Exploring Services and Requirements – Part two

Author: Charles Edwards
Version: 0.09
Date: 20th June 2008

Abstract

As the architectural concept of Service Oriented Architecture (SOA) is implemented in many organisations, it is starting to introduce somewhat of a dilemma between the Business Analysts and Architects. People in these roles are wondering just how the Business Services, Business Processes and Use Cases all integrate and fit together.

Most systems are implemented as projects, with the scope limited to the associated requirements of the project. However these days, in order to maximise Service re-use, the business community in the enterprise have to think about and define their Business Capabilities and Requirements at more of an Enterprise Level rather than just at a particular System or Project level.

The aim of this series of papers is to explore the different semantics and terminology to align the areas of overlap between the various concepts, in order to suggest an overall simple integrated approach that works satisfying both the Business Analysis viewpoints and the Enterprise Architecture viewpoints.

Last time in part one

In part one, we looked at the scope of each of three roles and defined the problems and challenges the Business Analysts and Enterprise Architects face with the changing landscape of introducing SOA into the enterprise. We discussed how the roles overlap and where some of the common ground is to be found.

This time

As ever semantics is the issue in many of these methodology discussions. What do we really mean by terms like Service, Business Service, Business Process, Business and System Use Case, User Requirements, etc?

So before discussing the possible overlaps and issues in more detail, let's define what is meant by the semantics. We are going to set the scene with some base semantic definitions of three main groups of concepts:

1. **Services** – with related concepts such as Service layers, Interfaces, Operations and Implementation
2. **Business Processes,** Business Use Cases and Business Object Models.
3. **Requirements Specifications** and System Use Cases.

This is so that we can show the meaning of how they are used and then bring them all together in the next paper.

1. Semantics (meaning) of 'Services'

Just to show how broad term 'Service' is the OASIS (organization) defines service as:

> **Service:** "a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description." [reference 3]

ITIL (organization) defines an IT service very specifically within the context of the business of IT as:

> **IT Service (ITILv3):** A Service provided to one or more Customers, by an IT Service Provider. An IT Service is based on the use of Information Technology and supports the Customer's Business Process. An IT Service is made up from a combination of people, Processes and technology and should be defined in a Service Level Agreement.

The trouble with these definitions is that they are in the context of **Systems Architecture** not **Enterprise Architecture**, so my proposed definition of Service which lifts it a little more into the abstract and does not differ that much is:

> **Service:** "a definition of capability for work to be done or processes provided by a service provider to performs action(s) and optionally offers results back, all through a prescribed interface, as requested by a service consumer, based upon a service contract and service interactions."

This service could be conceptualized in human or technology terms, it does not matter. If a human provides or does some service (work) for a consumer, or if a computer automatically does or provides it, a Service still works for both options.

A service provider provides some capability of work for a service consumer. This is where some of the confusion arises between what is put into business use cases and system use cases, because there has been the concept of people do business processes and computers do system processes. More on this topic will be discussed in the next paper.

1.1 A Simple Service

So what does a simple service look like? Using class models we have modeled some of the concepts. This figure below is the simplest view of a service, which we build upon in subsequent sections below:

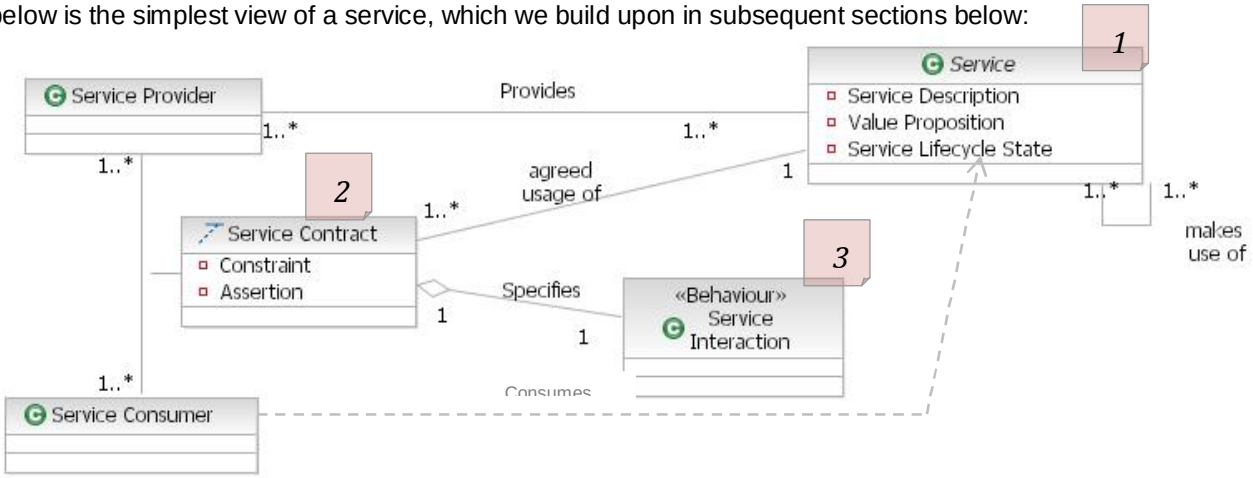


Figure 1 - Most Basic Service definition

A **Provider** provides a **Service** (capability to do some work or activities), which a **Consumer** consumes:

1. Services can **make use of** other services for work to be done. The notion of atomic, complex and orchestrated services is based upon this concept of Services calling other (lower level) sub-services.
2. A **Service Contract** is specified between the Provider and the Consumer to ensure a mutual understanding of the successful usage of the service. This sets out assertions and constraints on how the service will be used, consumed and provided. This Service Contract offers a "black box" view of requirements, because at this point the Consumer does not particularly care **how** the Provider is going to achieve this promise to deliver. It only specifies **what** will be delivered. Thinking about it, does this not sound like a formal set of "Requirements" on the Service to you?
3. The **Service Interaction** is a definition or specification of how the Interaction between the Consumer and the Service should behave. Still a "black box" view of the service, because provided the Consumer and Service work to this "protocol" the Service can implement the interaction any way it likes behind the scenes. Thinking about it, does this not sound like a formal set of "Use Case Interaction Steps" on the Service to you?

1.2 The Layers of Services

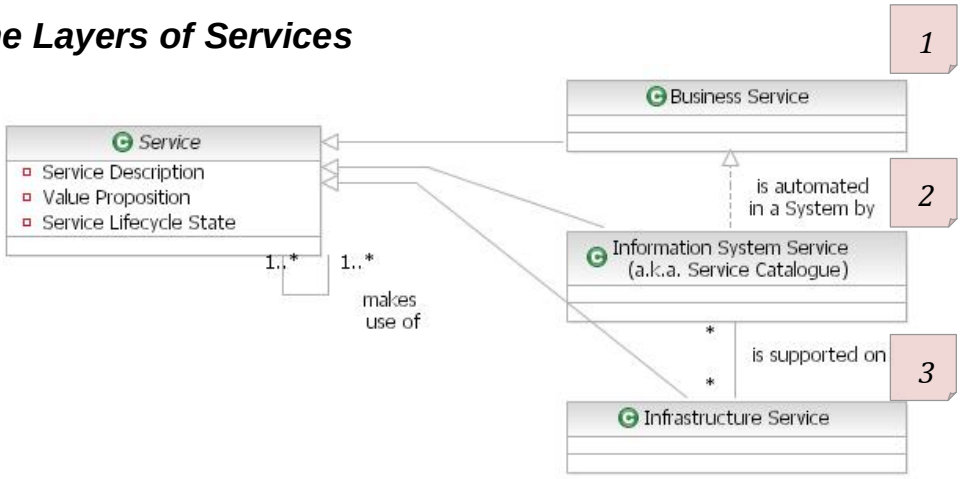


Figure 2 - Breakdown of Service Types

The figure above shows three types of layers of Service being described, they are:

1. **Business Service** – This describes the Business services in Business terms. E.g. '*Place Order Service'*, is a service the business offers to its customers in order to do business.
2. **Information Systems Service** – This describes the information systems that support the Business Services, such as *Place_order()*, which as we mentioned before could call on other services one of which could be *Authorize_Payment() or*
3. **Infrastructure Service** – These Infrastructure Services specify the Infrastructure that supports all the Information System Services. E.g. *A software Web Server service, a software Application server service, a software Database service, a hardware Web Server service, a hardware Application server service, a hardware Load-balancer service, a hardware Storage Service, a Wide Area Network connectivity service*, etc.

Sometimes through loose use of terms, and also depending upon the context, confusion can arise. It seems as though the term 'Business Service' has the potential to confuse because of the SOA focus being mainly in the Information Systems layer at the moment.

There are two possible uses of the term 'Business Service' and the difference is quite subtle but confusing:

1. In the above definition a '**Business Service**' means a specific business Service describing how the Business provides actions for consumers (as opposed to IS Services that implement an actual service)
2. In other loose forms it could simply mean a **'Service' that supports the business**. (the implication is an 'IS service' that supports a business implementation.)

In the article "Building SOA composite business services, Part 1: Develop SOA composite applications to enable business services[reference 4] it talks about a 'Business Service' in the latter sense when actually they mean an **Information System Service**. This lack of clarity does not help semantics in general.

1.3 Service Interface, Operation and Implementation

Taking a Service to the next level, and slightly more complex, we introduce the concept of *a Service Interface, a Service Operation and Service Implementation.*

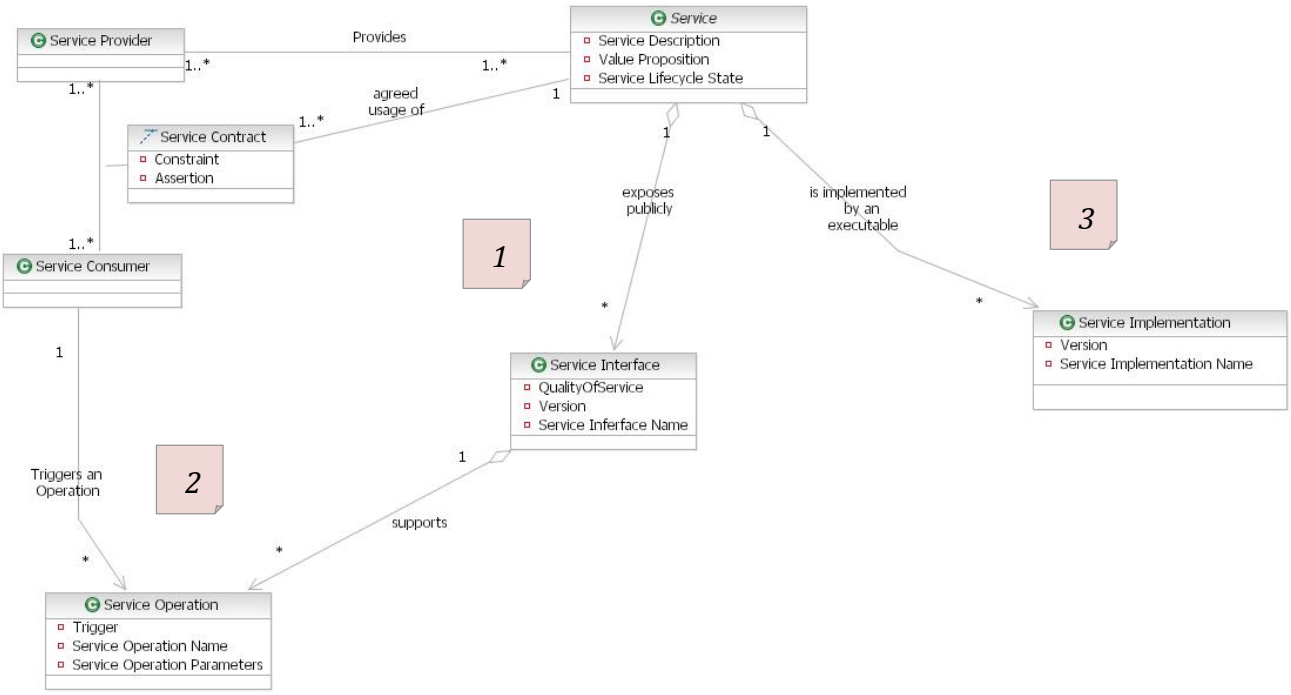


Figure 3 - More Refined Service Definition

. These are defined in the numbered figures above as:

1. The **Service Interface** is the public facing side or Black Box View of a Service which Service Consumers call, hiding 'how' it does this work. This is a deliberate architectural mechanism for separating out complexity. All that remains constant is how a consumer calls the service, meaning that the service can change 'behind the scenes'. Notice, the Consumer can only call on the Service via the 'Service Interface'. So change can happen without breaking other services that rely on it, provided the 'Service Interface' remains true to the 'service contract' then Service Consumer is none the wiser.
2. The **Service Operation** – This is simply the set of operational calls that can be made on the Service as part of the Service Interface, which will either do something and supply an answer or just do something. Typically there could be one or many of these per Service Interface. Usually this is the most detail you would see or need to know about a service, the rest is usually hidden behind the scenes in the "black box". Notice too that the Service Consumer must call calls on the detailed Service Operation as part of the Service interface to invoke the service.
3. The **Service Implementation** is the private internal part or White Box View of a Service. This is the detail of 'how' the service works. This can change, in two ways, by changing the detail it relies described in the 'Service Interface' or by keeping it the same. The service implementation is the part of the Service that needs to know the Process it should follow.

1.4 Service Operation and Service Implementation Detail

Taking the Service definition down to the next level of detail, we introduce the concept of the 'Service Operation' and 'Implementation Detail'.

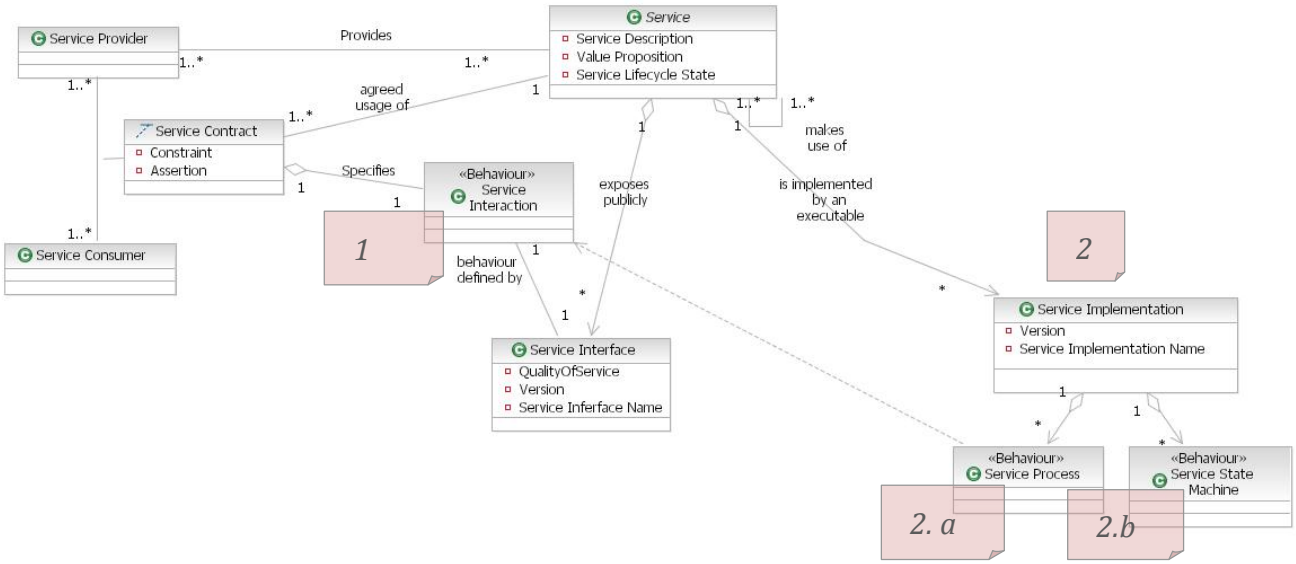


Figure 4 - Details about Service Implementation

These are defined in the numbered figures above as:

1. **Service Interaction** – This specifies in detail the 'protocol' and 'ordering' of the interaction between the Service Consumer and Service Provider.
2. The **Service Implementation** detail – The implementation, or 'how' the service actually does the work behind the scenes "white box", is further broken down into various parts:
    a. **Service Process** – This defines the 'process' that the service will follow to do the work, be that at a business level or a system level. If it is going to be automated this is the bit that will be used to define what happens and in what order. This is the behaviour of the Service (as opposed to the structure of the Service)
    b. **Service State Machine** – This defines the states the Service could be in at any point during the Process of doing the work for the service. This state is not shared out generally, retained or persisted, but is private to the service and is only used for internal control. E.g. If describing a grocery place order service, might be something like "Basket Empty", "Basket contains items", "Basket checked out".

2. Semantics (meaning) of Business Modelling

2.1 Business Process

A **business process** [reference 8] is a collection of interrelated tasks, which accomplish a particular goal.

There are three types of business processes:

- **Management processes**, the processes that govern the operation of a system. Typical management processes include "Corporate Governance" and "Strategic Management".
- **Operational processes**, processes that constitute the core business and create the primary value stream. Typical operational processes are Purchasing, Manufacturing, Marketing, and Sales.
- **Supporting processes**, which support the core processes. Examples include Accounting, Recruitment, IT-support.

A business process begins with a customer's need and ends with a customer's need fulfillment. Process oriented organizations break down the barriers of structural departments and try to avoid functional silos.

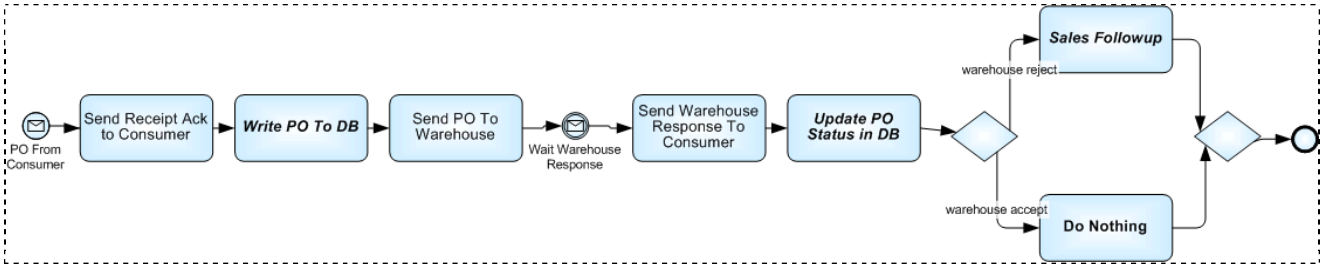


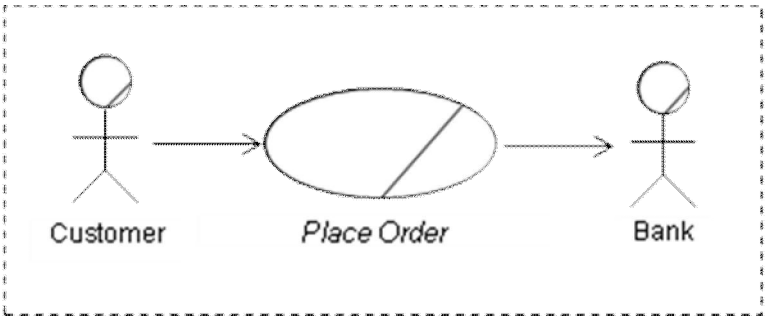
Figure 5 – An example BPMN Business Process for purchasing

A business process can be decomposed into several sub-processes, which have their own attributes, but also contribute to achieving the goal of the super-process. The analysis of business processes typically includes the mapping of processes and sub-processes down to activity level.

Business Processes are designed to add value for the customer and should not include unnecessary activities. The outcome of a well designed business process is increased effectiveness (value for the customer) and increased efficiency (less costs for the company).

Business Processes can be modeled through a large number of methods and techniques. For instance, the Business Process Modeling Notation is a Business Process Modeling technique that can be used for drawing business processes in a workflow.

2.2 Business Use Cases



A **Business Use-Case** (BUC) is "using a business": this recognises that businesses[footnote1] are created and organised in order to *do things for people* – mainly customers, but also other "actors" including actors inside the business. So a Business Use-Case is a way in which a customer or some other interested party can *make use of the business* to get the result they want – whether it's to buy an item, to get a new driving licence, to pay an invoice, to procure software, to deliver software, to define a business policy; whatever. An important point is that a single

[footnote 1] Calling these "businesses" can itself be (unintentionally) misleading. A large part of the economies of all countries is conducted by organisations other than commercial businesses – local and central government agencies, charities and other non-profit organisations, etc. Their activities equally fall into this area. Strictly, we should be saying something like "Businesses and other organisations", or maybe just "Organisations"; but we'll stick with convention.

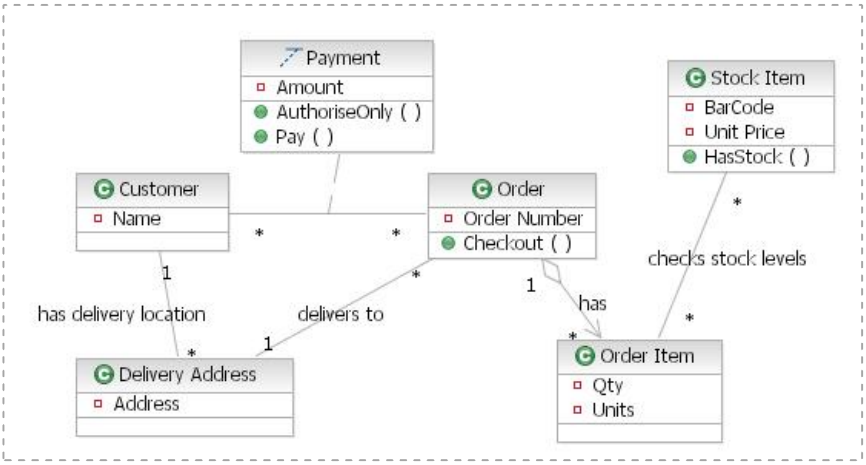
execution of a Business Use-Case should encompass *all* the activities necessary to do what the customer (or other actor) wants, and also to do any necessary internal tidying-up. So the duration of a BUC execution can vary greatly, depending on its nature. Some BUCs, like withdrawing cash from an ATM, can be done in less than a minute; others, like ordering goods for delivery, or getting a new phone line installed, can take days, weeks or even longer. The business use case also embraces any manual or automated issue such as “phone the customer once the order is confirmed”, “send the email“, etc. [Reference 6]

2.3 Business Object Model

Within Business Process Management all business objects, their attributes and their relations are defined in the **business-object model** (*BOM*). To describe the information processed by software modules, the semantics must be unified. That is called "semantic integration". The *business-object model* is a result of *semantic integration* and is a representation of enterprise semantics. [reference 9]

This is described using a UML Class modelling notation to define any Business Concept.

In this case the example in the figure to the right describes the concept of Customer, Delivery Address, Order, Payment, Stock items and Order items.



3. Semantics (meaning) of System Requirements

3.1. Requirement Specifications

Requirement – A requirement specifies a capability or condition that must (or should) be satisfied. A requirement may specify a function that a system must perform or a performance condition a system must achieve.[reference 5]

For much of the time requirements have been built in spreadsheets or requirements management databases and looked like the figure to the right. SysML has done some great work in the Requirements space recently in taking the traditional concept of requirements in its textual form and begun modelling it in a more visual fashion, but not losing the essence. The diagram which follows is the SysML textual representation of requirements (which as you can see is still possible using this new SysML)

table [requirement] Performance [Decomposition of Performance Requirement]

id	name	text
2	Performance	The Hybrid SUV shall have the braking, acceleration, and off-road capability of a typical SUV, but have dramatically better fuel economy.
2.1	Braking	The Hybrid SUV shall have the braking capability of a typical SUV.
2.2	FuelEconomy	The Hybrid SUV shall have dramatically better fuel economy than a typical SUV.
2.3	OffRoadCapability	The Hybrid SUV shall have the off-road capability of a typical SUV.
2.4	Acceleration	The Hybrid SUV shall have the acceleration of a typical SUV.

The figure below shows a modelled view of performance requirements. These diagrams were taken from the

SysML specification paper examples.

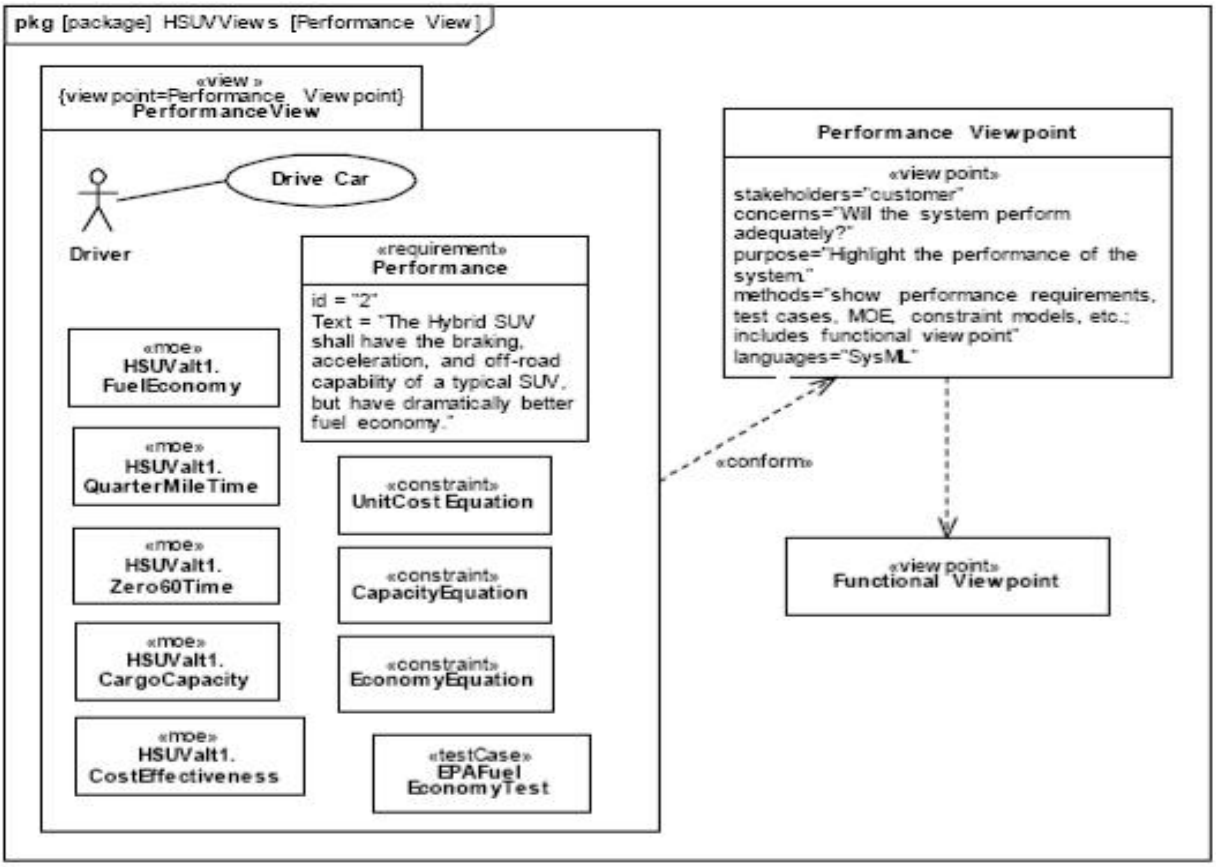
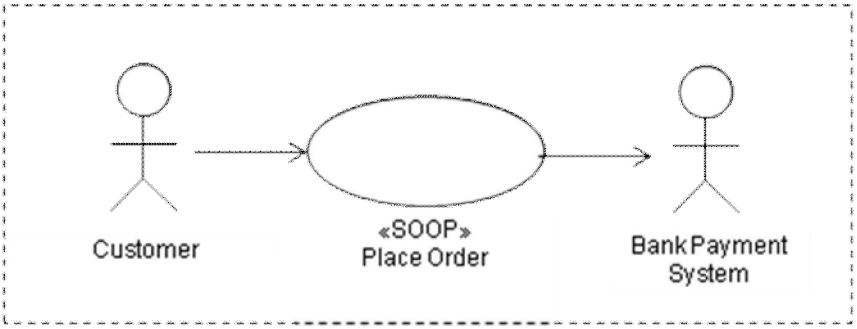


Figure 6 - SysML requirements diagram model

3. 1 System Use Case

In contrast, a **System Use-Case** is a way in which a user of a *computer system* can *make use of the system* to get the result they want. This will typically be something we can readily imagine as being done in a single sitting on a single computer, usually with a single UI, or a small number of closely-related screens such as a wizard, and taking maybe between a couple of minutes and a half-hour at most.



Alistair Cockburn suggests that a useful guideline is the “coffee-break test”: once the user has completed (a single execution of) the System Use-Case, s/he can take a coffee-break with a clear conscience. The system use case avoids all manual issues such as “file the printout” or “phone the customer once the order is confirmed”, etc. [Reference 6]

Next time

Next time we will use all the above definitions to show a worked example of a Business Use Case called ‘Buy Groceries’ which will have a Business Process and related Business Services, to see how they can map together and wherever possible identify re-use of existing services.

References

[3] OASIS website: http://www.oasis-open.org (Oasis Reference Model for a Service Oriented Architecture: http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf)

[4] 2006, Oct 3: Building SOA composite business services, Part 1: Develop SOA composite applications to enable business services http://www.ibm.com/developerworks/webservices/library/ws-soa-composite/index.html?S_TACT=105AGX04&S_CMP=ART

[5] 2007-09-01: OMG SysML Specification: http://www.sysml.org/docs/specs/OMGSysML-v1.0-07-09-01.pdf

[6] 2008 May: Langlands, Martin and Edwards, Charles: Business versus System Use Cases, http://www.agileea.com/portal/index.php/whitepapers

[7] 2008 June: Wikipedia defines of Business Process http://en.wikipedia.org/wiki/Business_process

[8] 2005 July : Harvey, Michael: What Is Business Process Modeling - http://www.onjava.com/pub/a/onjava/2005/07/20/businessprocessmodeling.html?page=3

[9] 2008 June: Wikipedia defines Business Object Model http://en.wikipedia.org/wiki/Business_Object_Model

Contact details

Charles.Edwards@processwave.com

www.processwave.com and www.AgileEA.com