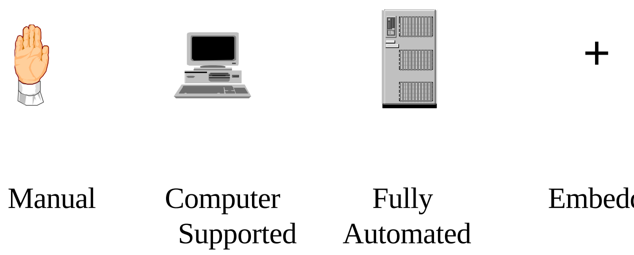
This framework provides a powerful model for discussing the relevant business processes with business executives. It is non-technical and allows us to focus on the essential issues related to the process, the objectives and the context of the process, without becoming bogged down in the detail of how the process is done now, or should be done in future. Once these issues have been clarified, we can move on to detailed modeling of current processes and engineering of new processes where necessary or appropriate. Once a suitable process is defined, we need to evolve the logical view into a specification and design for computerised support, where appropriate. The following section presents a modeling notation suitable to these tasks. It is followed by examples illustrating their use at different levels.

The Process Notation

The notation is a superset of use cases (borrowing actors, system boundary and activity bubbles) and activity diagrams (activities, dependencies, synchronisation, annotation of outcomes). We introduce it below:

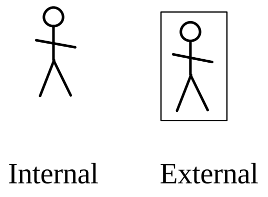
Differentiating Activity Types

Activities are shown in UML with an elipse. We use a rounded corner box, since it allows more text to be captured easily. Recall that our process models will include a complete business process, not just the computerised activities. We make use of the UML stereotype mechanism which allows adding notation to the model. Stereotypes may be expressed as a text within *guillemets*, thus: <<text>> or as an icon. We use these mechanisms to show which activities within a diagram are performed manually, with computerised support (but still requiring human interaction), in a fully automated fashion, or represent embedded processes. In text form these are shown thus: <<manual>>, <<supported>> and <<automated>> and <<embedded>>. As icons, they may be shown thus:



Agents

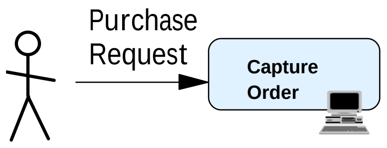
We borrow the use case stick man for human actors interacting with the process. We extend this to two forms:



These can represent individuals, classes of users, or organizational units.

Inputs and Outputs

Borrowing from earlier methods of data flow and event models, we document inputs from and outputs to Agents. On high level models we simply name them on the directional link. This is similar to use case conventions.



On more detailed models, we show them as a large bidirectional arrow, and define their attributes in detail. We optionally define a type, which uses the same symbol, shadowed.

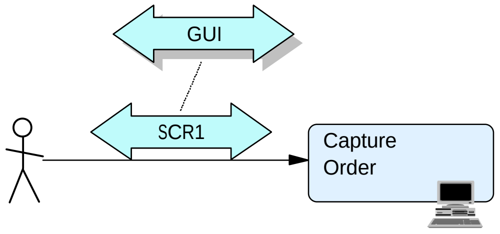


Figure.2 - Referencing IO Specification on the model

At a minimum, we must know the details of the data items which are input or output, their types and the medium through which the input or output will occur. We may document these details in the form of a view, as shown below. A view is not unlike a class, but usually without behaviours. In some tool environments, we have documented these views in UML static model diagrams, using a stereotype of <<View>> to distinguish them from domain classes. The data items mentioned should, of course, be in the repository where they can be typed, described and have legal values and ranges specified.

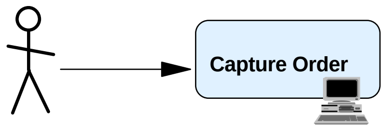
<<View>> **Purchase Request (SCR1-GUI)**
Date: aDate Time: aTime Location: aBranch Customer: aCustomer Products: aCollection of Products RecordedBy: anOperator
Medium:online capture from graphical user interface

Figure.3 - Example of a view definition

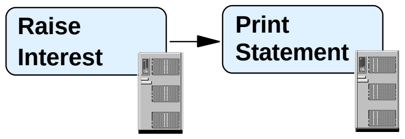
The view should have a unique reference, which we can use to link it to the flow on the graphic model.

Where prototyping is done, the IO ID can reference a prototype artifact, such as a screen, report, or other layout.

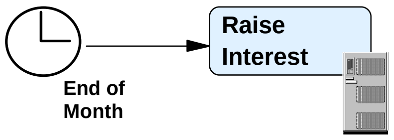
Triggers
Steps or activities in a process can be triggered by various means, including:
- Input from Agent



- Outcome from another activity



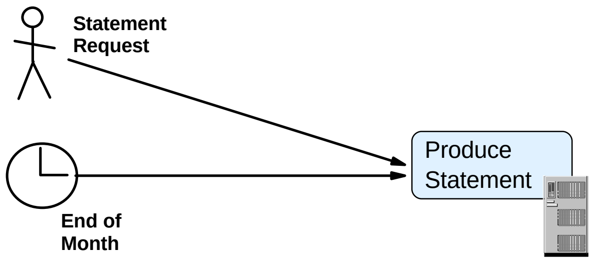
- Time (reached or elapsed), where we show a clock face, and the relevant condition



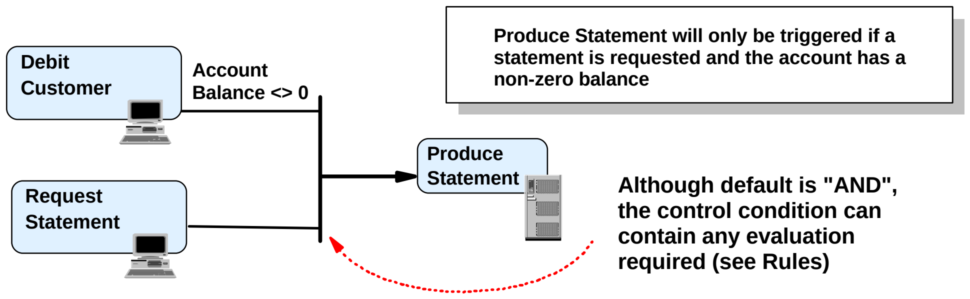
Arrow heads on links are optional if the flow direction is the default left to right and top to bottom.

Selective Invocation
Entry to some activities will be optional, or based upon certain criteria or conditions being met. These can be shown as follows:

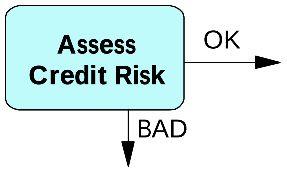
Either Condition Should Trigger the Activity



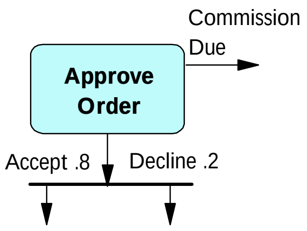
Activity is only triggered when all conditions are met


Debit Customer
Account Balance <> 0
Request Statement
Produce Statement
Produce Statement will only be triggered if a statement is requested and the account has a non-zero balance
Although default is "AND", the control condition can contain any evaluation required (see Rules)

Outcomes
Activities often produce outcomes. On high level models, we express these as an annotation emerging from an activity. If there are multiple possibilities, we can show them and what ensuing activity they trigger. As with Inputs, we simply label them on high level models. They will become more rigorous on design level models.



Where we know the relevant numbers and want to do detailed statistical analysis of the models, we can include probabilities of various outcomes being reached. These are shown as a decimal fraction between 0 and 1, with 1 being certainty. Outcomes can be disjoint, in which case they are shown as emerging from a synchronisation bar. If they are not joined by a sync bar, then they can occur simultaneously. Thus in the following model, we can Accept or Decline (but not both) while Commission may be due in either case.



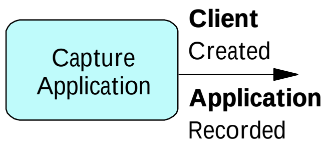
Linking the Activities to the Domain Object States
On high level models, we show outcomes fairly informally. To make our specification more rigorous and to validate the dynamic model against the static model (domain model), we now become more formal in the specification of outcomes during detailed analysis. Specifically, we document the effect of the activity on a domain object type (class) by recording the state that the object will reach. The notation is as follows:

[Object Type] [State]

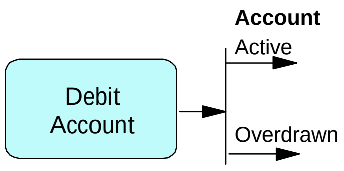
E.g.

CUSTOMER Debited
ORDER Created
STOCK Decreased

Outcomes on the graphical models are updated to reflect this more formal view.



As with the less formal ones on business process models, the outcomes may be independent, as above, or mutually exclusive, as below.



Typical Outcomes (Event Types)
Typical event types that we will be interested in include the following:
- Object creation, deletion or reclassification
- Instances of collections are added or dropped
- The state of an object is changed by updating attributes
- An external event is processed or initiated
  e.g. We get input from the user or change the state of a device

Decomposition and Expansion
Some activities result in multiple events, i.e. they alter the state of several underlying business objects. Where this occurs, we decompose those activities further, until we reach a level where the activities affect only one type of object (class).

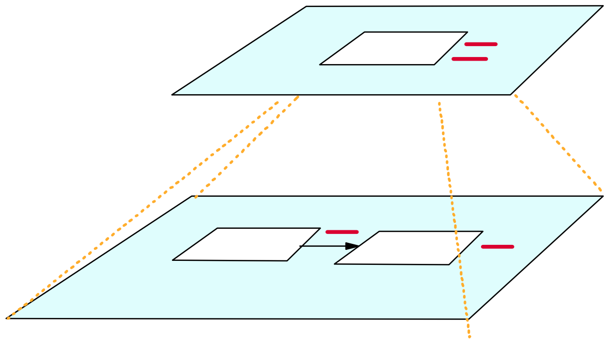


Figure.4 - Activites affecting multiple object types are decomposed

This has several advantages:
- It provides an exact guide to the modeler, as to when to decompose and when to stop, which was often lacking in previous methods (e.g. Functional decomposition or data flow diagramming)
- If an activity affects the state of only one object type (class), then it is a candidate for a method to be mapped to a domain class.

Reuse
To facilitate reuse, we can identify common processing elements which can be common to a variety of subprocesses. In the example shown, there are common issues in processing all kinds of transaction (e.g. Checking authorisation, logging). These common elements can be held at a generic level and "inherited" by sub-processes. The hierarchy in the event diagram should match that of the domain model classes, allowing us to map the common issues into methods on the parent class, while mapping the specifics to methods on the subclasses.

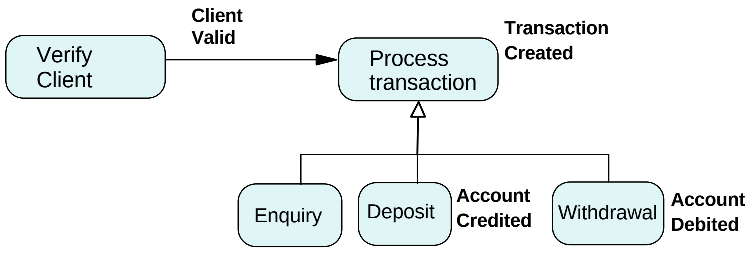


Figure.5 - Common logic in event model

Here, all types of transaction will result in a transaction being recorded, but other activities differ.

Swim Lanes and Bounding Boxes

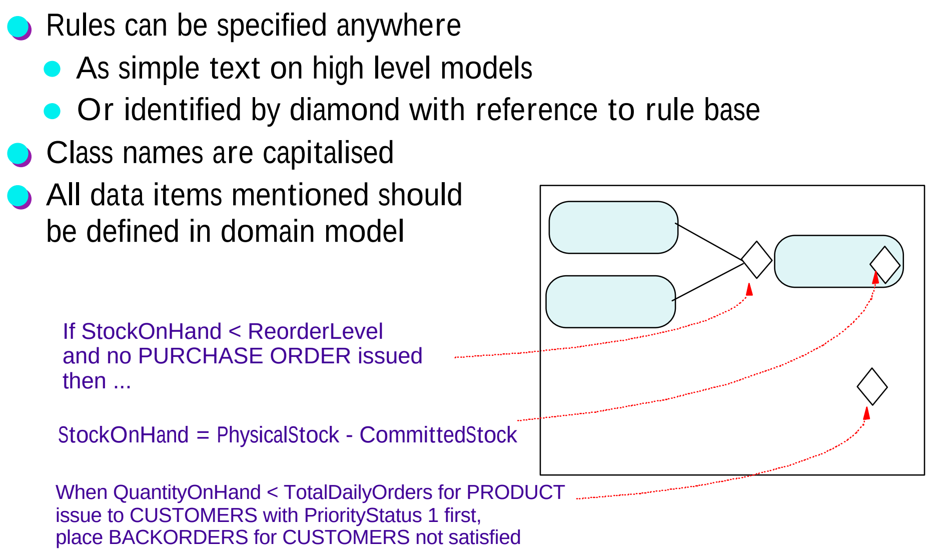
UML offers the concept of swim lanes to group activities which affect the same class of objects. We extend the idea to allow us to group things for a variety of purposes. We also allow the use of bounding boxes for similar purposes. We use them to show:

- Organizational responsibility
- Geographic location
- Business Objects Affected (system level models)
- Logical Transaction Start and Commit (deisgn level models)
- Platform for deployment (design level models)

We can have several layers or overlays per model.

Rules

Rules can be linked to our process models as guard conditions, specifications for how actions are performed or general policies which apply to the process overall. See the examples that follow:



- Rules can be specified anywhere
  - As simple text on high level models
  - Or identified by diamond with reference to rule base
- Class names are capitalised
- All data items mentioned should be defined in domain model

If StockOnHand < ReorderLevel
and no PURCHASE ORDER issued
then ...

StockOnHand = PhysicalStock - CommittedStock

When QuantityOnHand < TotalDailyOrders for PRODUCT
issue to CUSTOMERS with PriorityStatus 1 first,
place BACKORDERS for CUSTOMERS not satisfied

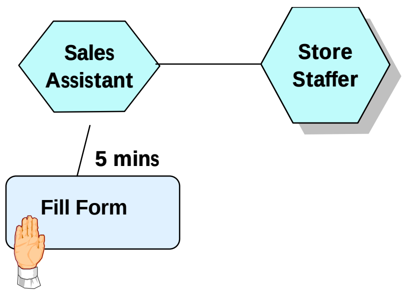
Figure.6 - Rules added to Process Model

We may wish to specify conditions which “guard” access to the activity that must be met before that activity will run. They are shown as a rule, with a diamond symbol, at the entry to the activity. A post-condition would be specified on exit from the activity, and could say under which conditions the activity will complete, or could specify under what conditions the activity will notify the system of an event of interest to other parties. We can also have derivation rules specified within the activity for how results will be arrived at, e.g. A calculation formula or algorithm. Finally, we can have a rule which is attached to the diagram as a whole, to express more complex business policies which may affect the overall process.

If we want rigorous rules, we express them in Object Constraint Language (OCL) [OMG, 1999b] which can appear on any model enclosed in braces viz. {*rule*}.

Hyperlinks and Multimedia
Borrowing from the CREWS (Customer Requirements Elicitation With Scenarios) Project philosophy, as presented at CAiSE'99, we allow hyperlinks to be placed on model elements (usually the name) which will link to further information. This can be in the form of web pages, documents, databases or other forms.

Resources
If we want to model the resources consumed by a process, or see the impact of resources as constraints or on costs, we can add resources to the model. Generally, resources are consumed or produced by activities. We can specify the resource usage in the properties of the activity, or we can show them graphically on the model. Resource *types* use the same symbol, but shadowed. The type allows us to specify constraints on the number of a given resource type available, e.g. We have 10 Clerks. We can also use it to hold a cost rate per unit of consumption.



Example

Using the above devices, we can now look at an example which will illustrate their use. We begin with a process intent model at the architecture level, then show its evolution to a business analysis stage and, finally, to a design level model for the computerised portion of the business process.

Our example deals with the processing of a customer sale, from receiving the original request to purchase, through checking the customer identity and credit, checking stock (potentially ordering from a supplier if insufficient), charging the customer account and fufilling the order by arranging delivery through a partner organization.

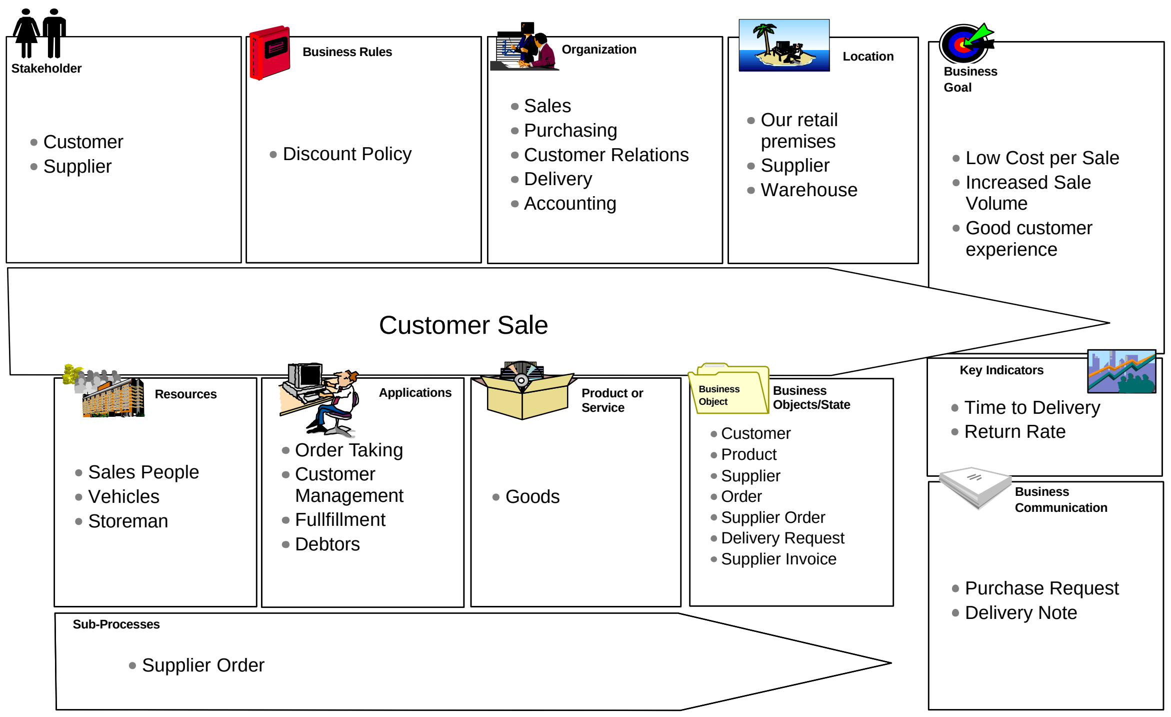


Figure.7 - Process Architecture for Customer Sale

Using the intent model as a starting point, we would develop a business process model. Often we will build an "as is" picture, as well as a "desired future" picture. For our example, the business process model appears as follows:

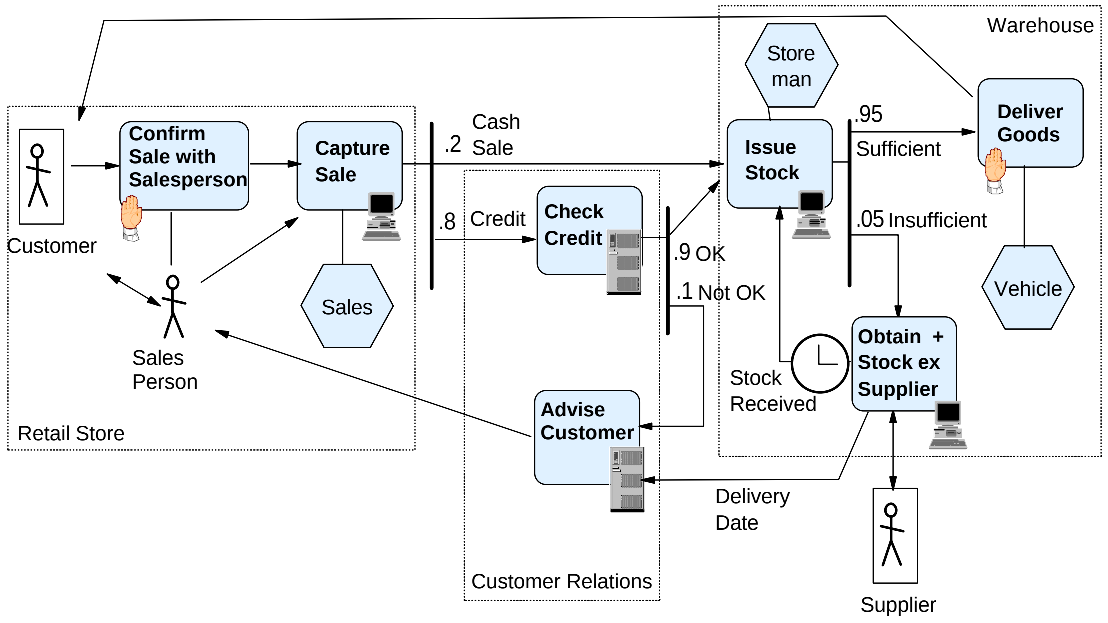


Figure.8 - Sale Business Process

Instrumenting the Process Model

We can add properties to the activities to express the following:

- Minimum, average and maximum **duration** of the activity (current, target)
- **Lead time** before the activity can commence (e.g. waiting for external activity). Minimum, average and maximum can be expressed if desired
- **Organizational responsibility** (which department, section, business unit performs it)
- **Resources consumed** (type of resource, unit of measure and consumption minimum, average and maximum)
- **Number of servers -** the number of resources available to perform the activity. This allows us to gauge the effect of adding or subtracting resources without changing the model structure
- **Geographic location**(s) (where the activity can be performed)
- **Cost** of performing the activity once (current, target)

All of the above extensions are optional. We may use the model to simply express the process and the flow, or add as much detail as desired or available. The more detail we add, the richer the understanding of the business process. A fully attributed model can allow sophisticated analyses of competing business process alternatives including:

- Determining the duration (minimum, average, worst case) of the overall business process using critical path techniques as used in project management.
- Project Evaluation and Review (PERT) techniques can be used to determine most likely times and probabilities of meeting various time benchmarks if required. A full treatment of these is not possible here, please consult [Kerzner, 92; McLeod 98]
- The cost of performing the process can be calculated by summing the costs of all activities traversed. Where activities are traversed more than once (e.g. picking stock for each line on an order), the sum of these occasions would be included. Where activities are optional, the probability of performing the activity times the cost will be included if we calculate an overall average scenario; or we can calculate the cost of various scenarios by computing the cost of specific paths
- The resources consumed can be calculated in a similar manner to costs
- Queuing effects can be brought into play. If the arrival rate of new requests for the process is such that a new request will arrive at an interval shorter than the processing time, then queuing will be experienced. We can use standard queuing network analysis techniques to determine the effect of such queuing on experienced duration by the initiating actor

Once we have a specification for how the process should be accomplished in business terms, we can extract the computerised elements and subject these to further analysis, adding rigour by defining the inputs and outputs in detail, and recording the effects of operations on state of domain objects. The system level model for our example follows.

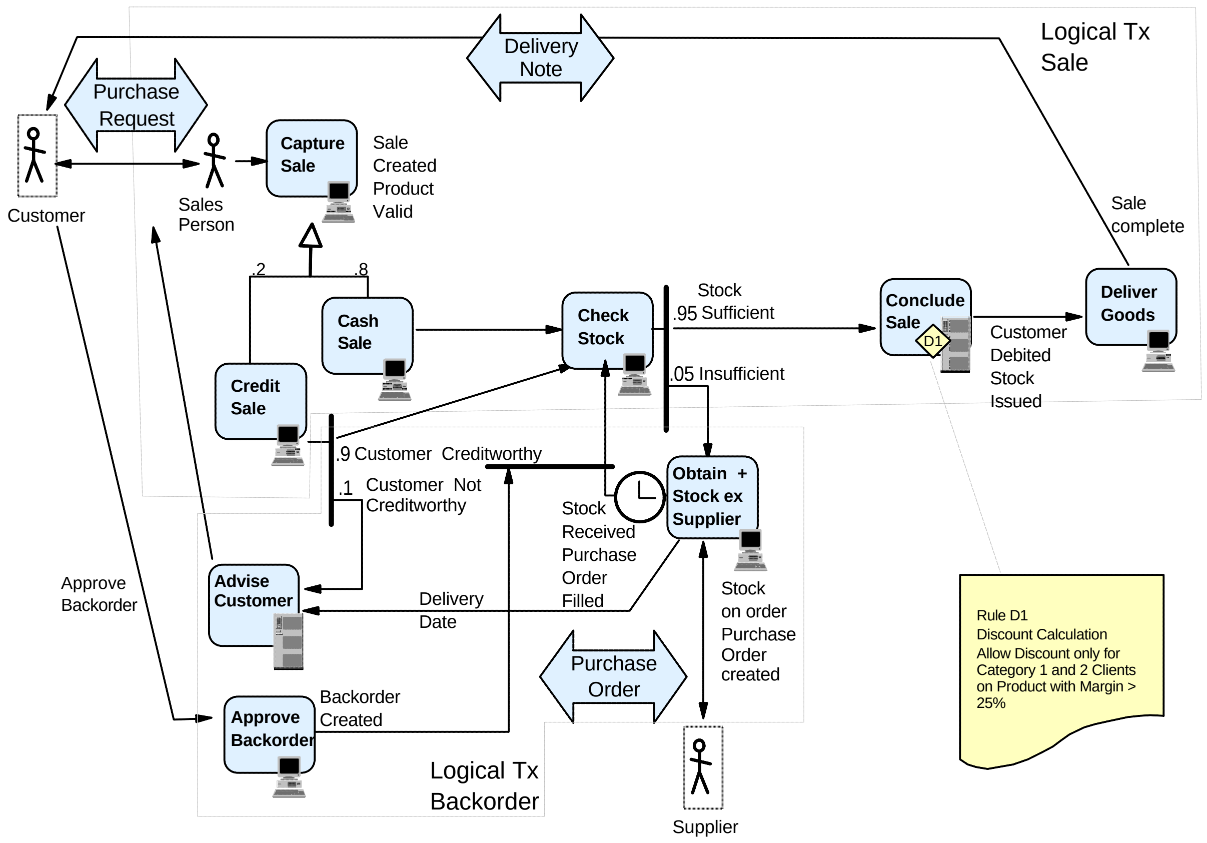


Figure.9 - System Level Model

Note the use of inheritance to factor common processing; bounding boxes to show logical transaction boundaries; a rule to capture the policy with respect to discount; and the more rigorous definition of outcomes in respect of the state achieved for affected business objects.

Design Level Models

As we transition our logical level models to design, we need to consider a number of issues. We discuss these below.

- Technical Objects and Events: We may need to add extra activities and events dealing with issues such as security, integrity and auditability. Examples would include logon procedures, logging updates, writing to an audit trail and so on. In some cases, we will need to add new items to our static model e.g. We may need a class for Authorisations to link users to permitted activities. We would add these to our class diagram. This is now evolving to be more than a domain model, to include system level classes which may not necessarily form part of the business domain. Nonetheless, the classes added will use the same notation and adhere to the same principles as domain classes.
- Geographic or Platform Distribution: Bounding boxes can allow us to indicate platform allocation (e.g. Client, Application Server, Corporate Server) or geographic split (e.g. Branch, Head Office).
- Capacity Planning and Performance Estimating: If we fully specify activities and detail resources and volumes, the models can be used for capacity planning and performance estimating.

Advantages over Standard UML Approach with Use Cases

Our work with the approach in teaching and industry has shown the following advantages:

- The models provide a powerful means of showing business processes, from a high level business view, right down to the detailed operation of the final system. This is important since there are no abrupt transitions of approach between the business, logical and physical dimensions of the system. This promotes user/analyst communication and ultimately results in applications which better match real world requirements
- The lack of translation facilitates an iterative approach to development, where it is easy to cycle back and enrich an existing model. This is much more difficult where several types of models are involved
- The rich attributes of the high level models permit competent business engineering and comparison of scenarios including elements of resources, costs, timing, location and organizational responsibility
- Validation of domain model
- Natural way of factoring functionality to match the corresponding data structures (held in the object model). Decomposition is controlled by the stipulation that each activity should affect the state of just one object type.
- Easy guided mapping of lowest level activities to methods of the appropriate class. In a previous paper [Mcleod, 1998b] we showed the translation of event models into business process logic within a layered design architecture. There is thus a natural and easy transition from high level business models to the architecture of the actual runtime design.

Bibliography

Castellani, Xavier, 1998, **An Overview of the Version 1.1 of the UML Defined with Charts of Concepts**, Proceedings UML'98 Beyond the Notation, International Workshop, June 3-4, 1998, Mulhouse, France

Frank, Ulrich, 1994, in Ege, R; Singh, M; Meyer, B (Hg): **Technology of Object Oriented Languages and Systems,** Prentice Hall pp 367-380

Hammer, 1990, Re-engineering work: Don't Automate, Obliterate, **Harvard Business Review**, July-Aug pp 104-112

Henderson-Sellers, Brian, 1999, **Introduction to the OPEN Method with UML**, Tutorial, TOOLS 29, June 7-10, Nancy, France

Inspired, 1999, **Inspired Architecture Frameworks**, *Inspired* Box 384 Howard Place, 7450, South Africa, www. inspired.org

Jacobson, Ivar; Erissson, Maria & Jacobson, Agneta, 1994, **The Object Advantage: Business Process Reengineering with Object Technology,** Addison Wesley

Kerzner, Harold, 1992, **Project Management: A systems approach to planning, scheduling and controlling**, Van Nostrand Reinhold

Martin, James & Odell, James 1993, **Principles of Object Oriented Analysis and Design**, Prentice Hall, Englewood Cliffs, NJ

McLeod, Graham, 1992-1997, **Advanced Systems Engineering with Objects**, Inspired, Box 384 Howard Place 7450 South Africa

McLeod, Graham, 1998, **Extending UML for Enterprise and Business Process Modeling**, Proceedings UML'98 Beyond the Notation, International Workshop, June 3-4 1998 Mulhouse, France.

McLeod, Graham, 1998b. **Linking Business Object Analysis to a Model View Controller Based Design Architecture**, Proceedings of the Third CAiSE/IFIP 8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design EMMSAD'98, Pisa, Italy, June 8-9, 1998

McLeod, Graham, 1999, **Comprehensive Object-Oriented Business Process Modeling**, Proceedings TOOLS 29, Nancy, France June 7-10, 1999, IEEE Computer Society

OMG, 1999, **OMG Unified Modeling Language Specification**, Vsn 1.3, Object Management Group, June 1999

OMG, 1999b, **OMG Object Constraint Language Specification,** Object Management Group, June 1999

Porter, Michael and Millar, VE; 1985, **How Information gives you Competitive Advantage,** Harvard Business Review, July-Aug pp 149-60

Rational Corporation, 1997, **UML Notation Guide, version 1.0**, Rational Corporation

Seybold, Patricia & Marshak, Ronni, 1998, **Customers.com: How to create a profitable business strategy for the Internet**, Century Business Books

Simons, Anthony, 1999, **Use Cases Considered Harmful,** Tools 29, Proceedings of the TOOLS Conference June 7-10, Nancy, France, 1999, IEEE Computer Society

Taylor, David, 1994, **Business Engineering with Objects,** John Wiley