
Stakeholder Communications – Pictures not Words

Jan Kusiak, Training Services Manager
IRM Training Pty Ltd ACN 007 219 589
Suite 209, 620 St Kilda Rd, Melbourne, Vic. 3004, Australia
Tel: +613 9533 2300

Overview

Many people on our *Business Analysis* workshop ask why we use dataflow diagrams (DFDs). Why not Use Case...or even BPMN? After all DFDs have been around for 20 years, surely the world has moved on?

Well, has it? The primary purpose of a business analyst is to communicate – to stakeholders and to solution providers – and when it comes to communication we all know that pictures (diagrams) are much more effective and less ambiguous than words. Remember the phrase "A picture is worth a thousand words". The question is – which type of diagram best suits our needs? In this article we'll look at stakeholder communications whilst future articles will cover communications with solution providers.

Introduction

Definitions are always a good starting point. Australia's Macquarie dictionary defines a diagram as.....*a drawing or plan that explains the operation of something.*

Analysts need to *explain something* to two sets of people - stakeholders and developers - each with their own needs. The stakeholder wants to know how the *something* functions as a business process and how it meets their business objectives or solves their problem. The developer (or solution provider) wants to know how the *something* functions logically so they can build it.

Both the stakeholder and the developer want the same thing but at different levels of detail. Ideally we need a layered diagramming technique with the top layer showing the concept while lower layers provide more detail. Our ideal diagramming technique will need to answer the following questions:

- Can it be easily understood by non-technical people
- Can it explain the business process
- Does it support layering of detail
- Can it be used as a design input by the developer

As a training provider we also have a few more questions to ask:

- Is it a proven technique that's in widespread use
- Can delegates see how it fits into the business analysis process
- Can course delegates easily use it

The last question goes to the heart of our training. Can delegates pick up the notation quickly so they can use it to solve the business problems in the case study – rather than spending time learning the notation before they can apply it?

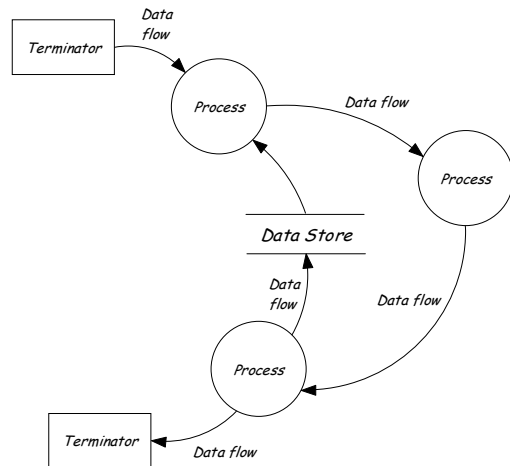
Now lets have a look at how DFDs can meet our needs. Note that we're not evangelists for one notation technique over another. We've been offering UML training for over 5 years and plan to introduce a course on BPMN techniques next year. Like all good business analysts should, we're simply asking the question – what's the best tool for the job?

Can it be easily understood by non-technical people?

Whilst nearly all business systems involve the processing of information (data), the stakeholder is usually only concerned with the logic of the business process. To draw this we use 4 symbols.

The terminator shows the boundary of the system and may be a person, a department, a customer, another system. Terminators are usually only used on the highest level diagram but this is not compulsory. The highest level diagram is called a context diagram, see example below.

The arrow shows the movement of data and the process bubble shows where data is transformed (acted upon). The data store shows data at rest (database, filing cabinet).



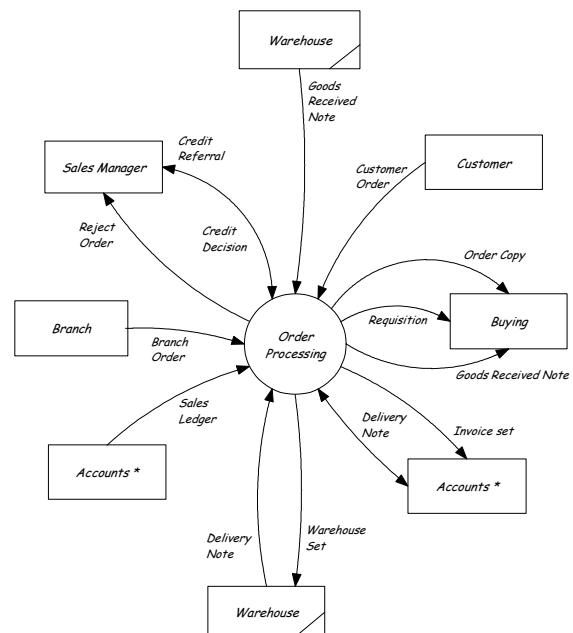
It's a simple notation, but one that has been used successfully for over 20 years and one which most people can grasp intuitively.

Can it explain the business process?

At the highest level of DFD (called the context diagram), we show the boundaries of the project or system.

In this diagram there is only one process bubble, the order processing system. The boundaries of the system - customers, warehouse, branch office etc. are shown outside the bubble.

These boundaries (terminators) send and/or receive data from the system. They are it's users and can be departments, individuals or other systems. For our purposes however we're interested in what's going on inside the order processing bubble. We want to know how the system works.



We also need to distinguish between two viewpoints - physical and logical. A physical diagram is most often used to represent how the system currently operates. It is often used to document (and to help people understand) the operations of the current business system. Note that a business system can be a mixture of automated (e.g. computerised) and manual processes. We can represent both in the one diagram.

Logical models allows us to show how we would like the system to operate – the “to be” system. Logical modelling is a powerful tool for re-designing a current business system or specifying how we want a new system to function because it focuses purely on the business logic that’s required. Logical modelling also helps us to avoid getting sidetracked by implementation issues (e.g. what database to use, is it web services or client/server...etc). Remember...*don’t design the solution before you’ve analysed the problem.*

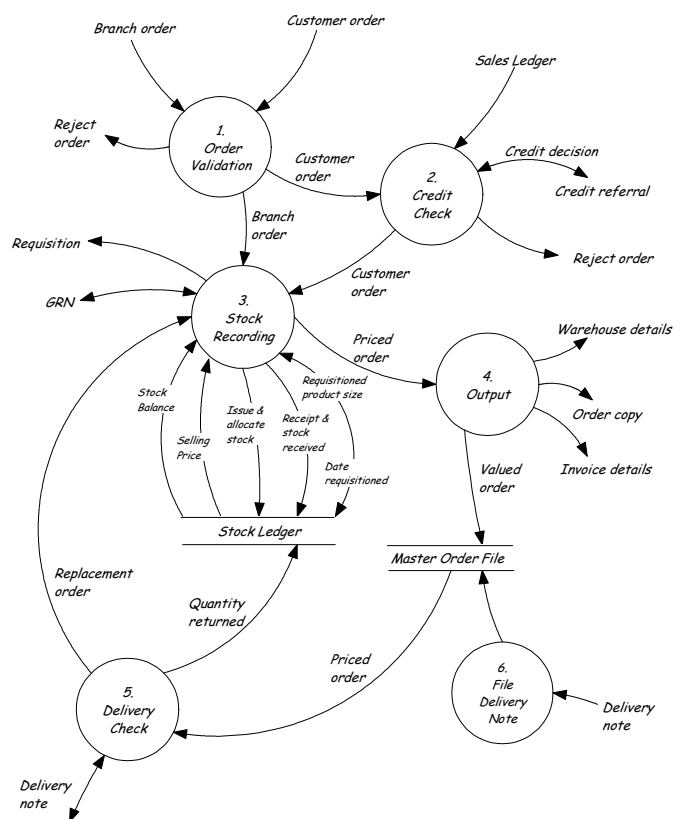
Does it support layering of detail?

Order processing can consists of many tasks. In this example our system consists of the following processes:

- order validation
- credit check
- stock recording
- output
- Checking delivery
- recording delivery

We show the logical sequence of activities by numbering our processes and assign meaningful names to them – and to data flows – so they can be more easily understood.

Again, we are controlling the amount of detail so as not to overwhelm our stakeholders. People can only take in so much information and a general rule of thumb is about 7 process bubbles on a page. Because we are representing what happens logically in the system it is perfectly valid to combine processes to hide complexity. A modern order processing system can consist of 20 or more processes (or modules if describing a software application). Grouping several modules under one process name hides complexity and aids in understanding -whilst still allowing us to accurately represent the functionality of the system.



Context diagrams are the highest level diagram and show the boundaries of the business process. This diagram here shows the next level of detail – the major processes of the system - and is called a level 0 diagram. We can continue our hierarchy of diagrams (levels 1, 2, 3, 4..etc) each time showing more and more detail of how the business process functions. Generally stakeholders and users only want to see the first few levels but we can continue with our diagrams until we get to yes/no statements (or binary for the technically minded!). At each level of diagram we are following the same rules for drawing DFDs – from stakeholders at the top to CPU instructions at the bottom.

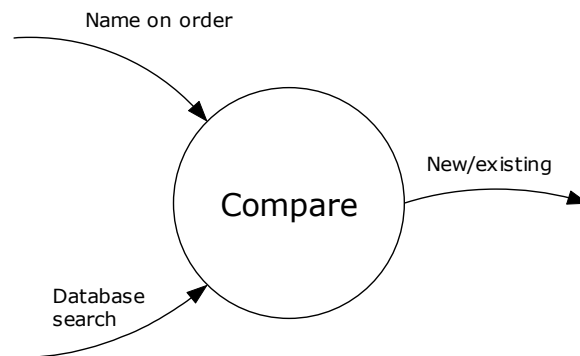
Can it be used as a design input by developers?

At no point so far have we specified how the system will be built. We are describing how our business system will function logically. System architects and developers will decide how it’s

built based on company standards and available technology. What they do need to know is what is the business logic they must deliver.

At the lowest level of DFDs we reach the functional primitive, the point at which a process cannot be partitioned further. Here we show the transformation of data. The developer now has something they can code because there are no questions left to answer –there is only one way the business function can be interpreted. To illustrate this we go back to our order processing system. As part of the order validation process we need to check if the order is from a new or existing customer.

We can draw a functional primitive to represent this and be assured that the developer will understand exactly what we want. At no point do we need to specify how to write the code, whether to use Cobol or Java etc. We only need to tell them what business logic we want performed. At this level we are asking a simple yes/no question. We know that this question can be plugged into a higher level diagram – all the way up to our context diagram.



Is the technique proven and in widespread use?

Dataflow diagrams are common to many structured techniques, the most well-known of which is SSADM (Structured System & Design Method).

SSADM was commissioned in 1981 by CCTA (the UK government IT agency also responsible for ITIL and PRINCE2) in order to standardise project development across government departments. The methodology was expanded in the 1980's and 90's to cover newer technologies such as interactive user interfaces, 4th generation languages, client server, distributed applications and object-oriented design. By the mid-1990s it was the most widely-used application development methodology in Europe, the USA and Australia. To quote the UK Government "*The purpose of SSADM is to assist with the development of well engineered systems by employing a number of mature techniques and approaches to produce clear and unambiguous specifications of what is to be developed.*"

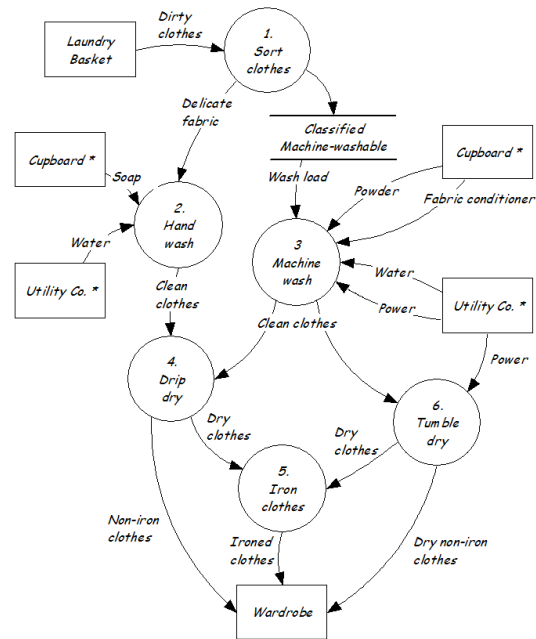
SSADM is often associated with the waterfall development methodology (more on this in a future article) but as it's a collection of tools and techniques you'll find bits of it cropping up on all sorts of projects - iterative, agile, prototyping, spiral... etc. Organisations often use a mixture of techniques on the one project – it's what works for them that's important.

DFDs were around long before Visio, PowerPoint and other desktop drawing packages. Diagramming notations had to be simple so people could draw and understand them. The effectiveness of DFDs has been proven by the thousands of projects which have used them. Structured techniques and DFDs are widely taught at tertiary level and most graduates will cover them in any information technology degree course. That DFDs are still widely used today is a good indication of the robustness of the technique.

Can course delegates easily use it?

Using only 4 symbols in a context diagram and 3 (or at most 4) symbols for every other diagram is well within the grasp of most people today.

Everyday events can be drawn easily using a DFD. Even if you found the preceding example on the order processing system a bit dry, you shouldn't have any problems interpreting this diagram – unless you're one of the few people who has never had to do household chores!



Putting it all together –the business analysis process

A business system is more than just dataflow diagrams however. To get a comprehensive view of the system we need:

- a model of the process logic (DFD)
- business rules defining the process logic
- a model of the data needed to support the process
- a definition of the data in the data model

Together, these techniques give us a powerful set of tools for analysing, modelling and designing business processes. The business analyst may only use DFDs for stakeholder communications but they will need to understand business rules and data if they are to communicate effectively with developers and solution providers. We'll take a closer look at this in a future article.

© 2002-2007 IRM Training Pty Ltd. All rights reserved.

Send feedback and comments to: training@irm.com.au

You may use this article in your newsletter or internal document free of charge provided that you do not alter it in any way and that you include the following:

By Jan Kusiak, ©2002-2007 IRM Training Pty Ltd ACN 007 219 589, www.irm.com.au