Majik Consulting™

"Your Technology Specialists"

Business Analysis

Whitepaper

Data Flow Diagrams and Use cases

Copy No:

Document – No Unauthorised Copying

Revision: 1.0 **27 December 2008**

DOCUMENT CONTROL SHEET

Contact for Inquires and Proposed Changes

If you have any questions regarding this document contact:

Name: Ronald Kohlman

Designation: Consultant

Phone: 03-9017 3184

Web: www.majik-consulting.com

Document Path & Name: ***C:\Majik Consulting\Majik Consulting Whitepaper Series Vol 2.0.doc***

Printed: **9 January 2011**

Record of Issues

Issue No	Issue Date	Nature of Amendment
.01	27.12.06	Initial Version

Majik Consulting Whitepaper Series

Data Flow Diagrams and Use Cases

Release Version 1.0

Written by

Keith J. Majoos, MACS

B. Sc. Diploma Datametrics, MBA, Grad Diploma Psychology

Ronald Kohlman

B. Bus (Business Administration)

TABLE OF CONTENTS

Majik Consulting Whitepaper Series

Vol 2.0 December 2006

Topic: Data Flow Diagrams and Use Cases

1 Objective

This **Majik Consulting Whitepaper Series** paper is to give you the reader, a view of Data Flow Diagram and Use Case approaches. Data Flow Diagrams are an excellent means to create functional views from a system's perspective. Use Cases provide functional views from an Actor's perspective. However, there are times when some confuse the two approaches and may create functional decomposition of use cases. Don't!

The purpose of this paper is to alert you to the pitfalls of using DFD techniques like "functional decomposition" with use cases; and also to show that DFDs are still good and useful techniques.

I assume that you know something of DFDs and Use case techniques.

2 Vignette

For this paper, we'll use the following vignette or small part of requirement. We'll consider a library that does everything manually and that wants to automate their processes. Assume that we've interviewed the librarian who told us the following:

> *"A prospective member fills in an application form. The librarian processes the application form and issues a library card (hereafter the prospect is a member). The member browses a catalogue and finds a loan item (Books, Magazines, Videos, DVDs, and Cassettes). The member borrows the loan item by giving his/her library card plus the loan items to the librarian. The librarian records the borrowings and returns the library card and loan items to the borrower. At some future date, the borrower returns the loan items to the library. The Librarian checks in the items. The member may also reserve a loan item or arrange an interlibrary loan from another library. When these items become available at the library, the requesting member is notified.*
>
> *The librarian, who is also a member, maintains the catalogue. The librarian gets a list of the latest items from distributors during the month. At month end, the librarian creates an order for the distributors. When the items arrive the librarian labels them and makes them available in our catalogue"*

Read section 5 first. This is all about Use cases, which are generally disjoint: a swag of functions grouped together in a system (subject). See how you go, and then read section 3 and read section 5 again.

3 Library DFDs

Data flow diagrams have 4 main components: There is the external entity (denoted by square), the data store (denoted by C shape), the process (denoted by a "sqound" – square with round edges), and the data flow between processes, between processes and data stores, and between external entities and processes (denoted by arrows)

For those of you who are interested in DFDs you may still find some good literature by Yourdon, Gane and Sarson, De Marco, Constantine and others.

3.1 Level 1 Diagrams

Figure 1 is not complete, but it depicts the essentials of the vignette above.

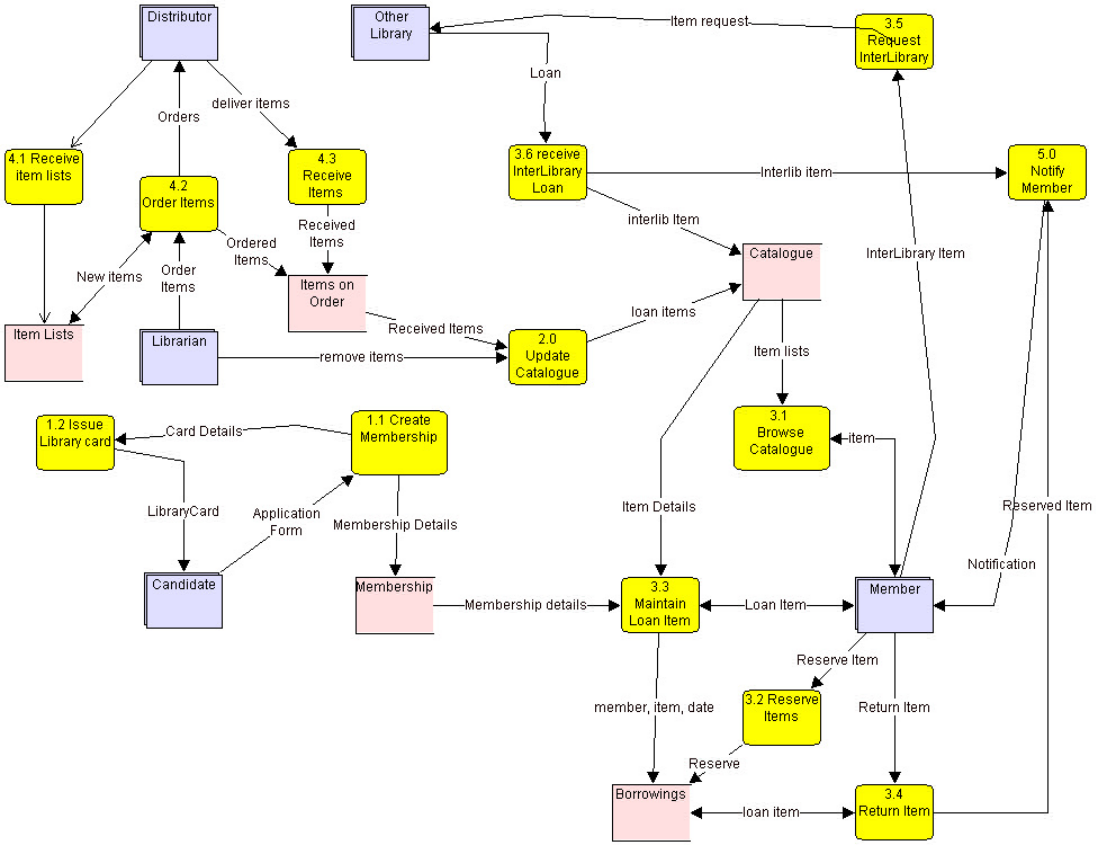

Figure 1: Level 1 data flow diagram with external entities

3.2 Level 0 Diagrams

If someone were to ask you "what are the main functional groupings for the new system?" you might aggregate the processes into 4-5 major processes:

Maintain Membership – This process includes everything to do with creating the initial membership, the issue of the library card and the management of subsequent changes to membership details.

Manage Borrowings - This process includes browsing the catalogue, reserving items, requesting inter library loans, notifications and returns.

Maintain Catalogue - This process includes deleting items, receiving new items and updating the catalogue

Manage Orders - This process includes receiving new item lists, ordering from distributors (suppliers) and receipting ordered items.

Manage Notifications - This process includes notifications when a reserved or inter library loan items becomes available.

The above aggregations are depicted in figure 2 below. Flows 1-7 denote data flows between internal processes and external entities.

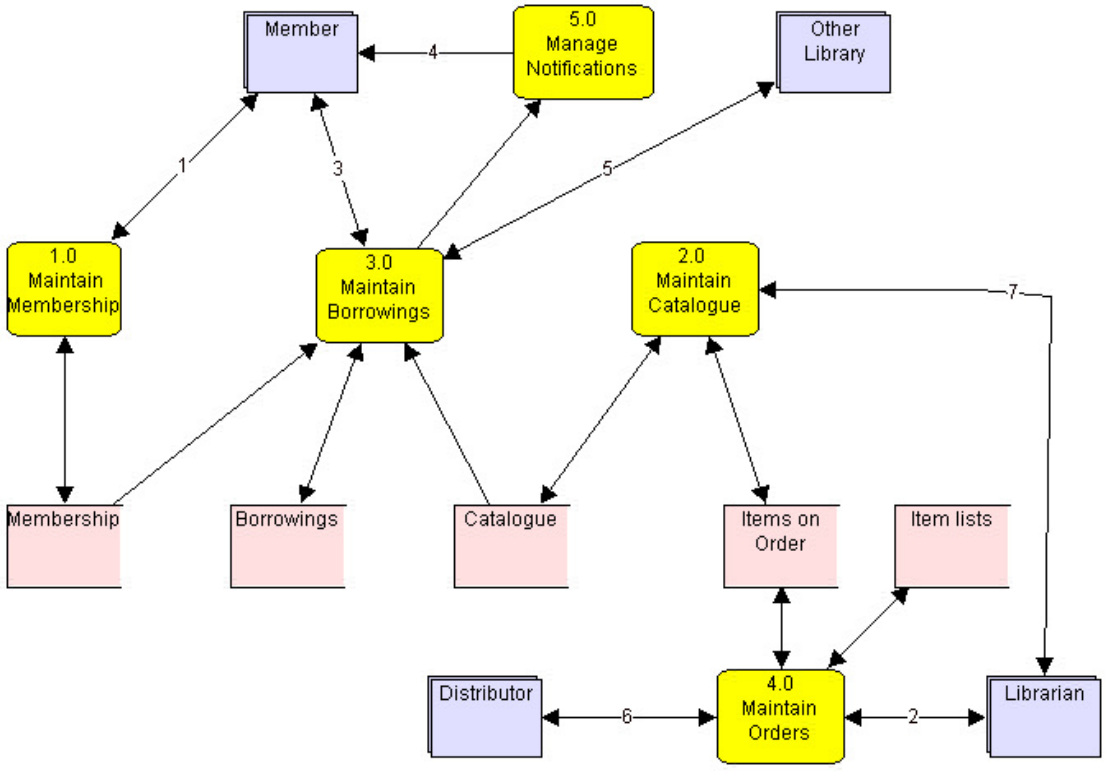

Figure 2: Level 0 data flow diagram with external entities

3.3 Context Diagrams

When we present DFDs, we normally start at the context diagram. I've deliberately started in the middle somewhere, because drawing a context diagram from scratch is difficult. Figure 3, depicts a context diagram.

Functional decomposition is when we start with a context diagram, then decompose the context into its 4-5 sub-functions. These functions in tern are then decomposed into a level 1 diagram, which intern is decomposed into a level 2 diagram etc. You continue this process until you have "primitive" processes. Ie process that cannot further be decomposed. Once you've reached these primitive processes you start writing (mini) specifications for each.

Creating DFD is a bit of an art form. You start at a level 1 or level 0 and work your way up to a context diagram (bottom-up approach). Once your context diagram and level 0 diagrams are well established, then you decompose functions into more details (top-down approach). Alternating between top-down and bottom-up approaches is called levelling.

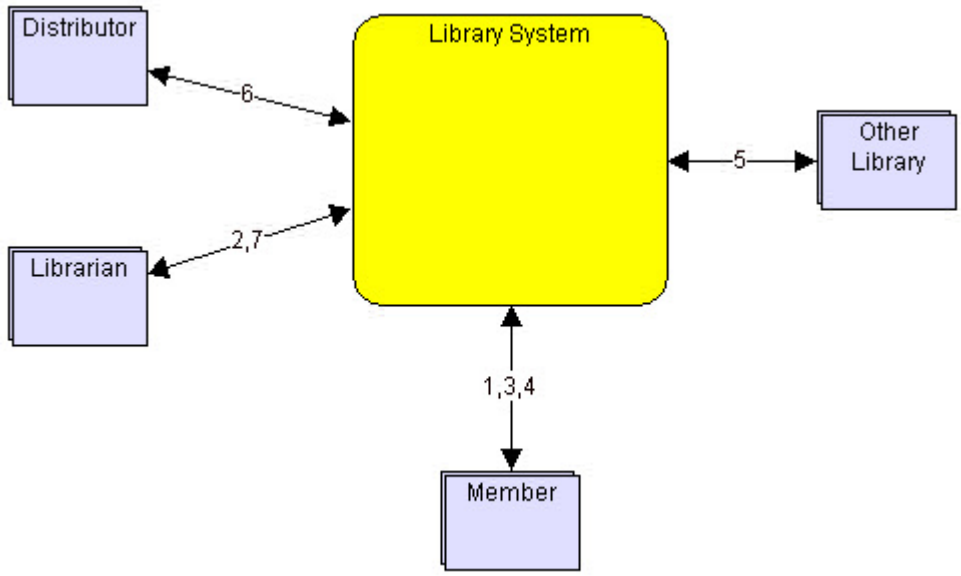

Figure 3: Context diagram

Flow	Description
1	• Membership form • Library card
2	• Order Items
3	• Item (for browsing) • Reserve Item • Loan Item • Return Item • Notification
4	• Notifications
5	• Interlibrary Request • Loan Item
6	• New item lists

Flow	Description
	• Orders • Delivery of items
7	• Remove Items

Decomposition and levelling work well for DFDs but not for use cases.

4 Functional Decomposition

You have noticed that functional decomposition and levelling are good strategies for DFDs to get to primitive functions. However, some people apply this technique to Use Cases and **it is WRONG to do so**. I'll illustrate then show you the correct way to utilise use cases for requirements recording.

The stick figure depicts the role of a user. The oval depicts the system function. The relationship between user and function is depicted by a straight line (no arrows). The relationship between functions has a dotted line and arrow.

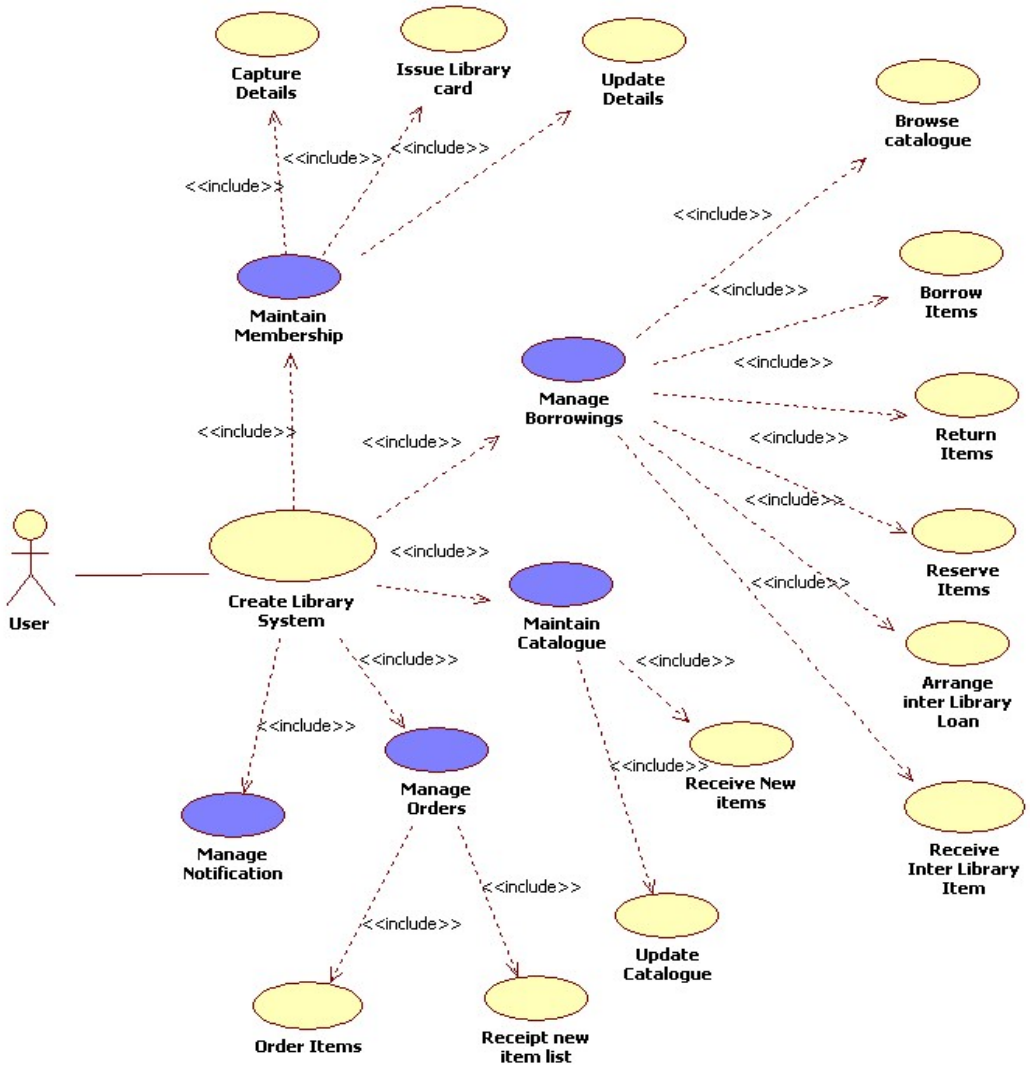

Figure 4: Create Library System (Functional Decomposition)

Although this might look OK, it is not! Take care not to create this kind of diagram and associated narrative for a system. Its obvious from this diagram (figure 4) that create library system is decomposed into five sub-functions. Each of these sub-functions is then further decomposed into sub-sub-functions.

A decomposition strategy works well for DFDs but NOT for use cases.

See http://www.martinfowler.com/distributedComputing/abuse.pdf.

5 Use cases

A use case is "A description of sequences of actions, including variants that a system performs that yield an observable result of value to an actor. " (Booch et. al., 1999 *Unified Modeling User Guide*)

5.1 Define actors

When creating use cases, one of the first things we do is to identify the Actors of the system. We note that we have the following actors:

- Member (also known as prospective member)
- Librarian
- Other Library
- Distributor (also known as Supplier)
- Time (at month end)

Note: not all actors are people. Some actors can be systems.

5.2 Define actor goals

An actor's goal is something of value that the actor expects from the system (in our case the computer system). The member is the one who borrows an item. However, it is the librarian who uses the system most. The member usually browses a catalogue, goes to a shelf, picks an item (book), then gives the item and library card to the librarian.

Member -	Applies for membership**
	Browses a Catalogue
	Receives a Notification
Librarian -	Maintains Membership
	Issues Library Card
	Creates Borrowings
	Updates Returns
	Receives an ordered item
	Updates the catalogue
	Reserves an Item
	Receives an inter library loan item
	Send notification
	Receives New item lists**
Other Library -	Request a loan

Distributor - Sends an Order (**Time**)

I've marked the "**Applies for membership**" use case with ** to indicate that it is special. If the subject is a new computer system then filling an application form and giving it to the librarian, is not a computer system use case. (More about this boundary/subject issue in the course). **Receive New Item lists** is also a manual function (a paper catalogue)

Other library and **Distributor** are two unique actors that interact with the new library system. That is, the new system will communicate with the external systems of **Other library** and **Distributor**.

5.3 Use case diagram

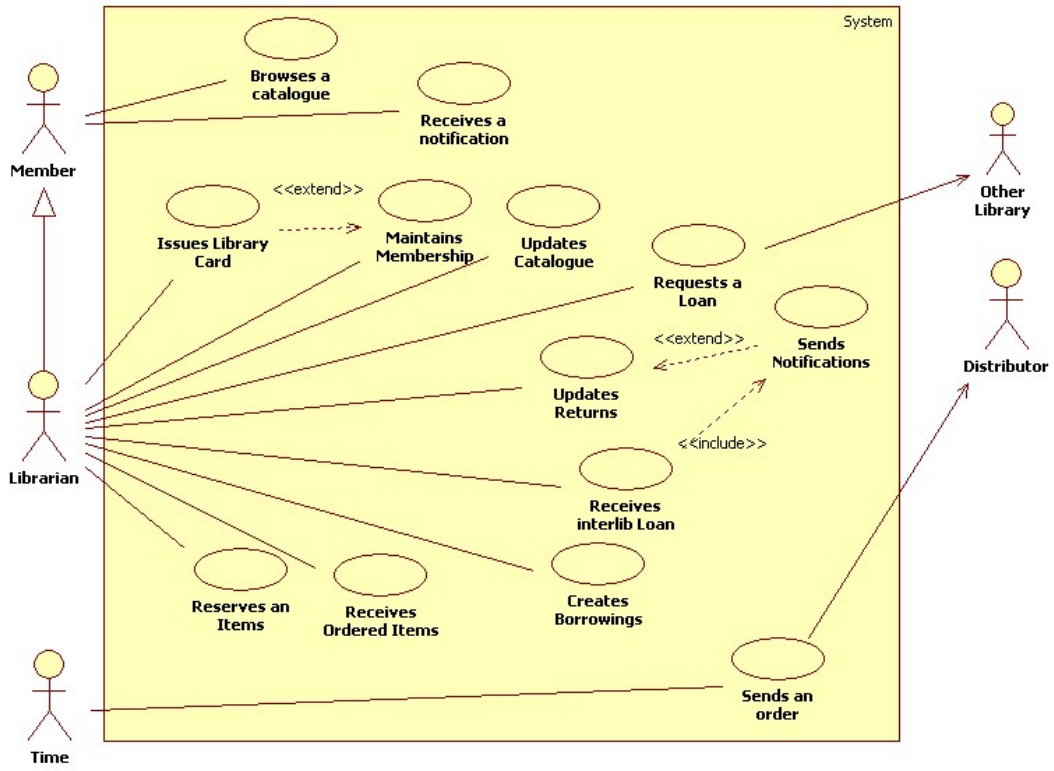

Figure 5: Library System use cases

Notice that the "**send notifications**" use case is used when the librarian "**receives an interlib loan**" the member is notified. Similarly, when a reserved item is returned, the "updates returns" use case will **send notifications** to a member, if the member reserved the item. In the member "**receives a notification**" use case, we would specify various means to notify people. For example mail, e-mail, phone call, or automated phone call.

My library uses a computer text-voice application that calls and reminds me that my items are late or overdue.

6 Can we still use DFDs?

DFDs are still relevant, just check out the AGILE sites. You can still create a single DFD where you may have to create multiple use cases and perhaps an activity diagram to provide insight into the process.

Let's use the following section of the vignette above:

> A prospective member fills in an application form. The librarian processes the application form (i.e. creates a member) and issues a library card. *There are times when the librarian has to re-issue lost or damaged library cards; and update member details like address changes etc*.

I've added some extra requirement, which are italicised. This elaborates a bit more on the issue of library cards. The DFD at 6.1 below shows how one could use the traditional DFDs to record the process. First, it is easy to read as it shows the process of creating a member, issue a library and manage changes to membership details.

6.1 DFD

The following DFD illustrates the vignette above. The processes in the diagrams have been modified slightly; I've added the "actor" to each process. Using 'old' DFDs in this 'new' way makes it easier to capture processes. I've also drawn the "automation" boundary around some of the processes, indicating which processes will be implemented on the computer system.

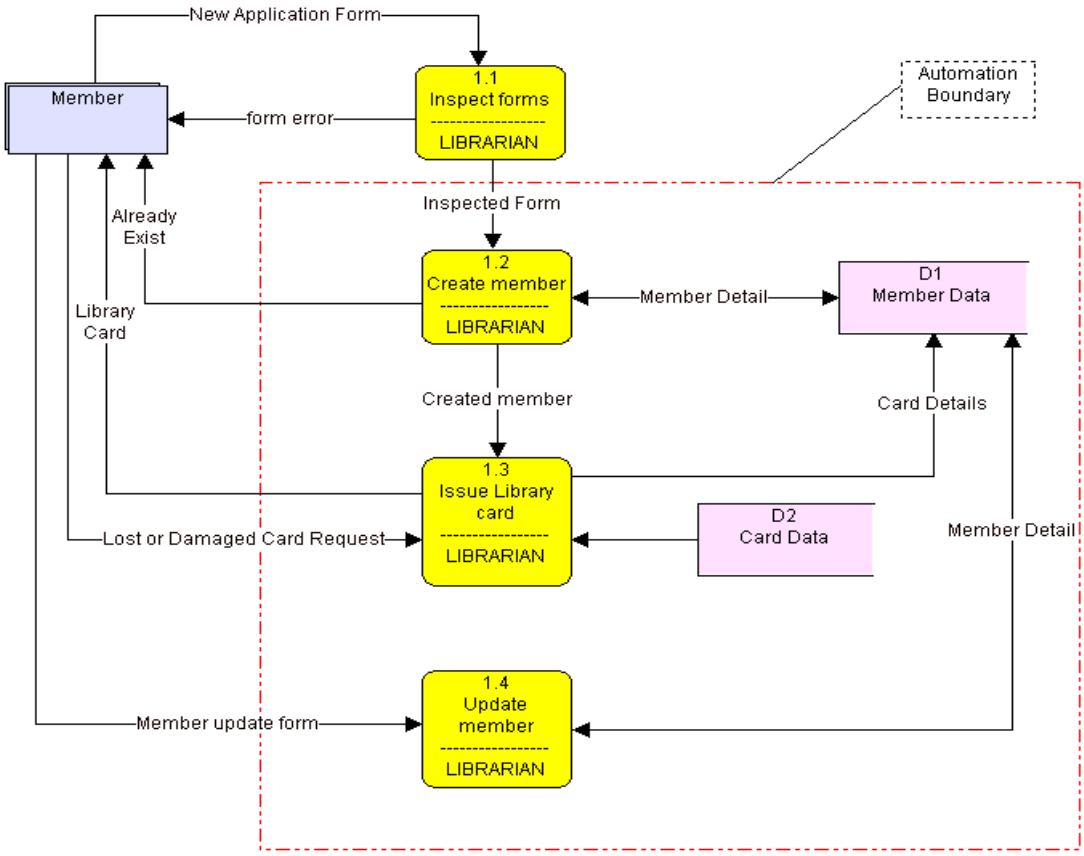

Figure 6: DFD with automation boundary.

This figure shows a little more details; in particular, it shows the process starting with an application form. Not all processes will become system functions; some will remain "manual process". However, those that will become part of the system are encapsulated in an "automation boundary" (see above). The processes are "primitive" enough for one to write a mini spec.

eg. "1.2 Create Member" could be written as the following mini-spec:

```
For every inspected form
     The LIBRARIAN creates a new member entry in (D1)
     MEMBER DATA.
     If the "MEMBER already Exists" tell the member.
End For
```

eg. "1.3 Issue Library Card" could be written as the following mini-spec:

```
For every lost or damaged card request
     The LIBRARIAN gets the next available card from
     (D2) CARD DATA.  (This could be a box of cards)

     The LIBRARIAN gets the old card details
     (note, if there are items and fees outstanding,
     then these items and fees need to be reassigned to
     the new issued card; and the card history is
     transferred to the new card)

     Find the member details in (D1) MEMBER DATA and
     update it with the replaced card details.

End for

For every created member
     The LIBRARIAN gets the next available card from
     (D2) CARD DATA.  (This could be a box of cards)

     Find the member details in (D1) MEMBER DATA and
     update it with the card details.

End For

Issue a library card.
```

6.2 Use Case

We could use a use case to capture the requirements for creating a member. In this case we may decide to have a use case for:

- Maintains Membership
- Issues a library card

When reading the diagram below, you notice that the librarian also issues a library card for first time members. The librarian can also issue a library card in the case when a card is lost or damaged.

The two use cases above capture almost the same “story” as the DFDs in section 6.1. It is not immediately obvious from the diagram that the member fills in an application form and gives it to the librarian. I’ve captured this precondition in the “maintains use case” below. This kind of information is sometimes recorded in an activity diagram.

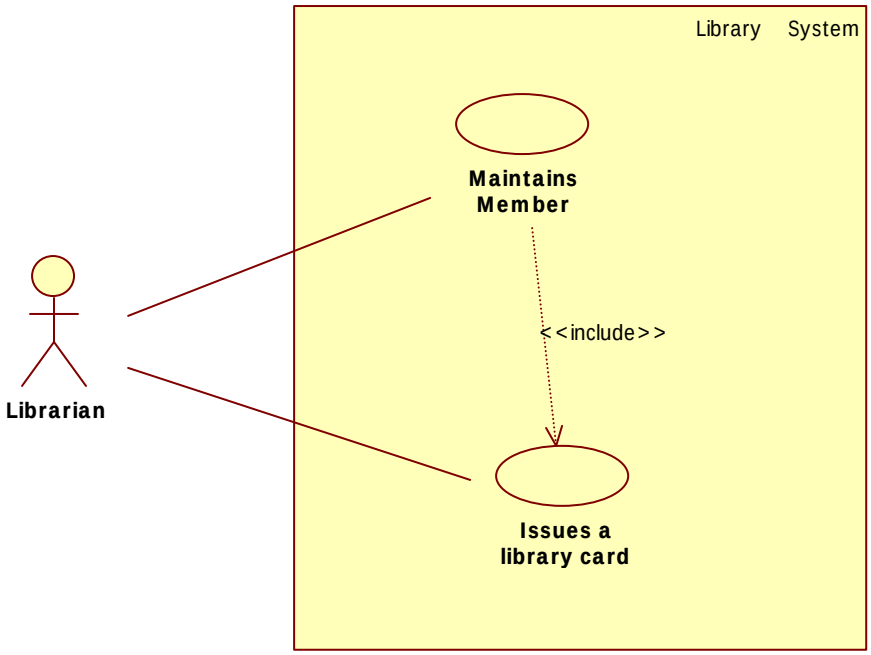

Figure 7: Use Case Diagram

6.2.1 Maintains Membership

Actor: Librarian
Level: User Goal.
Description: The librarian creates a member
Precondition: The application form contains the correct information
Main flow:
1) This use case starts when the Librarian selects “Maintains Membership” from the main menu
2) The Librarian enters Name, Address details from the Application form.
3) The system records the membership details.
4) The librarian issues a library card.

Post Condition:
The system recorded the member details
Alternate flows:
From 3), if the member exists, the system responds with “Member already exists”.

I’ve introduced the main flow only. This use case will change when we include inquiry, update and delete options.

Note: Where a use case references another use case, the referenced use case is underlined in the text.

6.2.2 Issues a library Card

Actor: Librarian
Level: User Goal.
Description: The librarian issues a library card
Precondition: The member exists
Main flow:

1. This use case starts when the librarian selects “issues a library card” from the main menu
2. The librarian gets the next card from a pack.
3. The librarian enters membership number and library card number.
4. The system records the library card.

Post Condition:
The system recorded the library card details
Alternate flows:
From 3) if this is a re-issue of a library card, (due to loss or damaged card), The LIBRARIAN gets the old card details and, if there are items and fees outstanding, then these items and fees need to be reassigned to the new issued card; and the card history is transferred to the new card.

7 Summary

When reading use case diagrams without a context it is difficult to form a process view because use cases are disjointed.

DFDs on the other hand give a process view of the system; further decomposition of processes provides a functional view.

Avoid using functional decompositions with Use Cases.

To further learn about the process and how to use UML 2 to create your specifications contact us at www.majik-consulting.com

Some planned papers in the Majik Consulting Whitepaper series:

- Creating user acceptance specifications
- Process Workflow and BPMN/UML2
- Creating Terms Of Reference (TOR)
- Planning your testing
- User Acceptance Testing overview

Note: Some diagrams in this paper were created using the FREE StarUML tool at www.StarUML.com.

--- end of document ---

To further learn about our services contact us at www.majik-consulting.com

