Business vs System Use Cases – Part two

Author: Martin Langlands and Charles Edwards
Version: 1.0 Date: 06 May 2008

Abstract

Use-cases are now all but universal as the basic concept for specifying requirements of commercial information systems. However, one area that causes problems is distinguishing between "Business" and "System" Use Cases. The aim of this three part article series is to shed some light on this issue by highlighting the *differences* between the two, and proposing a diagrammatic way of showing how they are *related*. This is illustrated using a more detailed and meaningful example than is used many introductory texts.

Previously in part one

In Part One we started to discuss the difference between the terms Business and System Use Case, and showed our worked Business Use Case example.

The System Use-Cases

OK, now we've seen what a Business Use-Case looks like, let's take a look at the relevant System Use-Cases. This will allow us to see clearly the differences between the concepts, and how they are related.

First, let's make the (perhaps obvious) observation that we are certainly *not* going to have one SUC corresponding to this BUC. The scope and nature of one execution of the BUC is not something that we can conceive of being done in a single sitting in front of a computer. The customer may perceive it as being so, but the groceries won't arrive the moment they finish placing the order!

Secondly, in order to illustrate the SUCs here, we're going to have to be a bit more imaginative than with the BUC above. At a guess, Figures 2 and (in part one of this series) would just about work for any home-delivery grocery anywhere in the world. However, the SUCs will depend on what systems SupaStores uses, and their functionality. So let's assume that they have a not-untypical mix of reasonably up-to-date systems, combined with some older legacy applications, and with some level of integration between them. These will be discussed as we go.

Finally before looking at the SUCs themselves, note that in doing this we are now moving from the *what* gets done, to *how* it's done, in terms of the specific system support for each step. From our (invented) knowledge of Supa-Stores' systems, we can go through the business use case and identify the various system use cases that deliver this support.

The first step – "Place Order" – is supported by one system use case, which has also been called *Place Order.* Figure 4, shows a simple Use-Case diagram for *Place Order*.

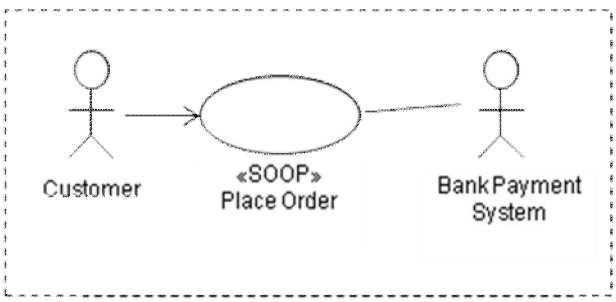

Figure 4: Use-Case Diagram for "Place Order" SUC

This system use case is implemented in the SupaStores Online Order Processing (SOOP) system, which delivers the order-entry pages to the SupaStores website, supported by an order database.

On the diagram, we've used a UML stereotype notation on the SUC to indicate the system, «SOOP», that implements the use-case. This system use case allows the customer – the primary actor – to select products into a virtual shopping trolley and choose a delivery time-slot.

Within this SUC, the customer also pays for the order using a payment card. This can't be done by SOOP alone; it needs to communicate with the banking payment system, which therefore needs to be shown as a separate, supporting, actor. (The primary actor initiates the system use case; supporting actors are called on by the system use case.)

There are a couple of modelling points to note immediately.

- First, in this case, the system use case name is the same as that of the business use case step. This is reasonable because this system use case implements pretty much the entire business activity.
- While in the Business Use-Case diagram (Fig 1 in Part 1 of this series), we showed the *Bank* – a business entity – as a supporting actor, we're now showing here the Bank *Payment System* as the corresponding supporting actor. This highlights the fact that business use cases are about *business* concepts, while system use cases are about *systems* concepts. "Well, obviously!" you say – but we suspect that a lot of the problems come from not applying this "obvious" principle.

We'll now go through each step in the business use case and ask: "What system use cases support this business activity?". Figure 5 shows the result.

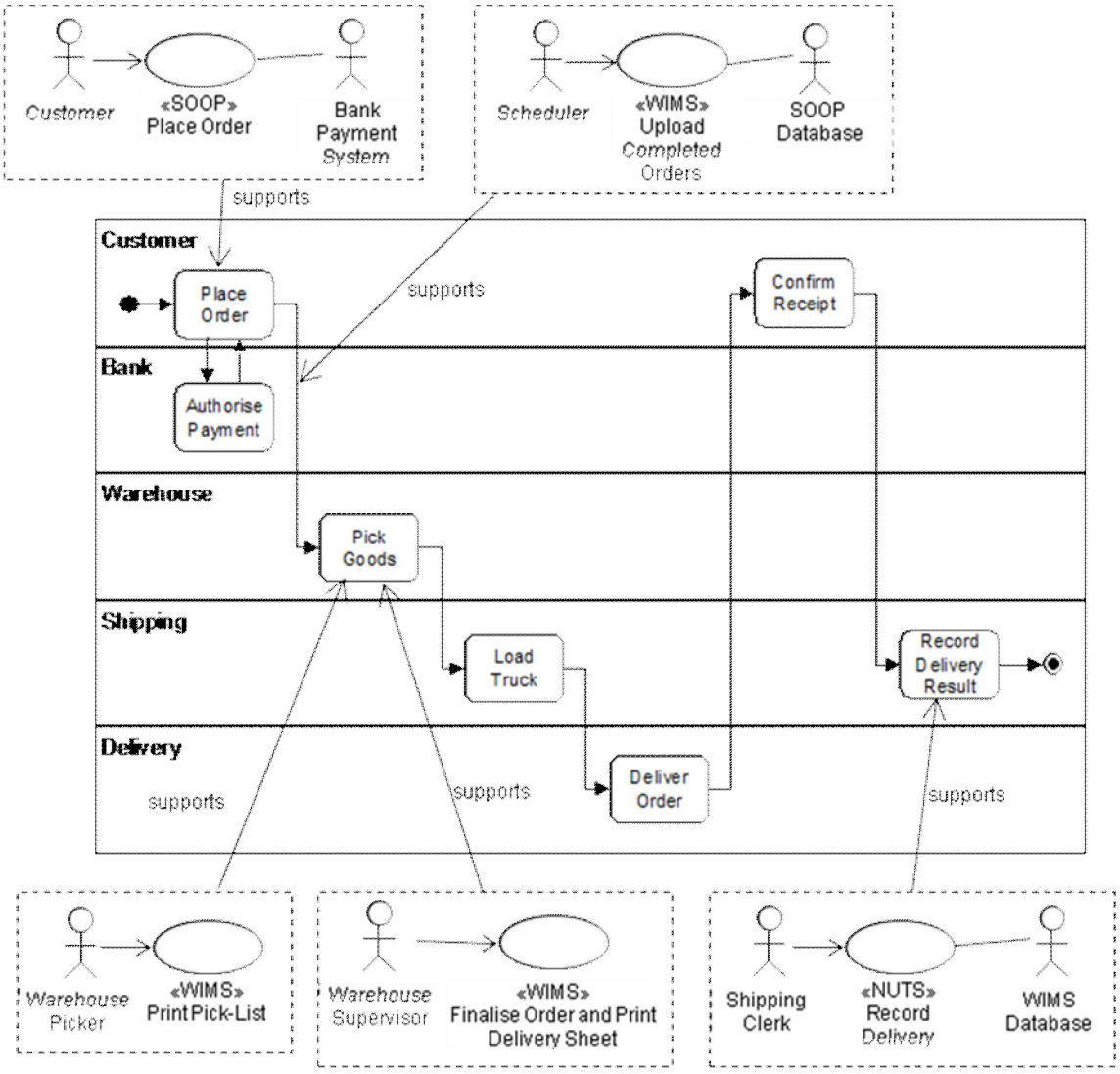

Figure 5: "Buy Groceries" Business Use-Case
Activity Diagram with supporting System Use-Cases

This is the same business use case Activity Diagram as Figure 2 (in part one of this series), but now with all the supporting System Use-Cases superimposed. The relationship between each system use case and the relevant elements of the business use case is shown by the associations labelled "supports".

Note that this is a new kind of diagram, which we call a *Process / Use-Case Support Diagram,* or *PUCS Diagram.* We'll come back to discuss this below. In the mean time, let's continue with completing the picture.

The second activity on the BUC, "Pick Goods", is performed by the SupaStores warehouse. This uses an older stock-management system called the Warehouse Information Management System (WIMS). Among many other things, WIMS allows the warehouse staff to print off a hard-copy pick-list of orders due to be shipped in the next couple of hours. This is shown by the *Print Pick-List* SUC. After executing this SUC, the picker continues with the BUC step by going round the warehouse shelves, assembling the order, and marking-up the pick-list by hand as s/he goes to show out-of-stock items, substitutions and so on.

However, there's something else that has to happen first: the order details need to get from the SOOP system to the warehouse's WIMS system. This is done by a batch job in WIMS, scheduled to run twice daily, that queries the SOOP database. The batch job is represented by the SUC *Upload Completed Orders*. This has two actors, both systems, rather than human actors: the primary actor is the Scheduler, and the SOOP Database is a supporting actor. Finally, note that the "supports" association from this SUC is not to an *activity* on the BUC; rather the SUC supports the *transition* between the Place Order and Pick Goods activities.

Now, going back to where we were at the "Pick Goods" activity: once the warehouse employee has finished picking goods for each order, s/he hands back the marked-up pick-list to the warehouse supervisor who uses a further WIMS System Use-Case, *Finalise Order and Print Delivery Sheet*. This allows the supervisor to flag in the system any items that couldn't be found on the shelves, indicate any substitutions, and print a delivery-sheet for the truck driver. So there are two SUCs that support the same BUC Activity. This might indicate that the activity needs to be split into two on the diagram (this iterative modelling between development of BUC and SUC models is typical), but we'll leave it as it is for now.

The next three BUC activities – "Load Truck", "Deliver Order", and "Confirm Receipt" – have no systems support. Route planning is done by the delivery staff and driver using their local knowledge and a street-map. The delivery sheet printed off by the warehouse supervisor provides all the information the driver needs en route, but reading and signing the paper can hardly be counted as a "system uses-case".

The final activity in the BUC – "Record Delivery Result" – does have system support. The clerk in the shipping depot records post-delivery details from the sheet into a third system, used to track completed orders and also to record utilisation of delivery vehicles, called the National Utilisation and Tracking System (NUTS). This accesses delivery data from the WIMS database, which is therefore again shown as a supporting actor.

Note that the SUCs we've listed above as supporting the BUC steps are far from being the only set we could imagine. In fact, we'll look in Part 3 at what a different set could look like, and how this would affect the overall picture.

Inside the System Use-Case

In Part One, we opened up our example Business Use-Case to see what one looks like inside. What happens when we do the same thing with a System Use-Case?

We've taken the "Place Order" SUC as the example to illustrate. Figure 6 shows the first-cut SUC Description. Again, as for the BUC before, we've shown only the Main Success Scenario or Main Flow.

Note that we could have drawn a UML-style Activity Diagram similar to Figure 2 (in the first paper of the series) to represent this SUC. However, at this stage, it would not be very interesting: just a single sequence of activities, more-or-less alternating between Actor and System swimlanes had we chosen to show them. Drawing such a diagram would, however, get much more interesting and valuable once we include alternative and exception flows.

Name : PLACE ORDER ***System Use-case***

Brief Description : In this Use-Case, the Actor uses the SOOP system to place a new order for delivery. The actor selects the required products, chooses a delivery slot from those available, and makes a card payment for the order.

Principal Actor : Customer *(*On-line)

Precondition : The Actor has successfully logged on to SOOP as a valid and registered customer

Main Flow

Step Name	Description
Trigger	Customer chooses to place a new order
Display Products	The system displays or presents the product offerings to Customer
Select Products	Customer selects as many products as wanted into the "shopping-basket", choosing from those offered, and entering the required quantity ordered of each.
Indicate selection complete	Customer indicates that the product selection is complete.
Display total price	The System calculates the indicative order price, applying multi-buy discounts, and displays the resulting total
Accept price	Customer indicates that the price is accepted.
Display available delivery slots	The System finds available delivery-slots for scheduled delivery runs covering the address held for the customer
Choose delivery slot	Customer selects the desired delivery-slot from those presented.
Present stored payment card details	The System displays details of payment cards held for the customer and requests the customer to choose a card
Confirm payment card	Customer selects one of the presented cards
Validate card payment	The System contacts the Bank Payment System which validates the card payment
Present order details	The System presents full details of the order (shopping-basket with indicative prices; delivery address; selected delivery slot; card payment details) for the customer to print if required
Email order details	The System sends an email of order details to the email address held for the Customer
End	The Use-case ends

Figure 6 : Text Description of System Use-Case "Place Order"

Superficially, Figure 6 is very similar to the Business Use-Case description in Figure 3 in part one. Both show an interaction, or dialogue, between an actor and the conceptual system that implements the Use-Case.

However – and this is critical to the central point of this article – the fundamental nature of the two Use-Cases is very different.

- In the BUC example, this conceptual system that implements the Use-Case" is the SupaStores *business* itself, whereas in the SUC it is the SOOP *system*.
- The nature of the interactions is completely different, too: in the BUC, they are "real-world" interactions involving physical people, goods, trucks and so on, while in the SUC they are probably[1] conducted via a PC screen, mouse and keyboard.
- Clearly, the duration of the BUC and SUC are very different – a few days in the first case, a few minutes in the second.

Note that, in contrast with the BUC in Figure 3, Figure 6 is a *black-box* view of the System Use-Case: it shows only the dialogue between the actor(s) and the system, with no internal implementation details. But this does not affect the fundamental points of difference.

The Key Differences

We're ready to summarise the key differences between the two concepts.

Aspect	Business Use-Case	System Use-Case
Who's the Primary actor?	Mainly a **business actor** e.g. customer; maybe other external party (regulator, shareholder) or an internal party (manager, etc)	Mainly a human user who initiates system behaviour; maybe another system, "scheduler" etc. But by definition a **system actor**
What's the use case for?	Something the actor wants to get done by using the **business / organisation**	Something the actor wants to get done by using the **system / application**
Who / what else may be involved?	May involve interaction with other **external business parties** as supporting actors	May involve interaction with **other systems** internal or external to the organisation.
What does it describe?	Describes an **interaction** involving the primary **actor**, the relevant parts of the **business**, and any supporting actor(s), in terms of their **business behaviour**	Describes an **interaction** involving the primary actor, the relevant parts of the **system**, and any supporting actor(s), in terms of their **system behaviour** . In the case of the primary actor, this means only their actions **detectable by the system,** such as making selections, supplying data etc.
How's it executed?	May involve **many organisation units**, systems (or not), technologies, manual / mental procedures etc.	Executed by **automated steps** in the system
Duration	Of **Varying duration** - May be very brief or very long-duration.	Typically quite **short duration** (Cockburn's "coffee-break" rule)

[1] We could imagine this done another way – eg. with an interactive TV and a remote control, or even, at a stretch, via a voice- and keypad-activated phone system. The key point is that the actor is interacting with a *computerised system*.

- When faced with a particular modelling task in a real-life project, these differences helps us to decide: "Which one am I modelling here?".
- Make sure you know the answer, and stick to it. Get yourself into the appropriate mindset, and don't mix the business and system views in the same Use-Case. Once you've got them sorted out, you can show how they're related using a Process / Use-case Support (PUCS) Diagram.

Requirements or Design?

We said above, in starting to discuss the System Use-Cases that support our Business Use-Case, that "we are now moving from the *what* gets done, to *how* it's done". This raises an issue that often causes problems, and in doing so typically generates more heat than light. The question that's asked is: "If we're now moving to the *how*, are we not moving from *requirements* to *design*? And if so, isn't there a conflict in that Use-Case Descriptions are supposed to be a *requirements* artefact?".

This topic has been discussed extensively (try Googling for "requirements vs design"), and we don't want to reiterate all the arguments here. But to summarise our view: Use-Case Descriptions are *both* Requirements *and* Design artefacts.

How can this be?

Bear in mind that there's a progression from coarse-grained outline business requirements through to implementation of a software solution. (And no, we're not advocating a waterfall approach here – it's OK to iterate back and forth along this progression.) At each stage, we take stated requirements and apply a range of factors – logical considerations, technical guidelines, best practices, constraints and so on – to come up with a design at the appropriate level of detail. That design becomes the requirements statement for the next level of design.

This "requirements vs design" issue is closely related to the viewpoints of the various stakeholders in a project. To a software developer, the Use-Case Description in Figure 6 might look very much like a requirements statement, and a fairly un-detailed one at that. To her, it's just waiting to be tackled with some decent design, covering UI layout and behaviour, object / component design, system integration, performance, security and so on. But to the business manager responsible for the new on-line order service – whose main concerns may be doing a deal with the right payments operator, or ensuring that warehouse and delivery people understand the impact of the new service – Figure 6 looks like pretty detailed system design.

And the thing is – they're both right!

Next time

- In part three we will look again at the Process Use Case Support Diagram and make observations and conclusions about the differences between Business and System Use-Cases.

Contact details

Martin.Langlands@blueyonder.co.uk

Charles.Edwards@processwave.com

www.ProcessWave.com and www.AgileEA.com

