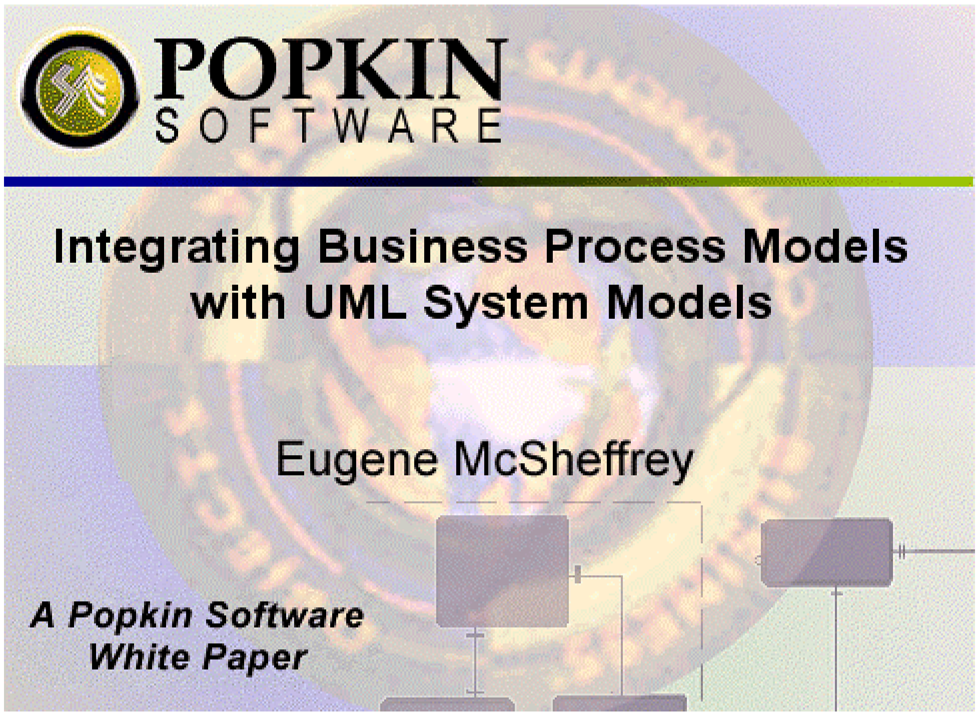

POPKIN SOFTWARE

Integrating Business Process Models with UML System Models

Eugene McSheffrey

A Popkin Software White Paper

Table of Contents

Integrating Business Process Models with UML System Models 1
Example 4
  Scope Model 4
  Enterprise Model 4
  System Model 5
  Technology Model 6
Conclusion 7
References 8

Integrating Business Process Models with UML System Models

The range of notations at the disposal of business modellers has never been greater. Traditional notations based on process flowcharts and hierarchical decomposition now exist alongside system models created in more modern notations such as UML. The popularity of UML has caused it to be promoted as a notation for business modelling, as well as its original purpose of modelling Object-oriented (OO) systems.

UML diagrams, which are often proposed for business process modelling, are the Use Case Diagram and the Activity Diagram. A Use Case Diagram shows processes in a non-sequential representation, while Activity Diagrams show sequence in a manner similar to a flowchart. The UML allows the Activity Diagram to be used to depict procedural flow of control in several contexts, from Classes and Operation implementations (implementation level) to Use Cases (analysis level).

However, the use of UML as a business modelling notation raises several problems. The UML naturally concentrates on representing OO concepts, and needs to be significantly extended to represent business concepts. No single UML business modelling extension is generally accepted at present. The UML 1.3 specification includes a "Standard Profile" for business modelling (UML 1999) but this describes business modeling extensions in terms of object-oriented concepts, with which most business users are unfamiliar. Others, for example Eriksson and Penker (Eriksson 2000) have proposed more elaborate extensions.

Moreover, the use of a single modelling language to span the abstractions required by business users, analysts, system designers and developers requires enormous care to ensure that the model will be correctly interpreted by all parties. The use of similar modelling constructs for several purposes can lead to confusion. Depending on the context, UML allows an "Activity" symbol to represent a step in a Use Case, the invocation of an Operation or a fragment of program specification.

The "Zachman Framework" (Zachman 1987) provides a useful list of "perspectives" or "views" which need to be addressed when designing an information system to meet the needs of an enterprise. In increasing order of complexity, these are:

1. Scope (Context View)
2. Enterprise Model (Conceptual View)
3. System Model (Logical View)
4. Technology Model (Physical View)
5. Detailed Representations (Out-of-context View) i.e. description of a particular instance of an implementation, rather than a model.

Zachman and others contend that as an OO system description notation, the UML is way of describing implementation views [1]. Using UML to describe Scope or Enterprise perspectives takes it out of its original domain and requires us to map the existing symbol set onto different concepts.

An alternative is to use a distinct notation for Figures 1 and 2.

The process model for Figure 1 tends to be text-oriented rather than pictorial, for example high-level business process descriptions. Figure 2 involves a degree of complexity, which encourages the use of diagrams. Two types of diagram are commonly used for this purpose: a process flowchart ("Process Chart") to show process sequence and a "tree" style diagram ("Process Decomposition") to show hierarchical relationships. In the Process Chart style shown, right-pointing arrows represent business events, left-pointing arrows represent business results and rectangles represent processes.

[1] John Zachman states "Clearly, OO was conceived of to address Row 4 or 5 issues … [David Hay's (Hay 1999)] point is not only that UML is a Row 4 language, but also that there are other languages semantically richer, especially for addressing the higher row issues." (Zachman 2001)

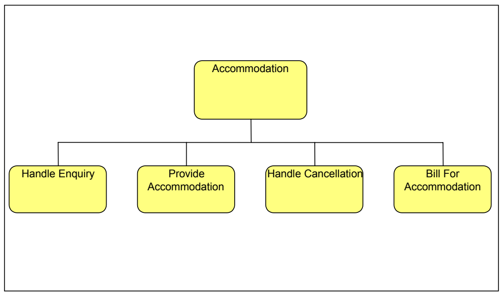

Figure 1: Process Decomposition

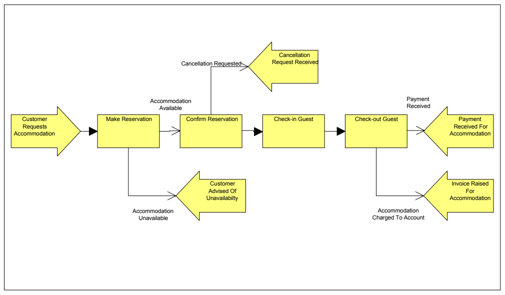
Customer Requests Accommodation
Make Reservation
Accommodation Available
Confirm Reservation
Cancellation Requested
Cancellation Request Received
Check-in Guest
Check-out Guest
Payment Received
Payment Received For Accommodation
Customer Advised Of Unavailabilty
Accommodation Unavailable
Invoice Raised For Accommodation
Accommodation Charged To Account

Figure 2: Process Chart

The concept of process is intrinsically hierarchical. Processes may be defined at any level from enterprise-wide processes to processes performed by a single person. Low-level processes may be grouped together to achieve a common business goal. To avoid modelling to an inappropriate degree of detail we must identify the lowest level of process, which is needed at the Conceptual view. This is usually taken to be the Elementary Business Process (EBP). Definitions of what constitutes an EBP differ, but a common definition is

> *"A process performed by one person in one place at one time, which adds significant value and which leaves data in a consistent state."*

In practice, the person performing the EBP may require to interact with others, so there may be several participants. This is generally the case with customer-facing processes.

Clearly, a business process may contain a mix of automated and manual processes. EBPs, which are to be supported by a system, need to be analysed further to identify system-level operations. The technique used and the nomenclature will depend on the system development method to be used.

UML is the current notation of choice for most enterprise application developers. The UML formally incorporates Use Cases as a modelling technique for defining functional system requirements. Jacobson introduced the technique in his Object-oriented Software Engineering approach (Jacobson et al 1992).

Although Use Cases have been proposed as a technique for business modelling, the definition of a Use Case makes it clear that this is a system-level construct[2]. In order to define system scope, the EBPs that are to be performed using the system need to be related to the system behaviour.
Opinion varies substantially between methodologists as to the scope of a single Use Case. However Jacobson allows Use Cases which seem to correspond to EBPs, for example “Generate Daily Report”, “Acknowledge Flight”, “Confirm Booking”. More recently, Cheesman and Daniels (Cheesman 2001) suggest that a Use Case is, if anything, smaller than a business process[3].

The simplest first-cut mapping is to create one Use Case for each EBP to be automated. More detailed Use Cases may be created if desired, so long as the relationship to the supported EBP(s) is clear.

UML provides a rich set of model artefacts for describing system behaviour to support Use Cases. Interaction diagrams can be used to define Use Case instances (scenarios) performed by objects. Activity diagrams can describe the procedural sequence within a Use Case or operation implementation (UML 1.3 para.3.84.2).

The artefacts, which appear to best support the “Process” aspect for each view, are therefore:

---

[2] Two definitions of Use Case are:

(1) “... a behaviorally related sequence of transactions [a user performs] in a dialog with the system.” (Jacobson et al 1992 p.127)

(2) “The specification of a sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system.” (UML 1999)

[3] “There seems to be no consensus on this issue, but here’s our view: To a first approximation we can say that a use case is smaller than a business process but larger than a single operation on a single component.” Cheesman (2001), p45.

View	**Artefact**
Scope	Text; Informal Business Concept Diagrams. High-Level Process Decompositions
Enterprise Model	Process Charts; Process Decompositions
System Model	Use Case Diagrams; problem-domain Sequence Diagrams; Activity Diagrams for Use Cases
Technology Model	Implementation-level Sequence and Collaboration Diagrams, Statechart Diagrams, Implementation Diagrams, Activity Diagrams for Classes and Operation Implementations

In this article we are concerned with describing a model which relates the Enterprise Model (Business view) to the System Model. The following example shows how this might be done.

Example

Let us imagine a hotel that needs a system to support its accommodation processes, and examine how the "Process" perspective might be modelled.

Scope Model

The Scope model will list the major Business Functions and the Process Groups. This is likely to be essentially textual, with some informal "Business Concept" diagrams and high-level decomposition diagrams showing the relationship between Business Functions and Process Groups. The sequence of processes is not likely to be explored in detail at this level.

Enterprise Model

In this view we are interested both in process sequence and the hierarchy of processes. The diagrams which will be used are "Process Decomposition", process-oriented "Relationship Map", "Process Chart" and "Process Map".

The Process Decomposition shows Process Threads and the Process Groups they belong to. Lower level processes may also be shown if required. A Process Thread indicates a process unit that is initiated by one or more events and ends in one or more results. A Process Thread usually corresponds to the process shown by single Process Chart.

A Process-oriented "Relationship Map" shows the sequential relationships between Process Threads.

The Process Chart shows sequential relationships between Elementary Business Processes within a Process Thread.

Since this is an Enterprise-level view, there is no indication as to which processes, if any, are to be implemented by a system.

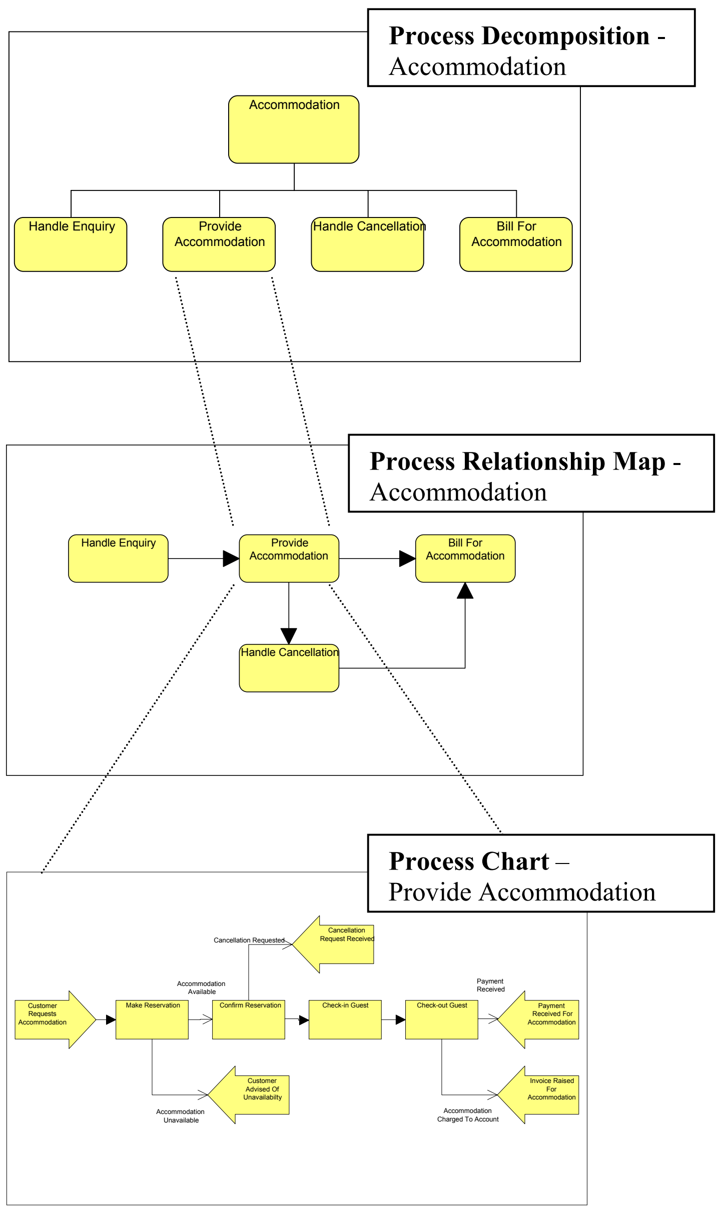
Process Decomposition - Accommodation
Accommodation
Handle Enquiry
Provide Accommodation
Handle Cancellation
Bill For Accommodation
Process Relationship Map - Accommodation
Handle Enquiry
Provide Accommodation
Bill For Accommodation
Handle Cancellation
Process Chart – Provide Accommodation
Cancellation Requested
Cancellation Request Received
Accommodation Available
Payment Received
Customer Requests Accommodation
Make Reservation
Confirm Reservation
Check-in Guest
Check-out Guest
Payment Received For Accommodation
Customer Advised Of Unavailabilty
Accommodation Unavailable
Invoice Raised For Accommodation
Accommodation Charged To Account

Figure 3: Enterprise Model

System Model

The Enterprise Model is a starting point for describing the functionality of any system within the enterprise. The system's capabilities will be defined in terms of the processes it supports.

In UML, the value a system provides to its users can be described in a Use Case Diagram and its supporting descriptions.

As a starting point we create one Use Case for each EBP performed with the support of the system.

The EBP "Make Reservation " is therefore shown as a Use Case performed by the Hotel Reservations System. An Actor can be added to show the staff Role that can perform the EBP.

The System is shown shaded on the diagram.

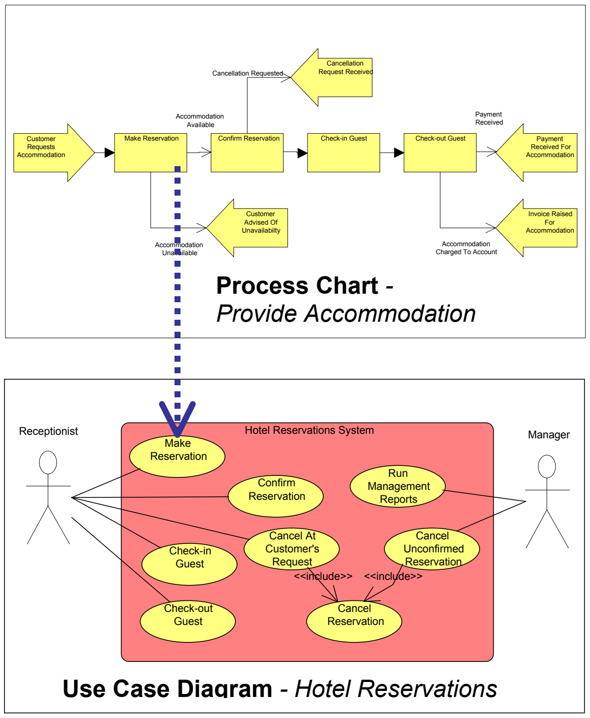
Cancellation Requested
Cancellation Request Received
Accommodation Available
Payment Received
Customer Requests Accommodation
Make Reservation
Confirm Reservation
Check-in Guest
Check-out Guest
Payment Received For Accommodation
Customer Advised Of Unavailability
Invoice Raised For Accommodation
Accommodation Unavailable
Accommodation Charged To Account
Process Chart - Provide Accommodation
Receptionist
Hotel Reservations System
Manager
Make Reservation
Confirm Reservation
Run Management Reports
Check-in Guest
Cancel At Customer's Request
Cancel Unconfirmed Reservation
<<include>>
<<include>>
Check-out Guest
Cancel Reservation
Use Case Diagram - Hotel Reservations

Figure 4: **System Model**

While performing an EBP, the user will carry out a number of sub-activities or "steps". Some of these will be carried out using the system but some will be performed manually. For example, while performing the EBP/Use Case "Make Reservation", many steps will involve obtaining information from the customer by telephone.

UML allows an Activity Diagram to be used to specify sequential behaviour in a Use Case. Used in this way, the Activity Diagram represents a decomposition of a single Use Case.

Both Process Charts and Activity Diagrams represent similar "flowchart" type relationships, but by using Process Charts to show business processes and Activity Diagrams only for system-level behaviour, the distinction between the two model views is made clear.

We can use the Activity Diagram to make clear which steps are carried out by the system itself. The variant of Activity Diagram with swimlanes is particularly useful for this purpose if we assume that the swimlanes map to Roles (treating the system as a Role for this purpose) rather than Organisational Units.

The Activity Diagram can show the behaviour associated with a single Use Case in either of two ways, depending on the user's preference.

In the left-hand diagram the shaded Activities are those performed by the system. In the right-hand diagram the entire system swimlane is shaded.

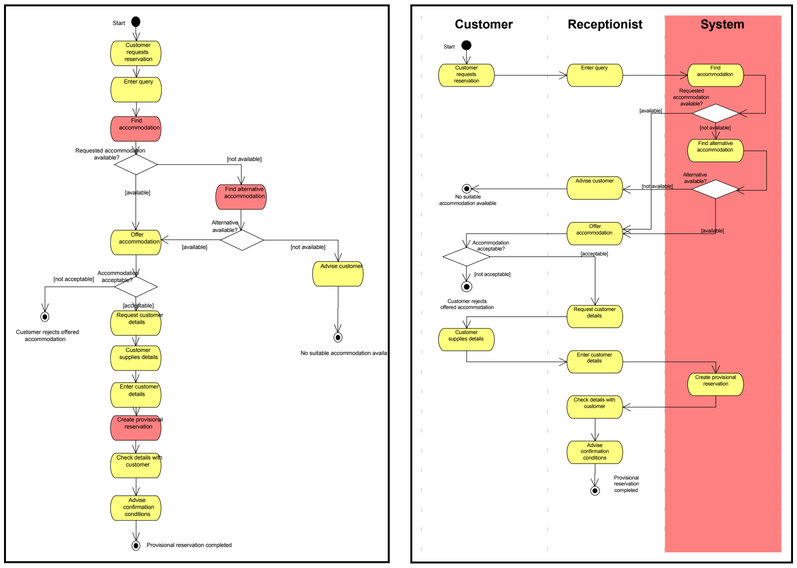
Start
Customer requests reservation
Enter query
Find accommodation
Requested accommodation available?
[not available]
[available]
Find alternative accommodation
Alternative available?
Offer accommodation
[available]
[not available]
Advise customer
Accommodation acceptable?
[not acceptable]
[acceptable]
Customer rejects offered accommodation
Request customer details
No suitable accommodation availa
Customer supplies details
Enter customer details
Create provisional reservation
Check details with customer
Advise confirmation conditions
Provisional reservation completed
Customer
Receptionist
System
Start
Customer requests reservation
Enter query
Find accommodation
Requested accommodation available?
[available]
[not available]
Find alternative accommodation
Advise customer
[not available]
Alternative available?
No suitable accommodation available
Offer accommodation
[available]
Accommodation acceptable?
[acceptable]
[not acceptable]
Customer rejects offered accommodation
Request customer details
Customer supplies details
Enter customer details
Create provisional reservation
Check details with customer
Advise confirmation conditions
Provisional reservation completed

Figure 5: **Alternative Activity Diagrams for a Use Case -** ***Make Reservation***

The Activity Diagram summarises several scenarios of a single Use Case. To make each scenario distinct we can use the UML Sequence Diagram. Initially this will be used to describe interactions between an Actor and the System. This interaction can be derived from the Activity Diagram and the sequential relationships refined to specify messages sent between the Actor(s) and the System. In the UML Sequence Diagram shown, the area enclosing the system is shaded.

System behaviour can then be described in terms of problem-domain objects added to the Sequence Diagrams. A Class Diagram may also be created but this is primarily a structure-oriented rather than a process-oriented artefact.

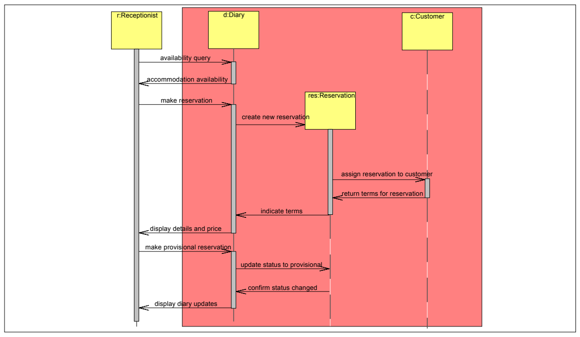
r:Receptionist
d:Diary
c:Customer
availability query
accommodation availability
make reservation
res:Reservation
create new reservation
assign reservation to customer
return terms for reservation
indicate terms
display details and price
make provisional reservation
update status to provisional
confirm status changed
display diary updates

Figure 6: Sequence Diagram for one scenario of the Use Case - *Make Reservation*

Technology Model

The Technology Model can be defined using standard UML artefacts such as detailed Sequence and Collaboration Diagrams (Interaction Diagrams), Statechart Diagrams to show internal object behaviour, Activity Diagrams at the Class and Operation implementation level, and (from the structural perspective) Implementation Diagrams to show software and hardware partitioning.

Conclusion

The technique allows a model expressed in business terms to be mapped to a UML system model. The approach avoids forcing business users to adopt the UML as a way of describing business concepts, and requiring a set of UML extensions to be adopted and explained to business users prior to embarking on This example assumes a simple mapping between EBPs and Use Cases, and shows how a model may be created so as to establish traceability between these “Business-Level” and “System-Level” artefacts. If more than one Use Case needs to be created for a given EBP, the techniques described can still be applied, so long as traceability between EBPs and Use Cases is maintained.

References

Cheesman 2001	Cheesman, J. and Daniels, J., “UML Components – A Simple Process for Specifying Component-Based Software”, Addison-Wesley, 2001
Eriksson 2000	Eriksson, Hans-Erik and Penker, Magnus, “Business Modelling with UML”, John Wiley & Sons, 2000
Hay 1999	Hay, David C. “UML Misses the Boat”, www.essentialstrategies.com
Jacobson et al 1992	Jacobson, I., Christerson, M., Jonsson, P., Övergaard, G.,“Object-Oriented Software Engineering – A Use Case Driven Approach”, Addison-Wesley, 1992
UML 1999	Language Specification” Version 1.3, June 1999, www.omg.org“OMG Unified Modeling
Zachman 1987	Zachman, John A., "A Framework for Information Systems Architecture." IBM Systems Journal, vol. 26, no. 3, 1987. IBM Publication G321-5298
Zachman 2001	Zachman, John A. “Implementing and Managing Enterprise Architecture – Seminar Notes”, ZIFA, Pinckney, MI, USA.