Portumna Community Library

Library Management System

SwiftSolutions Ltd.

Minimise Costs, Maximise Quality

CS4125 - Systems Analysis

Name	ID	Course
Colin Deasy	0436909	Multimedia & Computer Games Development
Clement O'Donnell	0810377	
Damien Hogan	9926275	Multimedia & Computer Games Development

CS4125: Systems Analysis
Assignment 1: Semester II, 2008-2009

Name:	ID:
Name:	ID:
Name:	ID:
Name:	ID:

	Item	Detailed Description	Marks Allocated		Marks Awarded
			Sub-total	Total	
	Presentation	• General Presentation • Adherence to guidelines i.e front cover sheet, blank marking scheme, table of contents		**2**	
4	Narrative	Narrative description of business scenario		**1**	
5	Software Lifecycle	Is it linear (Waterfall) or iterative (RUP). Discuss risk management strategy?		**1**	
6	Project Plan	Plan specifying timeline, deliverables, and roles.		**1**	
7	System Architecture	System architecture diagram		**1**	
8	Requirement	• Use case diagram(s) and structured use case descriptions(s) • Non-functional (quality) attributes • Screen shots / report formats	3 1 1	**5**	
9	Analysis	• Method used to identify candidate classes • Class diagram with generalisation, composition, multiplicity, dialog, control, entity, interface classes, etc. • Two communication diagram • Sequence diagram • State chart with annotated transition strings • Entity relationship diagram with cardinality	1 3 1.5 0.5 1 1	**8**	
10	Design	• Description of an architectural or design pattern that was evaluated. Cannot use MVC, Broker and Singleton. • Pattern incorporated into class diagram • Refinement of class diagram from analysis to include MVC architectural pattern, and collection classes • Refined interaction diagram from analysis with MVC elements	3 1 2 1	**7**	
11	Data dictionary	Fragments to illustrate different artifacts created during requirements, analysis and design.		**1**	
12	References			**1**	
14	Online Assessment	Week 8: Use cases Week 10: Class and Communication Diagrams from analysis phase	1 1	**2**	
		TOTAL		**30**	

Contents

1. Project Description 1

2. Requirements Summary 3

3. Software Lifecycle Model 4

4. Project Plan & Allocation of Roles 6

5. System Architecture 7

6. Requirements 9

6.1 Use Case Diagrams 9

6.1.1 Stock Management Use Cases 9

6.1.2 Member & Loan Management Use Cases 10

6.2 Use Case List 11

6.3 Use Case Descriptions 12

6.4 Detailed Use Case Descriptions 15

6.4.1 Member Management 15

6.4.3 Stock Level Management 19

6.4.4 Staff Management 22

6.5 Detailed Use Cases 25

6.5.1 Create Loan Use Case 25

6.5.2 Extend Loan Use Case 27

6.6 Non Functional Requirements Discussion 29

6.6.1 Modifiability 29

6.7 Prototypes 31

6.7.1 Sample G.U.I. 31

6.7.2 Sample Reports 32

7. Analysis Diagrams 34

7.1 Identifying Candidate Classes 34

7.2 System Class Diagram 35

7.3 Create Loan Communication Diagram 36

7.4 Extend Loan Communication Diagram 37

7.5 Loan State Chart 38

7.6 Entity Relationship Diagram 39

8. Design Diagrams 40

8.1 System Class Diagram 40

8.2 Member Management Package 41

8.3 Loan Management Package 42

8.4 Stock Management Package 43

8.5 Staff Management Package....44
8.6 Database Management Package....45
8.7 Control Package....46
8.8 Boundary Package....47
8.9 Return Loan Sequence Diagram....48
8.10 Return Loan Communication Diagram....49
8.11 Create Item Communication Diagram....50
9. Design Discussion....51
9.1 Design Patterns Used....51
9.1.1 State Pattern....51
9.1.2 Factory Method Pattern....51
9.1.3 Singleton Pattern....52
9.3 Operation Specification Using O.C.L....53
9.4 Architectural Pattern Considered....55
9.4.1 Model View Controller Architectural Pattern....55
9.4.2 Model View Controller Class Diagram Fragment....56
9.4.3 Model View Controller Sequence Diagram....57
9.5 Architectural Considerations Summary....58
10. Data Dictionary....60
References....64

1. Project Description

Introduction to Portumna Community Library.
Portumna Community Library was created in 1986 after the failure to secure a main government funded library facility for the area.
It is a non profit organization which derives its funding in the main from state assistance which is supplemented by local fundraising initiatives. There are one permanent and two part time staff members. The library lends books, audio (books/CD) and Video/DVDs items to its members.

Whilst the member and stock base of the library have grown over the years, funding only allowed for ad-hoc improvements to the manual system of management. However in 2008 one off funding was obtained from the government to allow for the purchase of a computer system with software capable of managing the basic functions of members, stock and loans.

Swift Solutions was set up in 2006 by Damien Hogan, Colin Deasy and Clement O'Donnell to develop software solutions primarily for small organizations. Their focus is on software with a low ongoing maintenance cost, this in part being achieved by their software being developed using "free to market" tools (i.e. java code using the MySQL database) which frees clients from any future software licensing costs.
Swift Solutions retains the rights to all software and may use the developed generic solution for future client requirements.

Portumna Community Library existing systems.
The library uses primarily manual paper based systems of management which have been supplemented by the purchase of a personal computer running the Windows operating system. A Microsoft Access database was purchased which is used by the one permanent staff member to record member and stock data, and to allow limited browse facilities. However the loan management is still done via manual paper means.

The library want a very simple to use system which will allow both full time and part time volunteer staff to manage member, stock and loan agreements in an integrated fashion via an easy to follow screen interface.

Ideally the system should allow for storing barcodes against both member and stock data to allow for the future use of scanning devices should funding become available.

Business Activities
Portumna Community Library allows all locally based people to use its services.
There are three types of member: Staff member, Adult member and Child member.
Adult members are allowed more items on loan for longer periods than children.
Staff members differ to both adult and child members by not being bound by loan return deadlines and related charges.

Anyone wishing to become a member must present themselves at the library and have identification with proof of address. Children must be accompanied by an adult who will guarantee by signature any potential costs associated with overdue item fees etc.

The librarian will take their details, (names(s), address, Telephone, email, member type, Child guardians name etc) and will write their member number on a pre-printed library card. This card must be presented in all future dealings with the library.
These details are stored in paper member cards and updated to the access database.

Periodically new items are purchased/obtained by the library. Each item has its details (i.e. for a book the Title, Author, ISBN code etc) and stock quantities updated onto the access database. Note that all stock items will be assigned a unique library code. The physical item (regardless of type) will have a paper card taped on the inside of its cover which will be used to record the issue of the item out on loan. These cards are replaced as they become full throughout the items loan history.

On occasion library items are sold or swapped between similar organizations to maintain a stock turnover for its members. All monies are reinvested in stock items.

An item loan is initiated by a member taking the item to the staff counter. Staff first validate from their paper filing system whether the member has already the maximum items borrowed or owes fees on any outstanding loan item(s). If so the librarian informs the member that they are unable to loan any more items until the current loaned item(s) are returned and any relevant fee paid.
If all is well with the member's loan status, the librarian will stamp the "Loan History Card" on the inside of the item cover with a "loaned out" date and a "Return" date. The librarian will update the transaction onto their paper filing system within the member "Loan history card" and may update the access database in a similar fashion.

On return of an item by a member, the librarian will qualify (via data recorded on the "Loan History Card") whether the item is overdue and charge an appropriate fee.
Until the fee is paid no further item loans will be allowed to the member.

2. Requirements Summary

1. To record details of the library members and the loans to those members.

1.1 To record new members name, address, and other relevant details.
1.2 To update members details.
1.3 To suspend a members account.
1.4 To delete a members account.
1.5 To change a members type (Staff, Adult & Child)
1.6 To browse for member details and related loan history
1.7 To record details of each loan for each member including loan period.
1.8 To extend the period of a loan for a member.
1.9 To record loan return and calculate any overdue fees.
1.10To show a members outstanding balance and record payments.
1.11Ability to view overdue loans using multiple search criteria

2 To provide the Librarian with a means to record items available in the library.

2.1 To record new stock item details including title, genre, type etc.
2.2 To manage the stock level (number of item copies) for each item.
2.3 To browse for stock items & allow view of relevant data.
2.4 To delete an item record from the system.
2.5 To check for item availability.
2.6 To reserve a stock item for a member.

3 To record library staff details and their authorisation to the system.

3.1 To maintain staff records and control their authorisation to the system.

4 Non Functional Requirements

4.1 To backup data every day.
4.2 To allow the system to be deployed on the existing library computer and be easily redeployed on a new computer at a later date.
4.3 To allow items to be imported from the existing MS Access system.
4.4 To allow new item and member types to be added in the future with minimal modification of the existing system.

3. Software Lifecycle Model

Swift Solution's company motto is to produce quality software at an affordable price for clients. In order to achieve this, it is important for the company to utilise a lifecycle model which will facilitate providing a quality piece of software that the client is happy with while also keeping development costs down.

The lifecycle model used by the company is an adapted version of the Spiral model for incremental delivery. The Spiral model is an iterative lifecycle with each iteration either adding new functionality or improving existing features relative to the previous iteration. Using this approach, the software can be seen growing and evolving toward the final deliverable product through successive increments.

The traditional Spiral model, as proposed by Boehm (1988), was broken into four main steps. These steps allow an incremental development of a software product. The steps are:

1. Determine Objectives, Alternatives, Constraints
2. Evaluate Alternatives; Identify, Resolve Risks
3. Develop, Verify Next-Level Product
4. Plan Next Phases

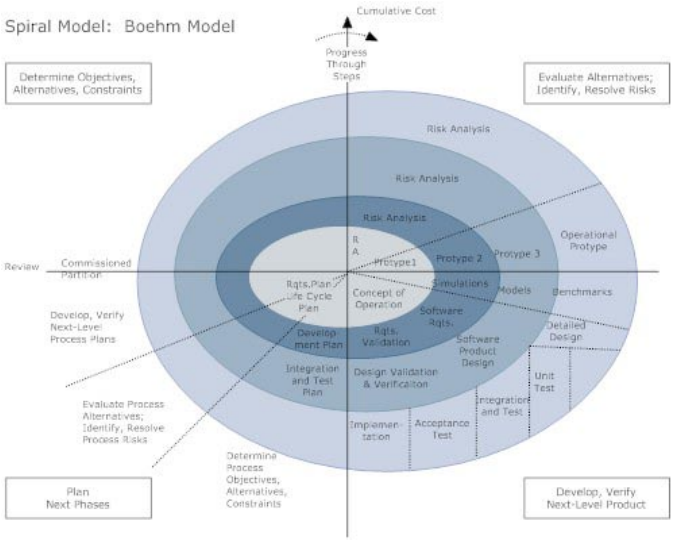


Figure 1. Traditional Spiral Lifecycle Model

(Image: http://www.smartdraw.com/examples/preview/index.aspx?example=Spiral_Model_-_Boehm_Model)

This model places more emphasis on risk analysis than other major models but also can be considered to include other lifecycle models in its process. A project repeatedly progresses through the four steps in iterations or spirals. The process starts with requirements and is followed by assessing the risk involved. The risk analysis phase aims to identify risks with the solution and propose alternatives before a prototype is produced in the next phase. Development is followed by testing in the same phase before an evaluation is carried out on the software output from the iteration and the project moves into the next spiral.

The traditional Spiral model was proposed for large scale projects and is not suitable for a small team like Swift Solutions. The overheads involved in implementing a model like this would not allow the team to produce cost effective solutions for clients. Swift Solutions has adapted the traditional model to a streamlined one which supports the companies goal of provide quality software at an economical price.

Swift Solutions has adapted the traditional Spiral model into an efficient process which is suited to the companies size and ambitions. The four steps in the process are:

- Analysis
- Design
- Implementation
- Testing

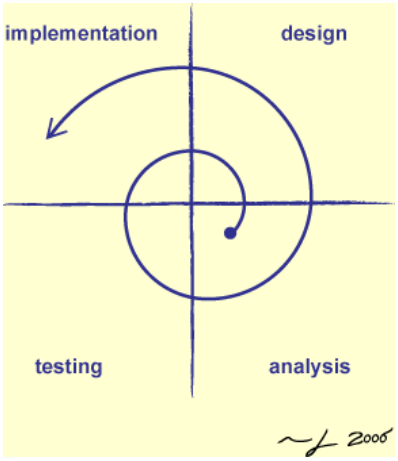


Figure 2 - Swift Solutions Adapted Spiral Lifecycle Model

(Image: http://www.osl.iu.edu/~pgottsch/swc2/lec/dev01.html)

The lifecycle starts with the initial analysis phase during requirements are gathered. This is followed by the first design phase where the Unified Modelling Language (UML) is used to model the product. Once the initial comprehensive design has been completed, this step is generally unlikely to consume a great deal of time during further iterations. This is followed by implementation of the first components which are then tested to complete the iteration. Clients are regularly involved in the testing phase to ensure the software is evolving to their satisfaction. Any changes can be accommodated in the next iteration with the aim being to ensure any changes to the system are made as early in the process as possible to reduce development costs. Each implementation phase endeavours to produce a component which can be demonstrated to the customer to ensure conformity.

The spiral model uses a risk management approach to software development. Some of the features which allow this include prototyping to reduce risk, accommodating requirement changes, and early detection of design problems. By using an iterative approach which includes regular communication with the client, the risk involved in the project is significantly reduced. This model incorporates a strong risk management strategy which reduces the risk through regular testing in each iteration.

4. Project Plan & Allocation of Roles

Item	Description	Delegation	Due Date
Narrative	Description of the project and overview of the software system	Clement	20 Feb 2009
Project Outline		Colin	20 Feb 2009
Life-cycle model	Discussion of the software model used during the project	Damien	20 Feb 2009
Requirements			
Functional	The required externally visible behavior of the system	Clement	24 Feb 2009
Use case diagrams	One detailed use-case and use-case description.	Colin	27 Feb 2009
Non-Functional	Discussion of quality attributes	Damien	4 Mar 2009
Analysis			
Candidate classes	List candidates using noun identification technique	Clement	4 Mar 2009
Class Diagrams	Class diagrams showing inheritance/aggregation/composition etc	Colin	11 Mar 2009
Sequence Diagrams	One sequence diagram depicting the process's for one part of the system	Damien	16 Mar 2009
Communication Diagrams	One communication diagram to show interaction within the system	Clement	21 Mar 2009
State Chart	Derived in relation to the communication or sequence diagram	Colin	23 Mar 2009
ER Diagrams	To show the relationships between classes	Damien	23 Mar 2009
Design			
Overview	Description of methodologies and any principles and styles adhered to	Clement	26 Mar 2009
MVC	Describe use of MVC in relation to this project	Colin	29 Mar 2009
Data Dictionary	Depict notations in the UML Diagrams	Damien	31 Mar 2009

5. System Architecture

This Library system's underlying functionality is to automate manual tasks currently present in the Portumna Community Library. The architecture of this project is sub-divided into five packages i.e. Stock Management, Member Management, Loan Management, Staff Management and Database Management. In the beginning it was considered to use a third-party Library catalogue system, but this was ruled out by the client to try and minimize costs. As a result a simple interface with the database for searching or browsing stock will need to be developed in conjunction with this system. This will be a unified interface through which the aforementioned packages will use to search and update the database. The database system will take care of data integrity which removes this responsibility from our system. This will ensure that the business logic of the system is not tightly coupled with a specific DMS.

There is a focus on modifiability with this project. This client has shown an interest in allowing new stock/members to be added in the future and, while not knowing what these could be, wants the addition process to be simple and transparent. A key concern in software engineering is Modifiability vs Performance. Many techniques of enhancing modifiability require some overhead that affects the performance. However in this case, the client does not foresee major changes to the working of the library in the future and requested a single deployment on a local machine. In addition to this, the Librarians are the only users of the system. So because this is not a distributed system, it allows us to make an architectural design decision to place a higher priority with modifiability. We will adopt styles according to "Design Patterns Gamma et. al" to help within the design process.

The deployment architecture is currently running Windows 2000 but will likely be upgraded in the near future. This could mean a change in architecture that consists of a new set of API's or deprecates previous ones. This would result in a revisit to the design phase and possibly extra documentation. To overcome this, and again because performance is not a major concern, Java and various portable libraries will be used. This will ensure that any operating systems supporting the java VM can be used as a deployment machine for this application. Whilst the client may not yet foresee changing OS, choosing this level of portability will broaden the scope of deployment possibilities for this system. Also with the use of Java, our internal UI design team are highly skilled in the Java Swing API so an easy-to-use and intuitive Graphical User Interface will help the transition from a manual system to a computer based system.

Architectural Package Diagram

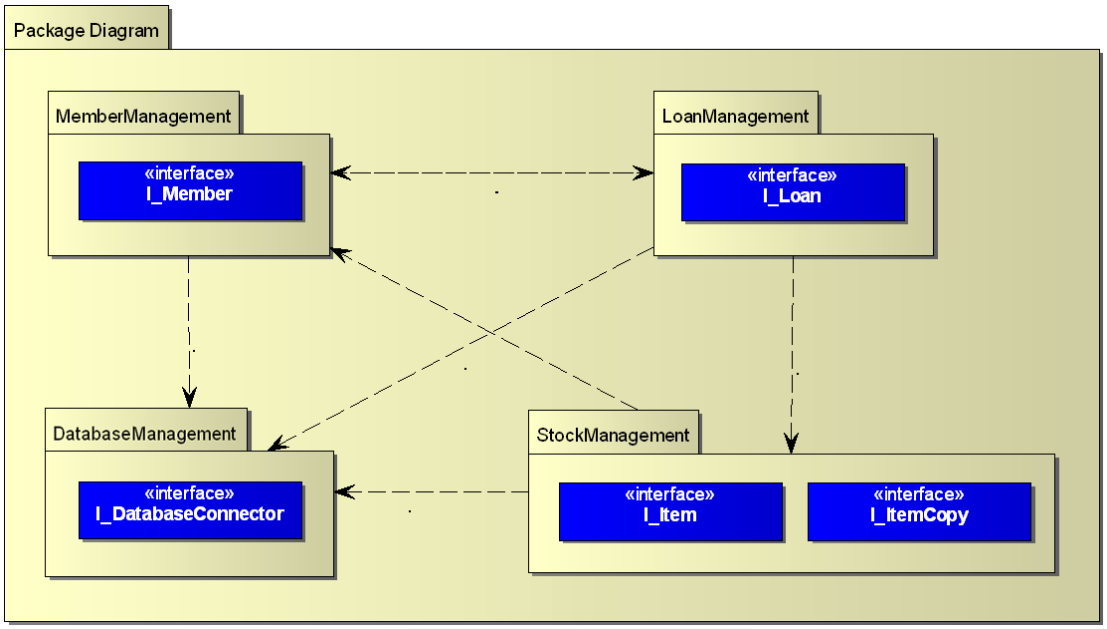


Figure 3. Architectural Package Diagram

Architectural Tiered Diagram

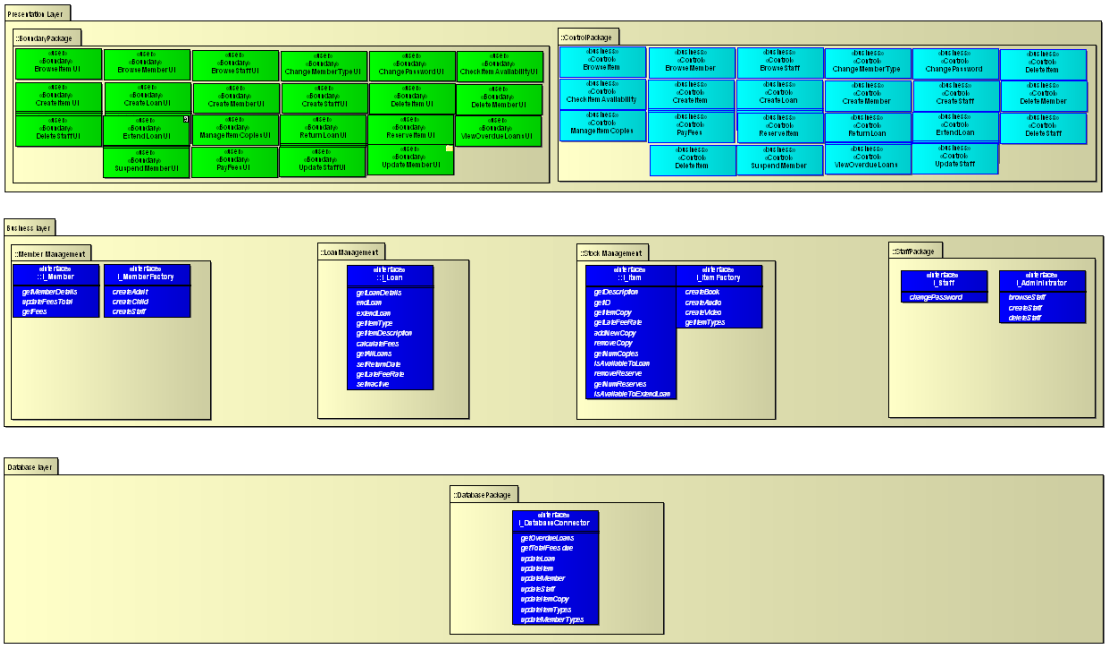


Figure 4. Architectural Tiered Diagram

6. Requirements

6.1 Use Case Diagrams

6.1.1 Stock Management Use Cases

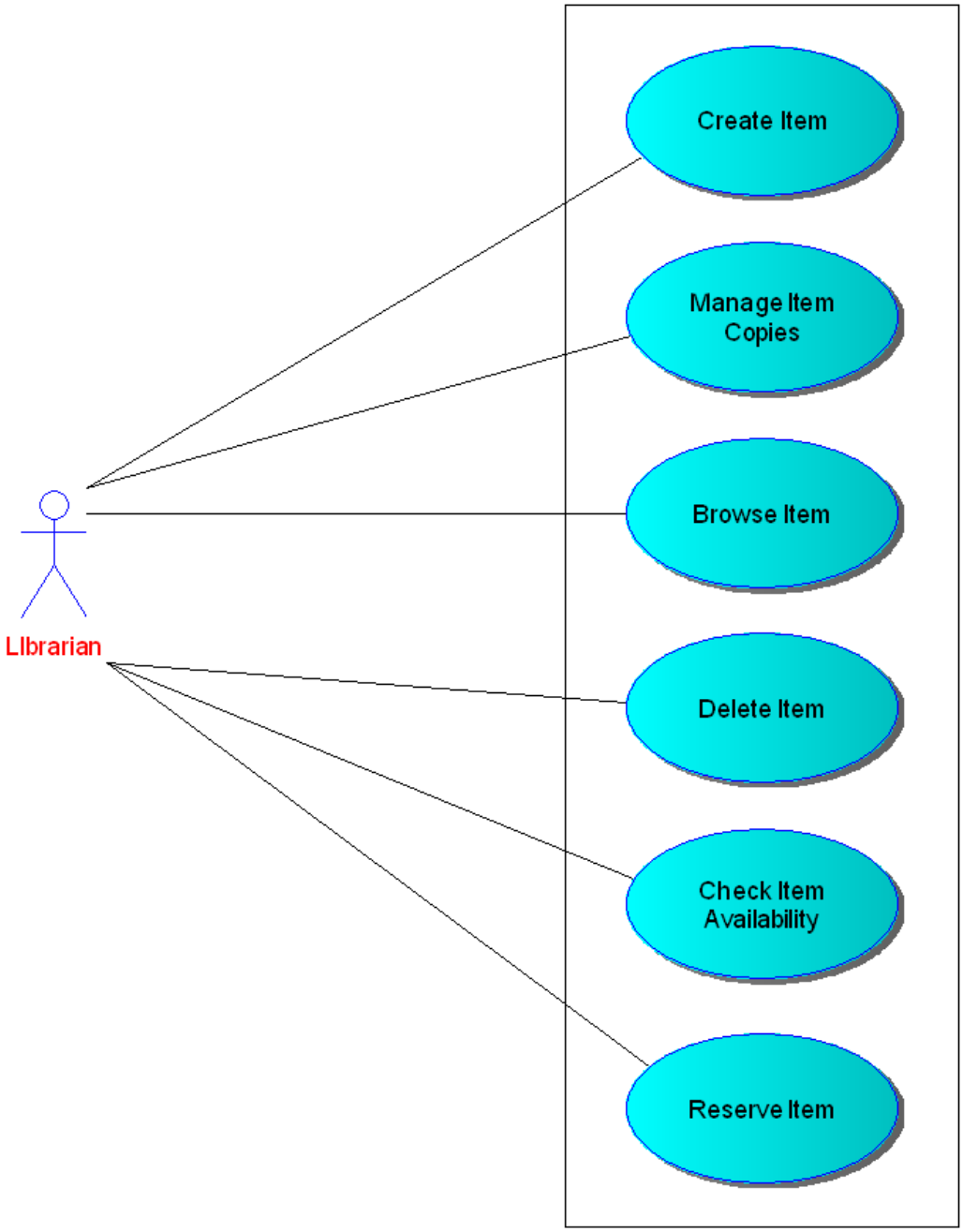


Figure 5. Stock Management Use Case Diagram

6.1.2 Member & Loan Management Use Cases

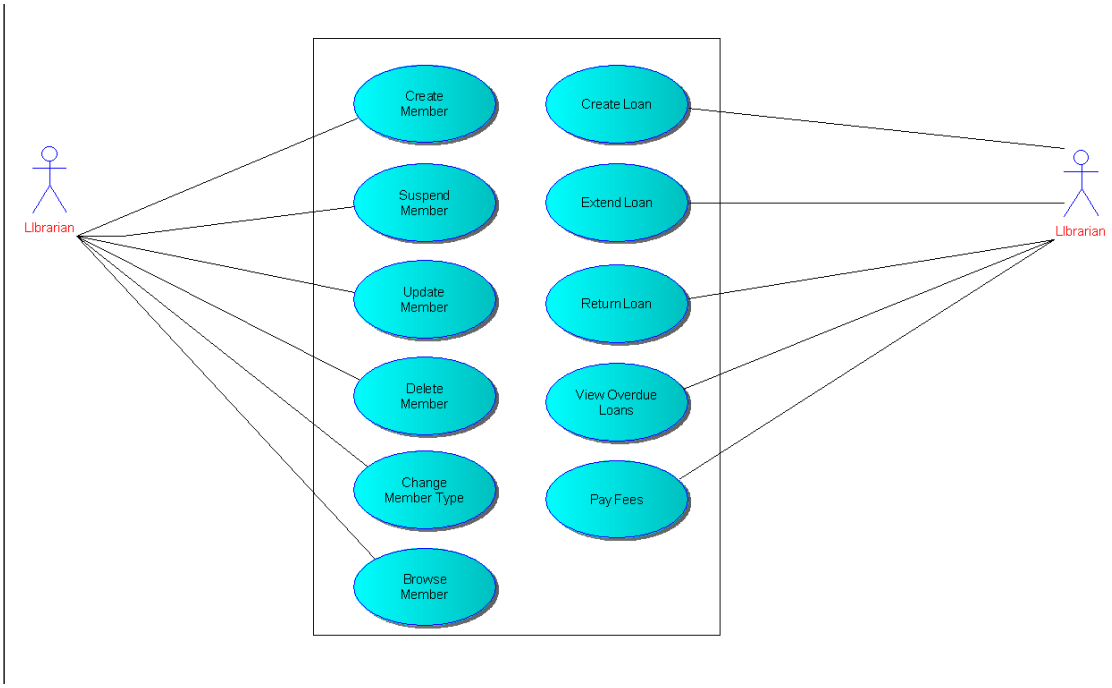


Figure 6. Member & Loan Management Use Case Diagram

6.2 Use Case List

No.	Requirement	Use Case(s)
1	To record new library members name(s) address and other relevant details	Create Member
2	To update member details	Update Member
3	To suspend a members account	Suspend Member
4	To delete a members account	Delete Member
5	To change a members type to either Staff, Adult or Child	Change Member Type
6	To browse for member details and related loan history	Browse Member
7	To record the details of each loan for each member. This will include the title of the item, the loan start date and loan due date.	Create Loan
8	To extend the period of a loan for a member	Extend Loan
9	To record a return of a loaned item and calculate any fees	Return Loan
10	To allow view of a members outstanding balance and record payments.	Pay Fees
11	Report on overdue loans	View Overdue Loans
12	To record new stock item details including title, genre, type etc	Create Item
13	To manage the stock level (number of item copies) for each item.	Manage Item Copies
14	To browse for stock items & allow view of relevant data.	Browse Item
15	To delete an item record from the system	Delete Item
16	To check for item availability.	Check Item Availability
17	To reserve a stock item for a member	Reserve Item
18	To browse staff details	Browse Staff
19	To create new staff details and grant access to the system.	Create Staff
20	To update staff details	Update Staff
21	To delete staff details	Delete Staff
22	To allow staff to change their system access passwords	Change Password
23	To backup data every day	Not applicable
24	To allow the system to be deployed on the existing library computer and be easily redeployed should the library change computer in the future.	Not applicable
25	To allow items to be imported from the existing MS Access system	Not applicable
26	To allow new items and member types to be added in the future with minimal modification to the existing system.	Not applicable

6.3 Use Case Descriptions

Use Case	Description
Create Member	When a person requests to become a member, the librarian will validate their required ID. Their details will be entered into the system, including Names(s), address, Contact details, Member type (Staff, Adult or Child). The system will generate and display a unique member code number which the librarian will manually print onto a library card along with the member's name.
Update Member	Sometimes a member will change address, surname etc, this facility will allow the librarian to find the relevant member record and update data.
Suspend Member	Occasionally a member's rights to the library may have to be temporarily suspended. This can happen for a number of reasons including refusal to pay overdue fees etc. The librarian will have a separate function to locate the member record and flag the member as suspended and update the suspension date. This system functionality will also be used to remove the suspension if required at a later date.
Delete Member	The librarian may wish to remove a member from the library system completely (perhaps the member has moved away from the area). First the members history is checked to ensure that there are no outstanding loans or monies owed to the library prior to allowing the removal.
Change Member Type	From time to time the librarian will amend members types. The types are Staff, Adult and Child. This is used in the main to move members from Child to Adult type as they come of age and also to change either an adult or child to a staff member if they become staff at the library.
Browse Member	The librarian can browse for a member via search screens with the intention of pinpointing a specific member and relevant information relating to them. Multiple search criteria can be used depending on task being carried out by the librarian.
Create Loan	The member selects the item they wish to loan from the library and presents to the librarian at the counter. The librarian will first check whether the member has any outstanding fees due, if so they may refuse the borrowing request from the member. The system will also qualify that the member has not exceeded their loan quantity and will only allow loans up to that quantity. Finally a check is made to ascertain whether the item copy has been reserved already by a different library member. Note that this entails checking the number of current reservations for this item against the number of item copies available within the library. All being well the librarian will enter the loan item details against the member on the system which will update the starting date for the loan.
Extend Loan	A member contacts the librarian prior to the loan due date being exceeded on a library item they have on loan and requests an

Use Case	Description
	extension to the loan. If the librarian agrees then he/she updates the loan details for that member which will extend the due date by a default amount of days and the new due date is relayed to the member.
Return Loan	When a member returns a loaned item, the librarian updates the members relevant loan details and any outstanding fee are calculated and shown within the members outstanding fee balance. Generally the librarian will request immediate payment and if not received will inform the member that no new loans will be granted until the outstanding fee balance is cleared.
Pay Fees	When a member pays fees to the librarian, this functionality shows the outstanding balance, allows entry of payment amount and displays the updated balance.
View Overdue Loans	This facility will allow the librarian to list the overdue loans and sort them by certain useful criteria such as listing members with the most overdue loans first etc. The librarian will use this data to contact members with large fee balances or may provoke the decision to suspend members.
Create Library Item	When a new library item is entered into the system, each has their details which are type specific (i.e. Book, Audio or Visual media) entered and a unique item ID is generated. NOTE that this item ID is printed onto a paper form that is attached to the inside of each item (either cover or box etc).
Manage Item Copies	Librarian maintains the stock level for individual stock items. The library will purchase and sell copies of the stock items and their stock balances will be maintained here.
Browse Item	The librarian can browse for a stock item via search screens with the intention of pinpointing a specific item and relevant information relating to them. Multiple search criteria can be used.
Delete Item	The librarian can delete an item completely from the library when there is no stock of the item carried and there is no intention of getting this item into the library anew.
Check Item Availability	Librarian may need to check whether a library item exists and whether there are any available to loan to a member. This facility can if required, display current loans of copies of the item in question to allow an estimate of availability to be given.
Reserve Item	The librarian can check an items availability for a member and if requested can place a reserve request against this stock item for the member. When the item becomes available the librarian can inform the member when next in contact.
Browse Staff	The browse staff details are available only to the system administer, and allows them to browse staff details and select existing data for either update or delete. Also within this facility the administrator will also have the facility to create a new staff member details.
Create Staff	When a new member is created, the administrator will have to enter a new unique user id, name, address and contact details for the new staff member. Also the new staff members access

Use Case	Description
	password can be set or defaulted to the same as their user id.
Update staff	The administrator updates any relevant details and/or resets the staff members access password.
Delete Staff	The administrator deletes the staff members details
Change Password	This facility allows all staff members to change their password should the need arise providing they know their current password.

6.4 Detailed Use Case Descriptions

6.4.1 Member Management

Use Case description: Create Member

Actor Action	System Response
1. Selects "Create New Member"	2. Lists the types of member that can be created (Staff, Adult or Child) Allow Browse members function here also to qualify that potential member does not already exist.
3. Selects a member type	4. Displays new member entry fields incl. Name, Address, contacts details etc. Also displays type specific defaults such as loan periods allowed by item type etc
5. Enters new members details & confirms create request.	6. Displays updated member details and new member unique number (ID)
Includes: Browse Members Use Case	

Use Case description: Update Member

Actor Action	System Response
1. Selects "Update Member "	2. Display entry field for member ID or Browse Members function (Use Case)
3. Selects a member	4. Displays member details
5. Changes relevant member data and confirms update request.	6. Displays updated member details.
Includes: Browse Members Use Case	

Use Case description: Suspend Member

Actor Action	System Response
1. Selects "Suspend Member"	2. Display entry field for member ID or Browse Members function (Use Case)
3. Enters/Selects a member	4. Displays member details including loans details. Displays a suspension reason entry note.
4. Enters reason description for suspension	5. Display member details and suspension reason description and request to confirm suspension.
6. Confirms suspension	7. Display updated member details showing the suspension date/time and ID of staff member (librarian) who applied the suspension. Also displays suspension reason note if completed.
Includes: Browse Member Use Case	

Use Case description: Delete Member

Actor Action	**System Response**
1. Selects “Delete Member”	2. Display entry field for member ID or Browse Members function (Use Case)
3. Enters/Selects a member	4. Displays Member details including loan history and Fees balance. Display delete confirmation message,
5. Confirms Deletion	6a Deletion not allowed as fees or loans outstanding. Displays message to allow ability to pay fees and return loans and proceed back to point 4.
	6b Displays confirmation that member was deleted.
Includes: Browse Members Use Case, Pay Fees Use Case, Return Loan Use Case	

Use Case description: Change Member Type

Actor Action	**System Response**
1. Selects “Change Member Type”	2. Display entry field for member ID or Browse Members function (Use Case)
3. Enters/Selects a member	4. Displays Member details including loan history and Fees balance. Lists the members types.
5. Confirms Type change	6 Update the members details with the new member type flag.
Includes: Browse Members Use Case, Pay Fees Use Case Alternative Courses: The user may already have accessed the member details via use case “Update Member”, if option taken from here then only steps 4, 5 & 6 apply.	

Use Case description: Browse Members

Actor Action	**System Response**
1. Selects “Browse Members”	2. Displays list of all members sorted by name
3. Enters search text	4. Displays only names matching the text
Alternative Courses: This use case can be activated through other use cases such as "Create Loan", "Extend Loan", "Pay Fees".	

Use Case description: Create Loan

Actor Action	System Response
1. Selects “Create Loan”	2. Browse Members (Use Case)
3. Selects a member	4. Displays member details
5. Enters the item’s I.D. number	6. Displays updated member details including current loans
7. Enters the item Copy ID number	8. Display new loan item on screen with calculated return date
9. Confirm new loan item	10. Update new loan to database and display loan screen showing all current loans.
Includes: Browse Members Use Case Alternative Courses: The actor may already have accessed the member's details through another use case such as "Browse Members", "Extend Loan", "Pay Fees". In this case, only steps 5 and 6 apply. Extension: Step 4 will display any outstanding fees, librarian may decide not to proceed with loan Extension: Step 8 may display “unable to loan” message due to member breaching loan capacity or all item copies already reserved against other library members, librarian may decide not to proceed with loan.	

Use Case description: Extend Loan

Actor Action	System Response
1. Selects “Extend Member Loan”	2. Browse Members (Use Case)
3. Selects a member	4. Displays member details including current loans
5. Selects the relevant loan to extend	6. Displays updated member details including current loans and due dates
Includes: Browse Members Use Case Alternative Courses: The actor may already have accessed the member's details through another use case such as "Browse Members", "Create Loan", "Pay Fees". In this case, only steps 5 and 6 apply. Extension: Step 4 display any outstanding fees, librarian may decide not to proceed with extension. Extension: Step 4 may display “unable to loan” message due to all item copies including those loaned being fully reserved by other members, librarian may decide not to proceed with extension.	

Use Case description: Return Loan

Actor Action	System Response
1. Selects “Return Loan”	2. Displays item I.D. box.
3. Enters loan item I.D.	4. Displays confirmation of loan return

Use Case description: View Overdue Loans

Actor Action	System Response
1. Selects “View Overdue Loans”	2. Displays list of overdue loans
Extension: After step 2, the actor prints an overdue loans report.	

Use Case description: Pay Fees

Actor Action	**System Response**
1. Selects “Pay Fees”	2. Browse Members (Use Case)
3. Selects a member	4. Displays member details including current balance
5. Enters amount to pay in pay.	6. Displays updated member details including account balance
Includes: Browse Members Use Case Alternative Courses: The actor may already have accessed the member's details through another use case such as "Browse Members", "Create Loan", "Extend Loan". In this case, only steps 5 and 6 apply.	

6.4.3 Stock Level Management

Create New Library Item

***Pre**: The actor wants to create a new Item for the Library*

	Actor Action		System Response
1	Selects create new Item button	2	Lists the types of library items that can be created
3	Selects the type of library item	4	Displays fields specific to library item, i.e author/director, year published/released etc
5	Fills in required fields and submits	6	Records new library item to database

***Post**: The system successfully records a new Item and specified information*

Manage stock level library item

***Pre**: The actor wants change the details of an existing Library Item*

	Actor Action		System Response
1	Selects Manage Stock Button	2	Browse stock items
3	Chooses just one result	4	Displays detailed information regarding the library item and rights to change fields and/or add and remove copies
5	Updates information	6	Updates the database record for this item
Includes: Browse stock items Extension: After step 2, there are no items matching the description and such a message is produced. The system exits the use case.			

***Post**: All details of the item are successfully updated to the database and consistent with the actor inputs*

Browse stock items

***Pre**: The actor has access to search the database*

	Actor Action		System Response
1	None, selects the browse button	2	Shows a search field where the actor can enter various keys to search for
3	Fills in various fields	4	Searches database for elements matching the input from the actor and lists results
Extensions: After step 4, the user selects a certain item to be displayed			

***Post**: The data returned adheres to the input parameters by the search*

Delete Library Item

***Pre**: The actor wants to delete an existing Item from the library*

	Actor Action		System Response
1	Delete Library Item button	2	Browse stock items
3	Chooses just one result	4	Displays detailed information regarding the library item
5	Selects whether to delete the item or not	6.a	If a unit of this item is on loan display message and do not delete item
		6.b	Deletes the item from the database
Includes: Browse stock items			

***Post**: 6.b The Item is completely deleted from database*

***Post**: 6.a The Item is not deleted from database*

Check Item Availability

Pre:

	Actor Action		System Response
1	Check availability button	2	Browse stock items
3	Chooses just one result	4	Checks and notifies if any copies of this item are in stock
Extensions: After step 4, shows an option whether to reserve or not. If reserve, system response leads to step 3 in Reserve Library Item			

Post:

Reserve Library Item

Pre:

	Actor Action		System Response
1	Reserve Item Button, None	2	Check Item Availability
3	If item available actor can enter an ID number for the user who wants to reserve the item	4	Browse user based with ID number as key
5	Selects the user with ID number returned	6	Puts a reserve status on the Library Item, associating it with the member ID number
Includes: Check Item Availability Includes: Browse user			

Post*: The Library Item has a new reservation associated with the ID number. The count of reservations of this Item is incremented*

6.4.4 Staff Management

Use Case description: Browse Staff
Pre: Only the staff with administrator rights will have access to Browse staff option

Actor Action	**System Response**
1. Selects "Browse Staff" option	2. Lists the staff details in alphabetic sequence allowing for the entry of search criteria on staff member name
3. Enters search text to initiate search by name 3a. Selects "Add new staff" function.	4. Displays only staff details which match entered search criteria 4a. Displays "Create new staff member" screen and allow relevant entries, when successful return to point 2 showing any new staff detail.
5. Selects staff member.	6. Displays staff details with options to: Update/Reset staff authorisation password Update staff details Delete staff details
7a. Selects option to reset password 7b. Select option to update staff details 7b. Select option to delete staff details	8a. Display current staff ID and password field defaulted to the ID, allow administrator to override password. 8b. Display staff details and allow update 8c. Display current details and allow confirm of delete decision.
9. Confirm maintenance choices	10. Updates relevant details and returns to point 2 above to display the list of current staff details
Includes: Create Staff, Update Staff, Delete Staff	

Post: Administrator browse of staff details and associated functionality successful

Use Case description: Create Staff
Pre: The decision to create a new staff member will have already been taken via the Browse Staff use case

Actor Action	**System Response**
1. None	2. Displays a screen showing the new staff ID number ("Lib" concatenated with a number) and entry capable fields for the users name, address, contact details and start date. Also displays the staff password field with the staff ID defaulted as its value.
3. Enters staff data and possibly new password.	4. Displays same screen details along with a message requesting a confirmation of update.
5. Confirms update	6. New staff member record and authorisation to system created.
Includes:	

Post: Creation of new staff member successful

Use Case description: Update Staff
Pre: The decision to update a new staff member will have already been taken via the Browse Staff use case

Actor Action	**System Response**
1. None	2. Displays a screen showing staff details, some of which are entry capable. Also displays facility to reset staff member authorisation password.
3a. Enters changed staff data 3b. Select function to reset password	4a Displays update confirmation screen 4b Displays window to reset password & allows confirmation.
5. Confirms update	6. Staff member details and/or password updated
Includes:	

Post: Update of staff member successful

Use Case description: Delete Staff
Pre: The decision to delete a new staff member will have already been taken via the Browse Staff use case

Actor Action	**System Response**
1. None	2. Displays a screen showing staff details and a delete confirmation message.
3. Confirms delete	4. Staff member details deleted
Includes:	

Post: Delete of staff member successful

Use Case description: Change Password
Pre: The actor (staff) knows their current password

Actor Action	**System Response**
1. Select "Change Password" option	2. Recognises the user, displays their user ID number and name on the screen. Displays an entry capable password field.
3. Enters new password.	4. Displays update confirmation screen
5. Confirms update	6. Staff password changed.
Includes:	

Post: Staff password changed.

6.5 Detailed Use Cases

6.5.1 Create Loan Use Case

USE CASE 7	Create Loan	
Goal in Context	Member brings library item(s) to librarian at counter, librarian qualifies that member can loan item and creates a loan record for each item the member presents, within the loan allowance criteria specific to the member.	
Scope & Level	Primary Task	
Preconditions	The person requesting to loan the item is a library member.	
Success End Condition	Library member loans the item	
Failed End Condition	Library member is not allowed to loan the item.	
Primary, Secondary Actors	Librarian (acting on behalf of the library member)	
Trigger	Library member presents an item(s) for loan to the librarian	
DESCRIPTION	**Step**	**Action**
	1	Library member present item(s) for loan to librarian
	2	Librarian selects Create Loan option
	3	Member supplies ID or librarian searches for member ID via Browse member functionality
	4	Member ID entered
	5	Members details displayed including loan history
	6	Librarian enters item ID number which is on the item.
	7	System displays updated member loan details
	8	Library member leaves with loaned item(s).
EXTENSIONS	Step	Branching Action
	3a	Member ID can be input if available or member ID searched for using names & contact details search criteria (Use Case 6 – Browse Members)
	5a	Library members has outstanding fees – agrees to pay immediately (NOTE payments taken in cash only). 5a1. Take the fees or part thereof from member (Use Case 10 – Pay Fees)
	6a	Item ID displayed on item is physically damaged, search for item ID via item type specific criteria. 6a1. Browse for item ID (Use Case 14 – Browse Item)
	7a	System informs librarian/member that member cannot loan the item as they will exceed their allowance for loans. (i.e. the quantity of a given item type a member can have on loan at any given time)

	7b	System informs librarian/member that member cannot loan the item as all copies in stock have been reserved by other library members. NOTE that copies are not reserved against specific item copy, there is a list of member reservations held against an item and the number of these exceeds the number of item copies currently available.
VARIATIONS	Step	Branching Action

RELATED INFORMATION	7. Create Loan
Priority	Top
Performance	45 seconds to complete a loan (assuming no physical issues)
Frequency	50/Day
Channels to actors	Interactive
OPEN ISSUES	a) Will system allow librarian to loan an item copy when a member has used up all their allowance for the particular item type? b) Will system allow librarian to loan an item copy when member has outstanding fees due? c) Will system allow librarian to loan an item copy when the number of reservations against the item in question exceeds the amount of item copies currently available in the library? (I.e. does the librarian want to be able to override this condition)?
Due Date	Release 1.0
…and other management information…	
Superordinates	
Subordinates	Browse Members (Use Case 6), Pay Fees (Use Case10), Browse Item (Use Case 14)

6.5.2 Extend Loan Use Case

USE CASE 8	Extend Loan	
Goal in Context	Member contacts librarian and requests extension to loan of a library item, member provides member ID and item ID (or relevant item data to ensure librarian can qualify the item in question). Librarian locates the member's relevant loan record and extends the loan by a period of time.	
Scope & Level	Primary Task	
Preconditions	The person requesting the loan extension is a library member and the librarian has enough item data (i.e. item ID or item description) to qualify the loan item.	
Success End Condition	Librarian extends the loan by a time period.	
Failed End Condition	Library unable to extend the item loan period.	
Primary, Secondary Actors	Librarian (acting on behalf of the library member)	
Trigger	Library member requests an extension to the loan period for a specific item(s).	
DESCRIPTION	**Step**	**Action**
	1	Library member requests an item loan extension and supplied related member and item information.
	2	Librarian selects Extend Member Loan.
	3	Librarian uses Browse members function to either enter the member ID or search for member with data provided.
	4	Member ID entered
	5	Member details displayed and a list of current item(s) on loan to member displayed. Each row includes item type, item description and current loan end date.
	6	Librarian selects loan item and is shown a default extension date (i.e. new loan end date) which can be overridden.
	7	Librarian confirms extension and the system returns to the loan items list (Step 5) which displays new loan end date.
	8	Librarian exits function & confirms extension with member
EXTENSIONS	Step	Branching Action
	3a	Member ID can be input or ID searched for using name etc search criteria (Use Case 6 – Browse Members)
	5a	Member details will show any outstanding fees – if member present, librarian can request and take fees at this point (Use Case 10 – Pay Fees)
	7a	System informs librarian/member that member cannot extend loan of the item copy as all copies

		in stock and on loan have been reserved by other library members. NOTE that copies are not reserved against specific item copy, there is a list of member reservations held against an item and the number of these exceeds the number of item copies currently available.
VARIATIONS	Step	Branching Action
	1	Member may request extension and supply information: Verbally either at the library or over the phone An Email to the library (Note not linked to library system)

RELATED INFORMATION	8. Extend Loan
Priority	Top
Performance	40 seconds to complete a loan extension
Frequency	20/Day
Channels to actors	Interactive
OPEN ISSUES	a) Will there be a maximum time period that is allowed for an item loan (defined by member type and/or item type) to any given member? b) Will system allow librarian to extend a loan when the number of reservations against the item in question exceeds the amount of item copies currently available in the library? (I.e. does the librarian want to be able to override this condition)?
Due Date	Release 1.0
…and other management information…	
Superordinates	None
Subordinates	Browse Members (Use Case 6), Pay Fees (Use Case10)

6.6 Non Functional Requirements Discussion

Quality attributes are required of any non-trivial software product. Modifiability was one that was required by the client in order to allow for future potential changes to the system. Usually this comes at a cost of performance so a balance must be struck. For example with collection classes there is a performance hit with more function calls however the management or *housekeeping* of the object instances is removed from domain specific classes. The database system that will be used will connect through a unified interface to the system allowing different databases to be used easily as the changes are localised to the implementing class.

This key non-functional requirement, modifiability, is specified using the *Volere* template for non-functional requirements.

6.6.1 Modifiability

Content

The modifiability of this software is the metric of how easy it is to expand or change the function of the system. There are many parts of the system that may require a significant change. For example new members or new item types may be introduced at a later date to facilitate for changing business requirements. The current deployment is on a Windows platform but this may later change to a Unix based system. A Unix system would likely have a very different API than the Windows platform.

Motivation

The reason the client wants to keep this system modifiable is to try and *future-proof* the software. It may be the case that the library will be fitted with a new computer, possibly introducing a new architecture or operating system. Unless the code was portable to be deployed on any platform then a significant amount would need to be ported. This is a cost that could potentially recur and become a financial burden to update code each time.

Examples

The software will use a cross-platform API in order to abstract the discrepancies amongst different operating system API's.

Clear changeable features of the library system allowed to change at run-time, effectively removing the need to recompile the system to allow new types.

Fit Criterion

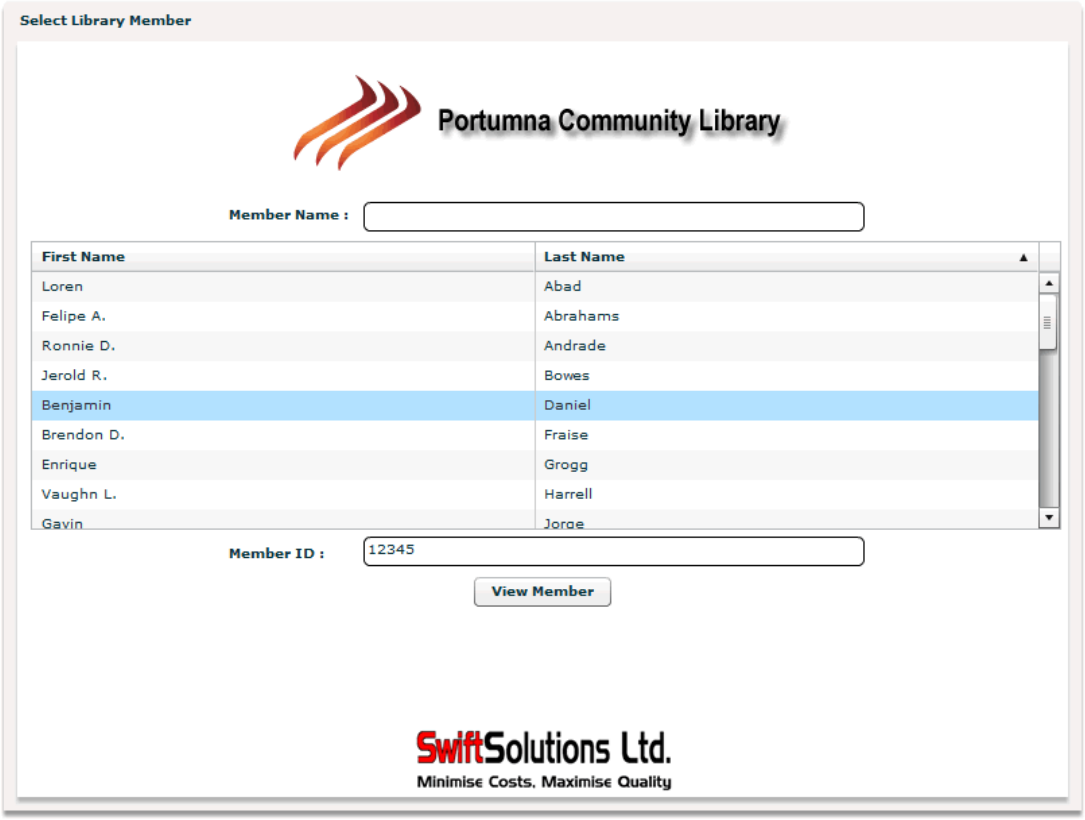
The system will be developed using Java Enterprise Edition. This will allow the system to be deployed on any architecture supporting the Java Virtual Machine. The use of design patterns featured in "*Design Patterns Gamma Elements of Reusable Object-Oriented Software" Gamma et al* will allow the extraction of different states from classes, and using composition or aggregation as a means to complete the functional requirements. This specific pattern is known as the state pattern and will be discussed further in the design phase.

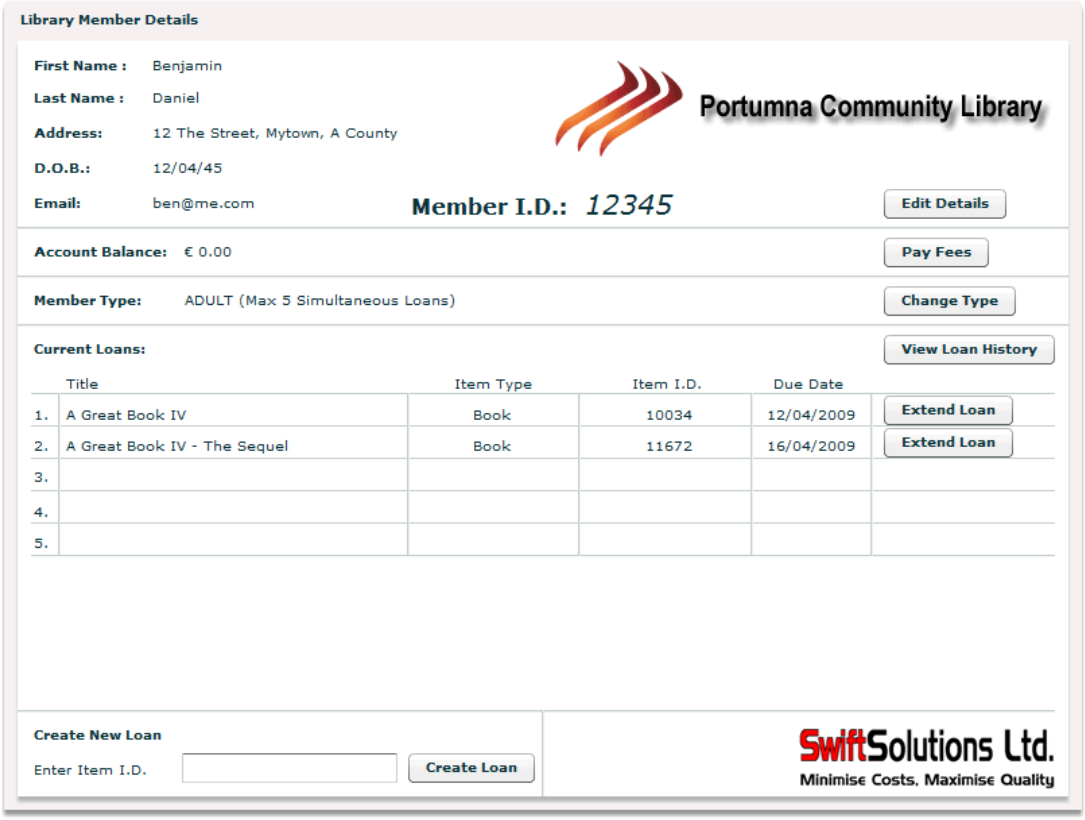
Considerations

Even though this system is taking pro-active steps to try and resolve the issue of modifiability there remains situations that will be difficult to eradicate. Most of the sequential operations are maintained in the control classes. These do make the business classes more reusable however if the sequence was to change these classes would need to be replaced.

6.7 Prototypes

6.7.1 Sample G.U.I.





6.7.2 Sample Reports

On-screen Report

Overdue Loans Report

Portumna Community Library

Member Name	Type	I.D.	Item Title	Type	I.D.	Due Date	Length Overdue ▲
Wylie, Vito G.	ADULT	1001	The Emperor's Academy	BOOK	44672	27-Feb-2009	01
Vancamp, Laurence F.	ADULT	2558	The Secret of the Dreamer	BOOK	78921	20-Feb-2009	08
Truesdale, Myles	ADULT	9522	The Shard's Rose	BOOK	46584	18-Feb-2009	10
Oropeza, Angelo	ADULT	7792	The River of the Edge	BOOK	23548	17-Feb-2009	11
Bowes, Jerold R.	ADULT	4578	The Wanton Stones	BOOK	46582	16-Feb-2009	12
Searles, Jarred K.	ADULT	4682	Word in the Alien	BOOK	48682	16-Feb-2009	12
Tolleson, Eloy B.	ADULT	1012	Misty Tales	AUDIO	95463	13-Feb-2009	15
Harrell, Vaughn L.	CHILD	1862	Seventh Ship	VIDEO	48652	10-Feb-2009	18
Witcher, Nickolas	CHILD	4892	The Time of the Planet	BOOK	35685	08-Feb-2009	20
Merlin, Billy J.	ADULT	8752	Ice of Secrets	VIDEO	78952	30-Jan-2009	29

Print Report

Printed Report

Portumna Community Library

Overdue Loans Report
01 - March - 2009

Name	Type	I.D.	Title	Type	I.D.	Due Date	Days
Wylie, Vito G.	A	1001	The Emperor's Academy	BK	44672	27/02/09	01
Vancamp, Laurence F.	A	2558	The Secret of the Dreamer	BK	78921	20/02/09	08
Truesdale, Myles	A	9552	The Shard's Rose	BK	46584	18/02/09	10
Oropeza, Angelo	A	7792	The River of the Edge	BK	23548	17/02/09	11
Bowes, Jerold R.	A	4578	The Wanton Stones	BK	46582	16/02/09	12
Searles, Jarred K.	A	4682	Word in the Alien	BK	48682	16/02/09	12
Tolleson, Eloy B.	A	1012	Misty Tales	AO	95463	13/02/09	15
Harrell, Vaughn L.	C	1862	Seventh Ship	VO	48652	10/02/09	18
Witcher, Nickolas	C	4892	The Time of the Planet	BK	35685	08/02/09	20
Merlin, Billy J.	A	8752	Ice Secrets	VO	78952	30/01/09	29

7. Analysis Diagrams

7.1 Identifying Candidate Classes

Identifying the right classes is one of the main skills in OO development (Stevens & Pooley 2000).

We have chosen to use the approach of Noun Identification Technique on the use case descriptions to obtain the key domain abstractions within the library system.

The approach is to take the contents of each use case description and underline its noun and noun phrases which will provide a list of candidate classes.

Example: Create Loan use case

Use Case description: Create Loan

Actor Action	**System Response**
1. Selects "Create **Loan**"	2. Browse **Members** (Use Case)
3. Selects a **member**	4. Displays **member details**
5. Enters the **item** **I.D. number**	6. Displays updated **member** details including **current loans**
7. Enters the **item Copy** **ID number**	8. Display new **loan item** on **screen** with calculated **return date**
9. Confirm **new loan item**	10. Update **new loan** to **database** and display **loan** **screen** showing all **current loans.**

Now apply a set of heuristics to each item on the list which will eliminate poor candidates from the list. The heuristics applied to each item on the list were:

Is it beyond the scope of the system?
Does it refer to the system as a whole?
Does it duplicate another class?
Is it too vague?
Is it too specific?
Is it really an attribute?
Is it really an operation?

Applying the above for the "Create Loan" use case the possible class candidates were:

Loan
Member
Item
Item Copy

7.2 System Class Diagram

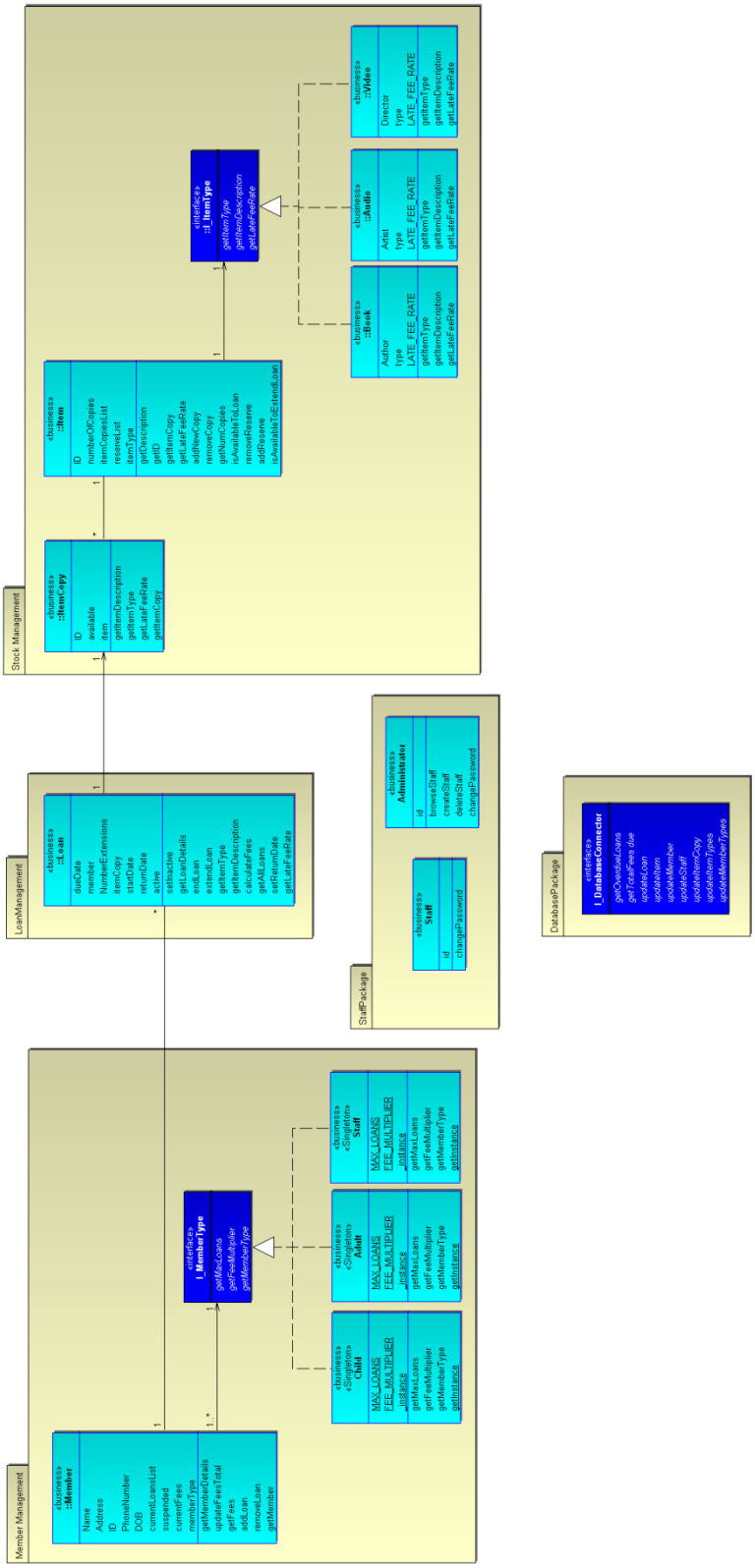


Figure 7. Analysis Phase - Class Diagram

7.3 Create Loan Communication Diagram

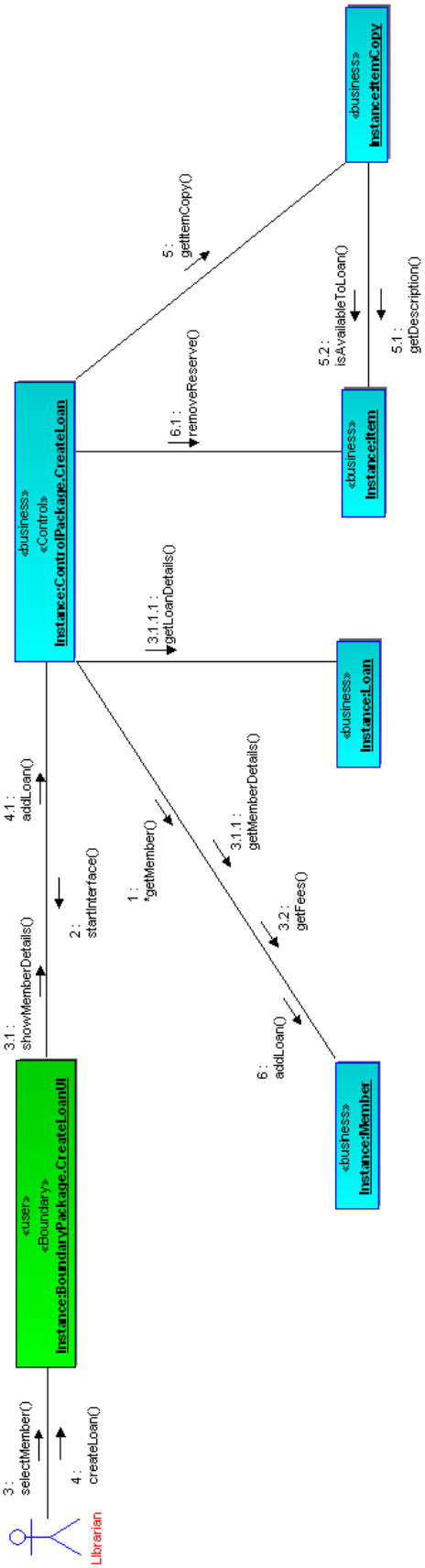


Figure 8. Create Loan Communication Diagram

7.4 Extend Loan Communication Diagram

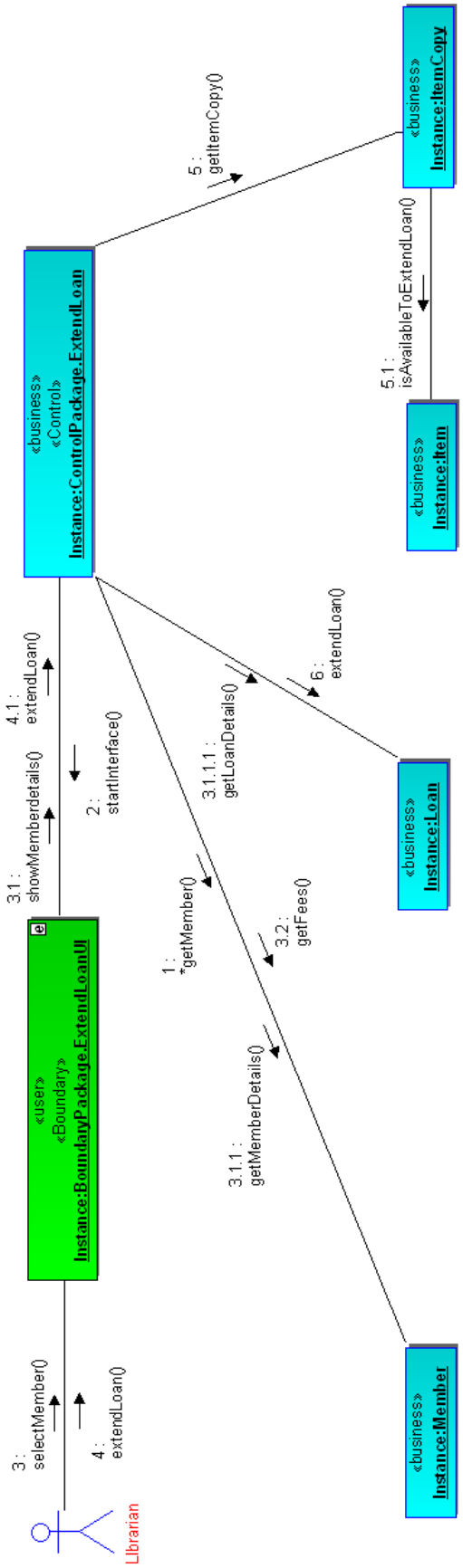


Figure 9. Extend Loan Communication Diagram

7.5 Loan State Chart

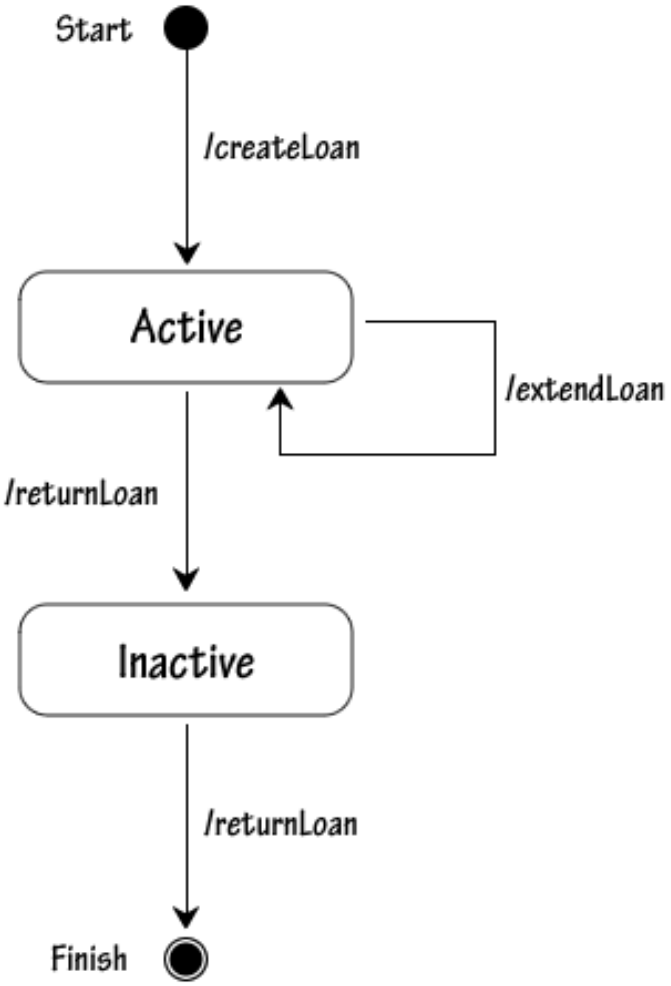


Figure 10. Loan State Chart

7.6 Entity Relationship Diagram

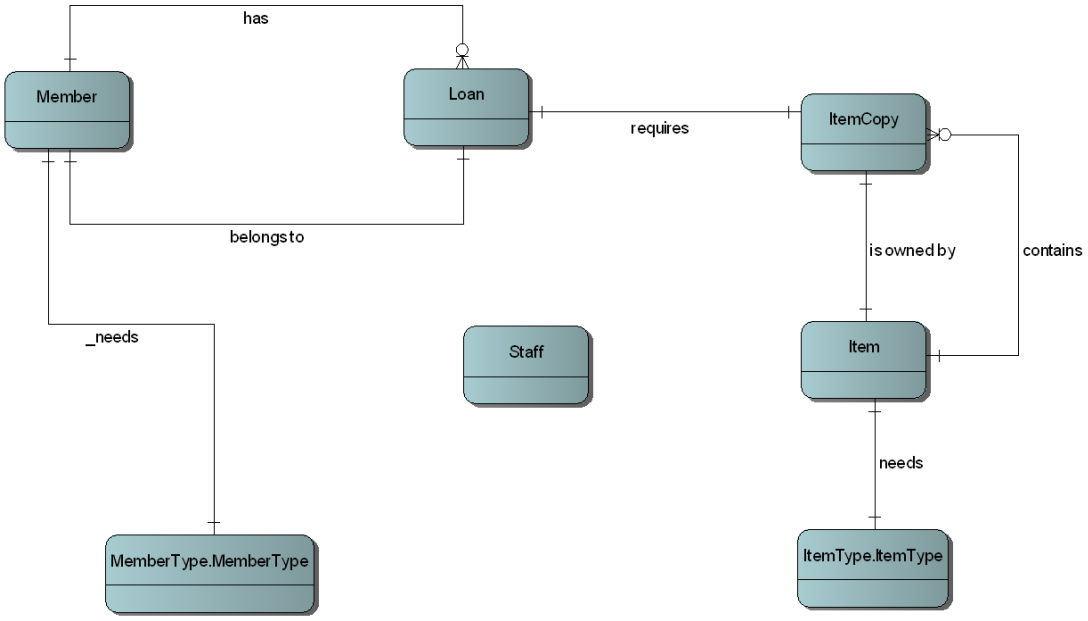


Figure 11. Entity Relationship Diagram

8. Design Diagrams

8.1 System Class Diagram

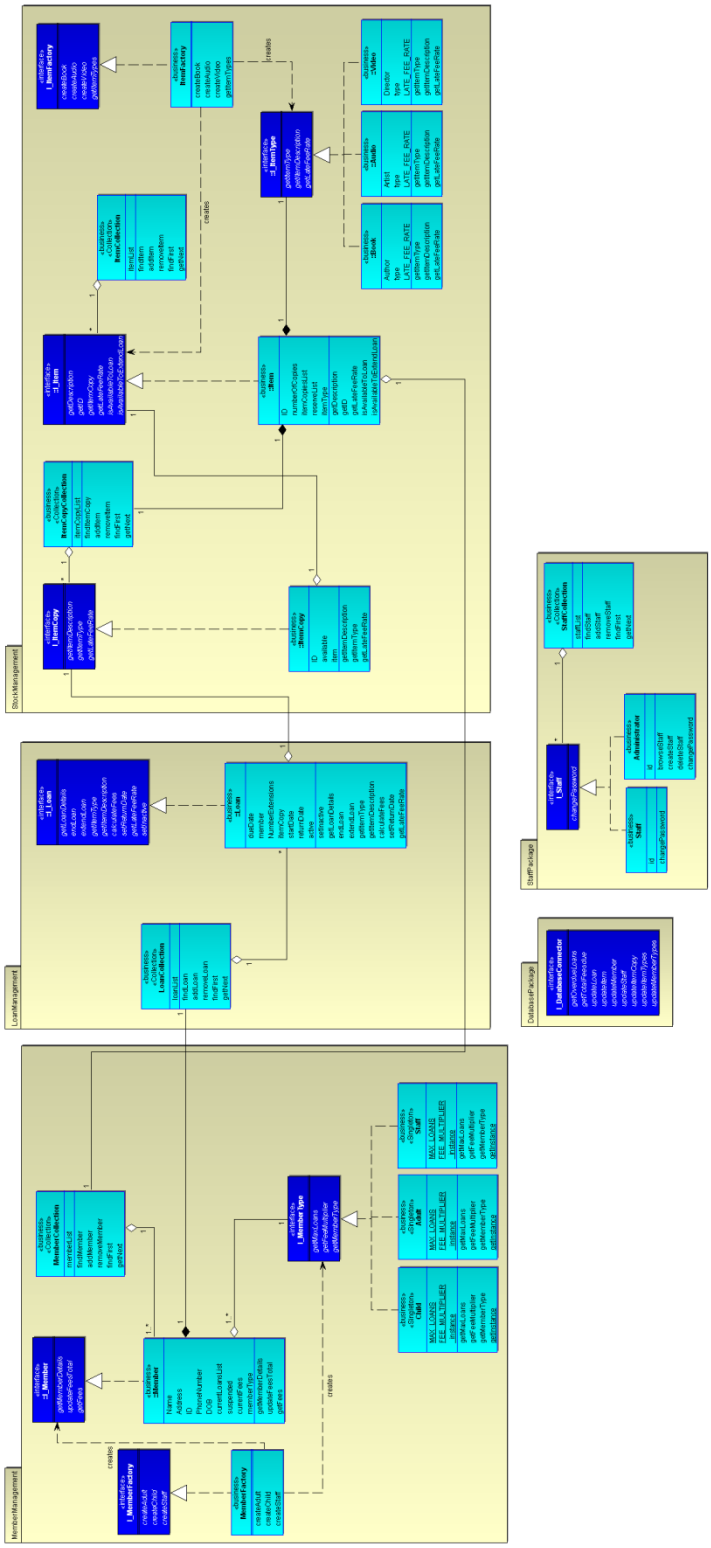


Figure 12. Design Phase - Class Diagram

8.2 Member Management Package

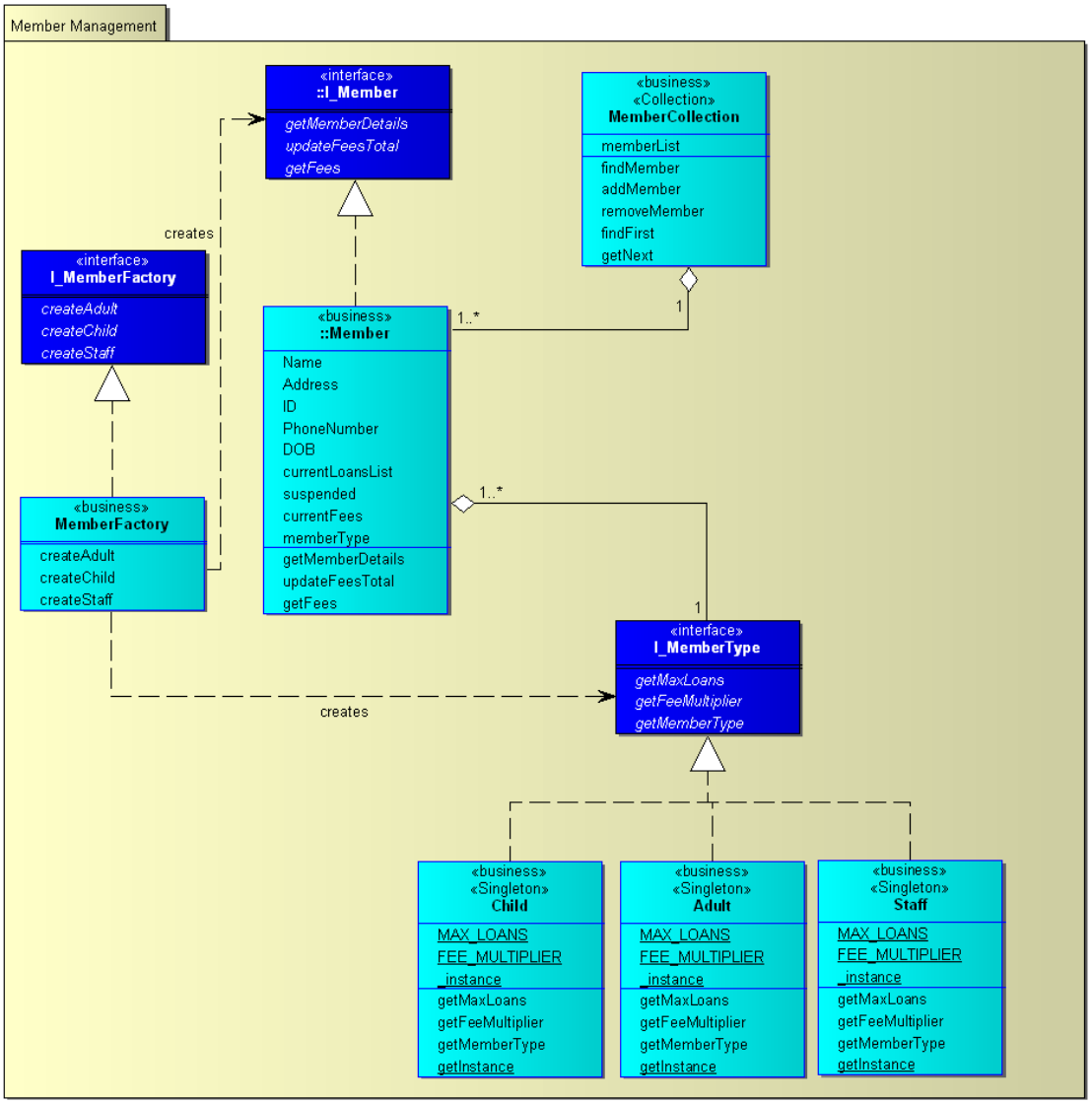


Figure 13. Member Management Package

8.3 Loan Management Package

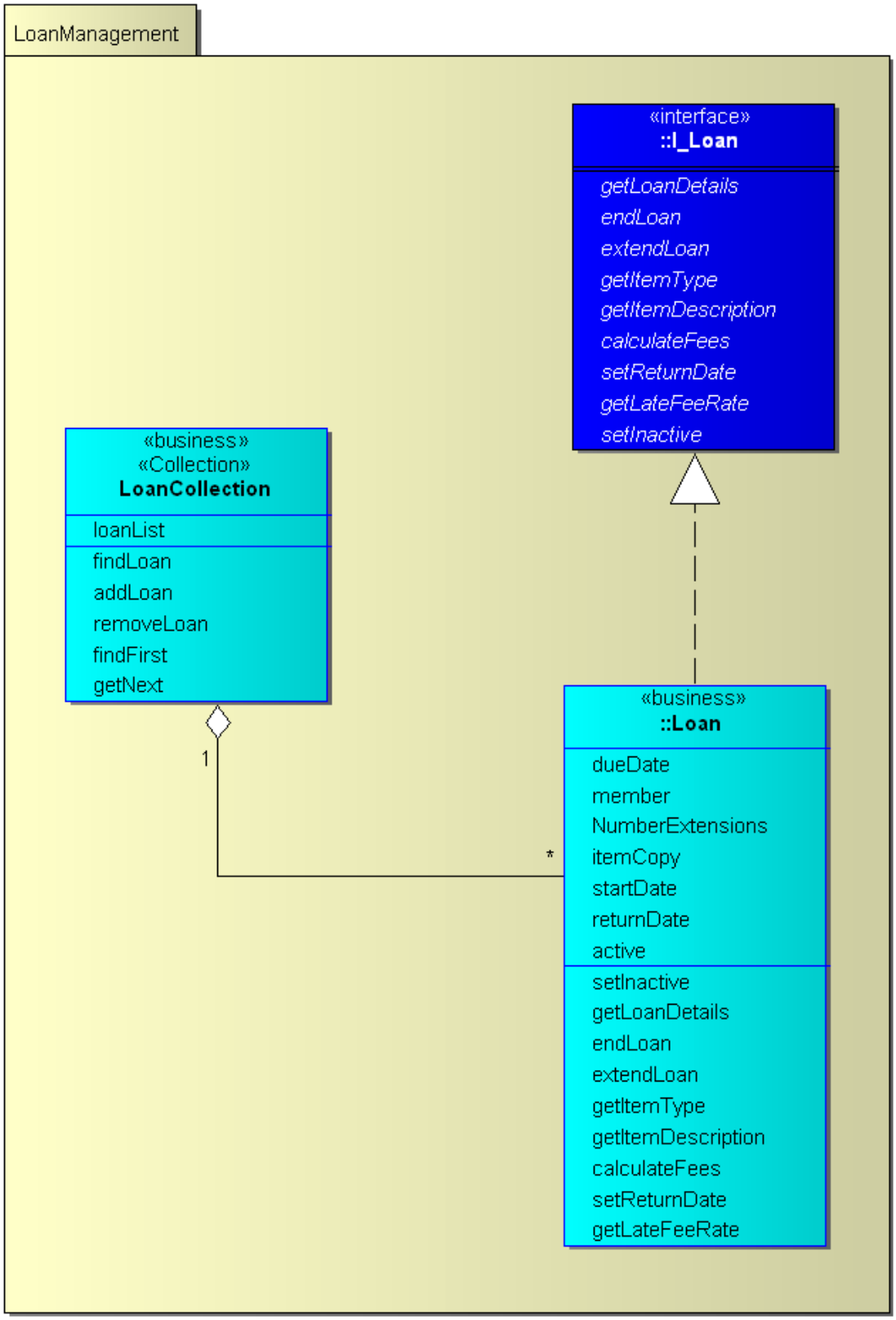


Figure 14. Loan Package Diagram

8.4 Stock Management Package

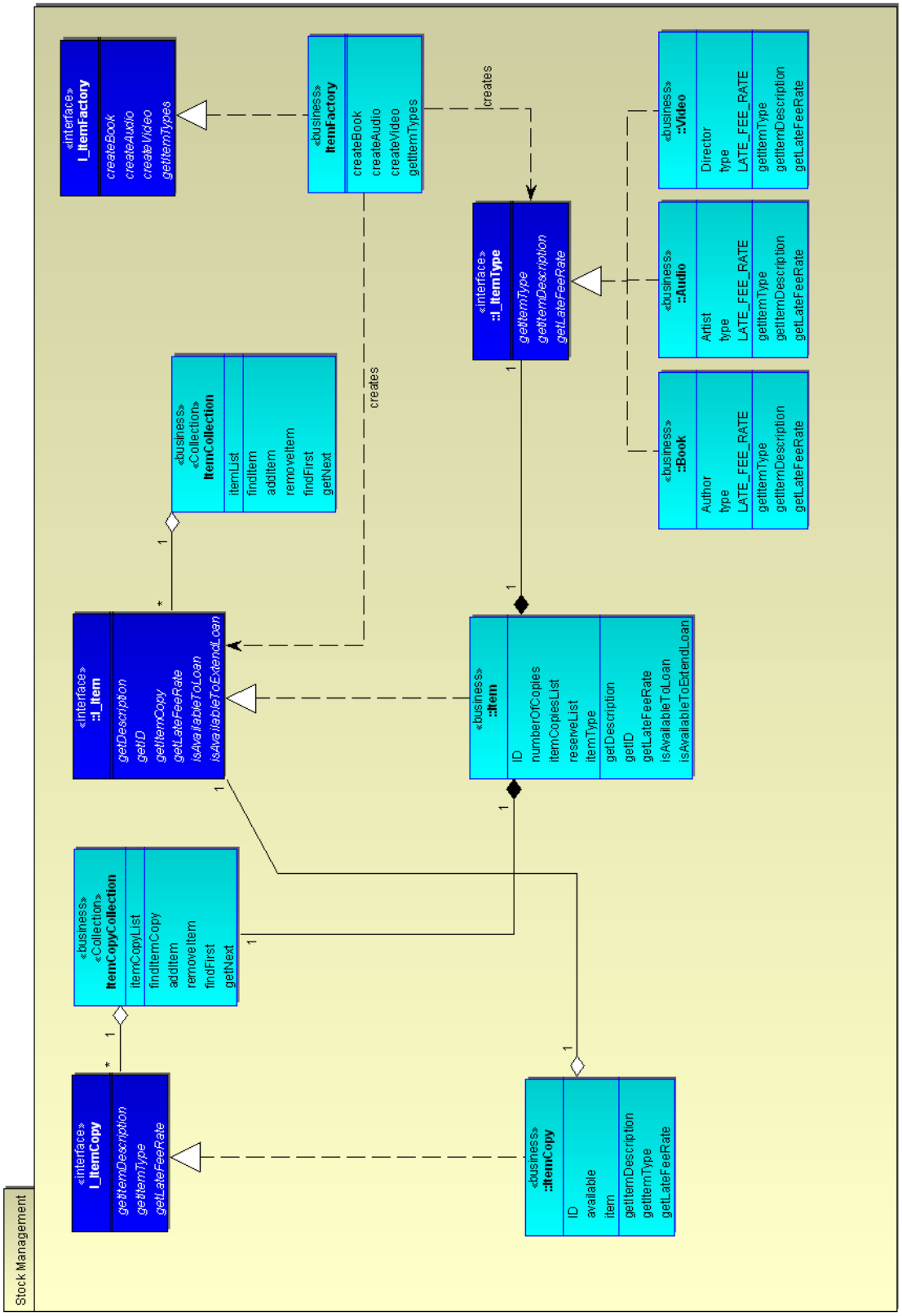


Figure 15. Stock Management Package

8.5 Staff Management Package

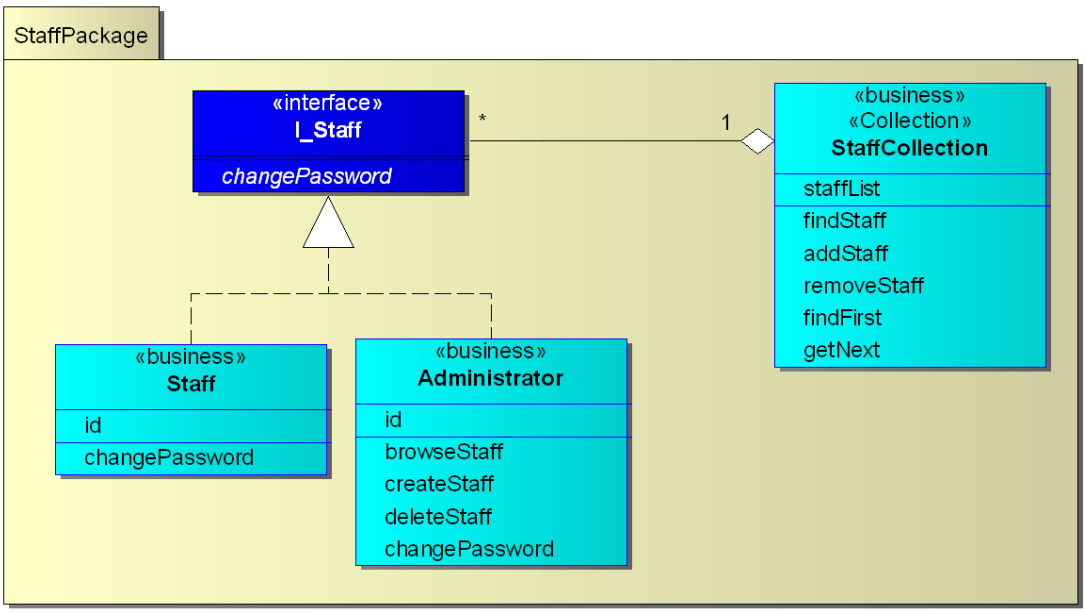


Figure 16. Staff Management Package

8.6 Database Management Package

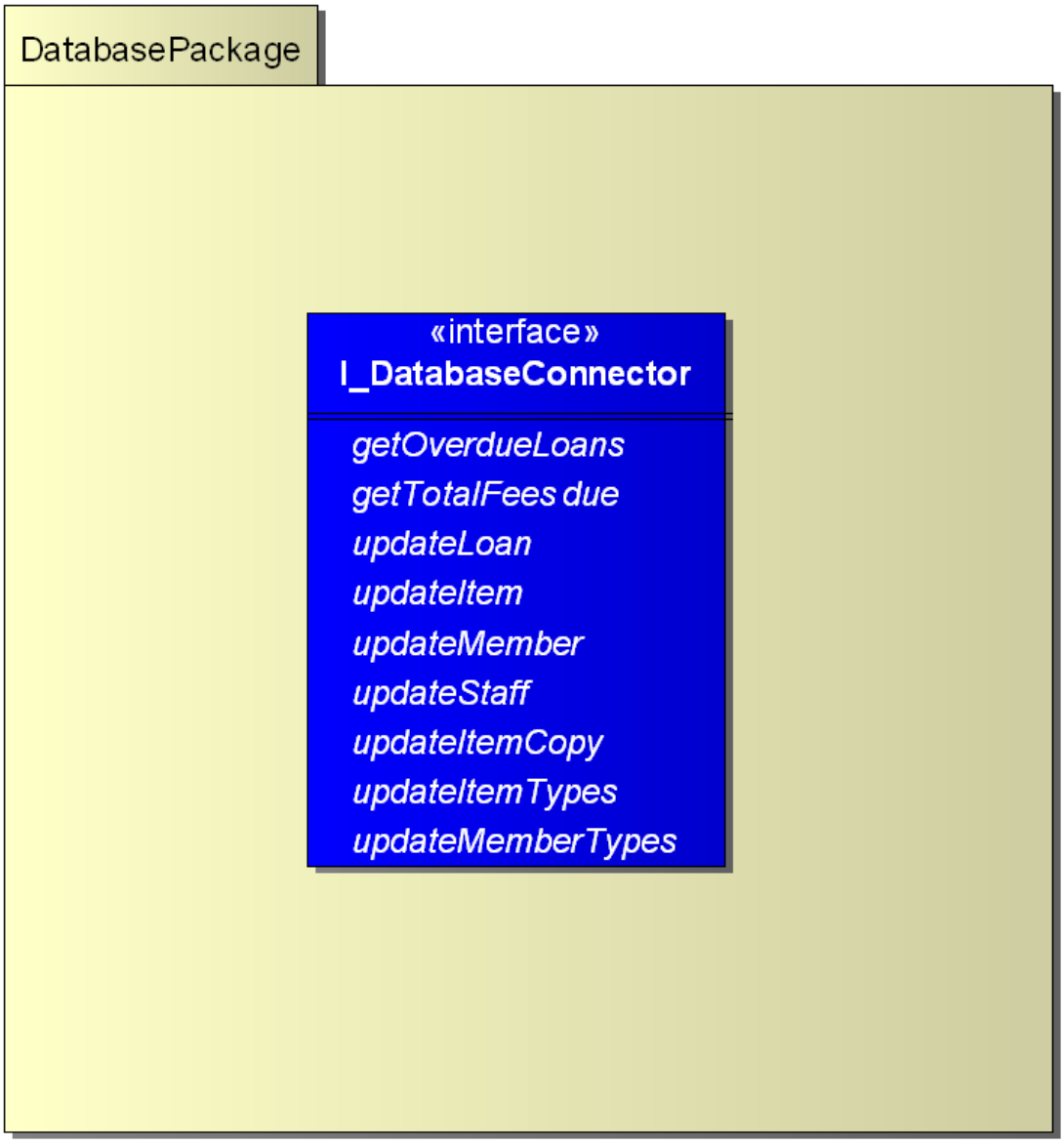


Figure 17. Database Management Package

8.7 Control Package

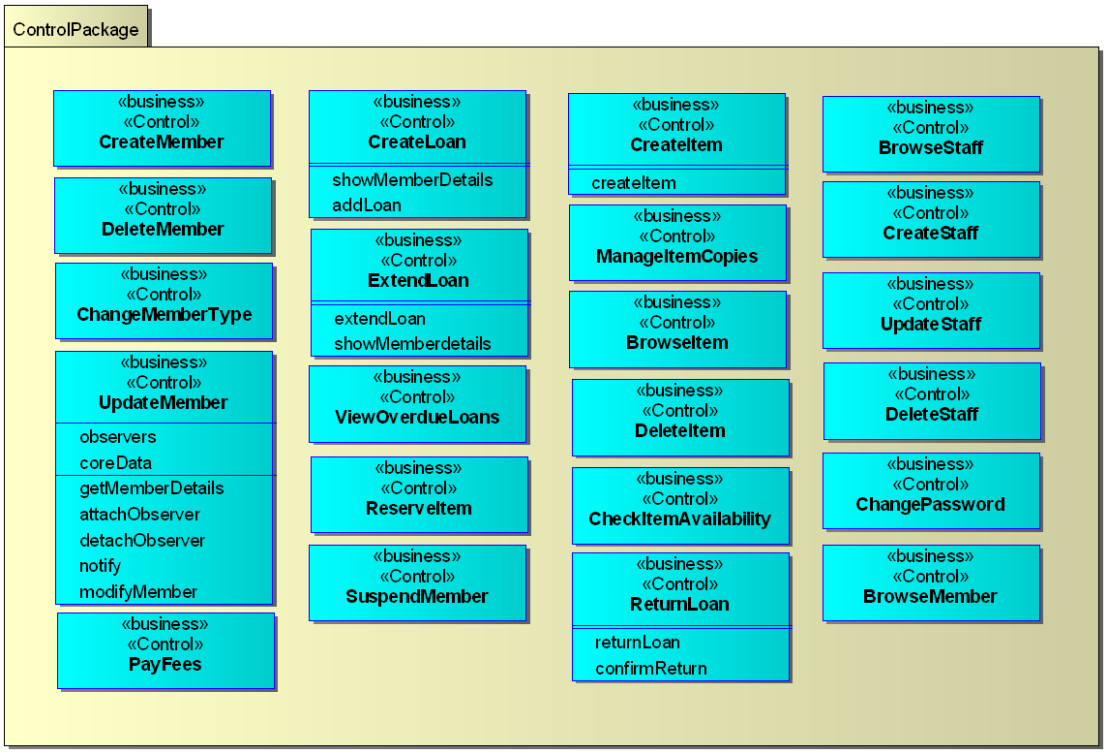


Figure 18. Control Package Diagram

8.8 Boundary Package

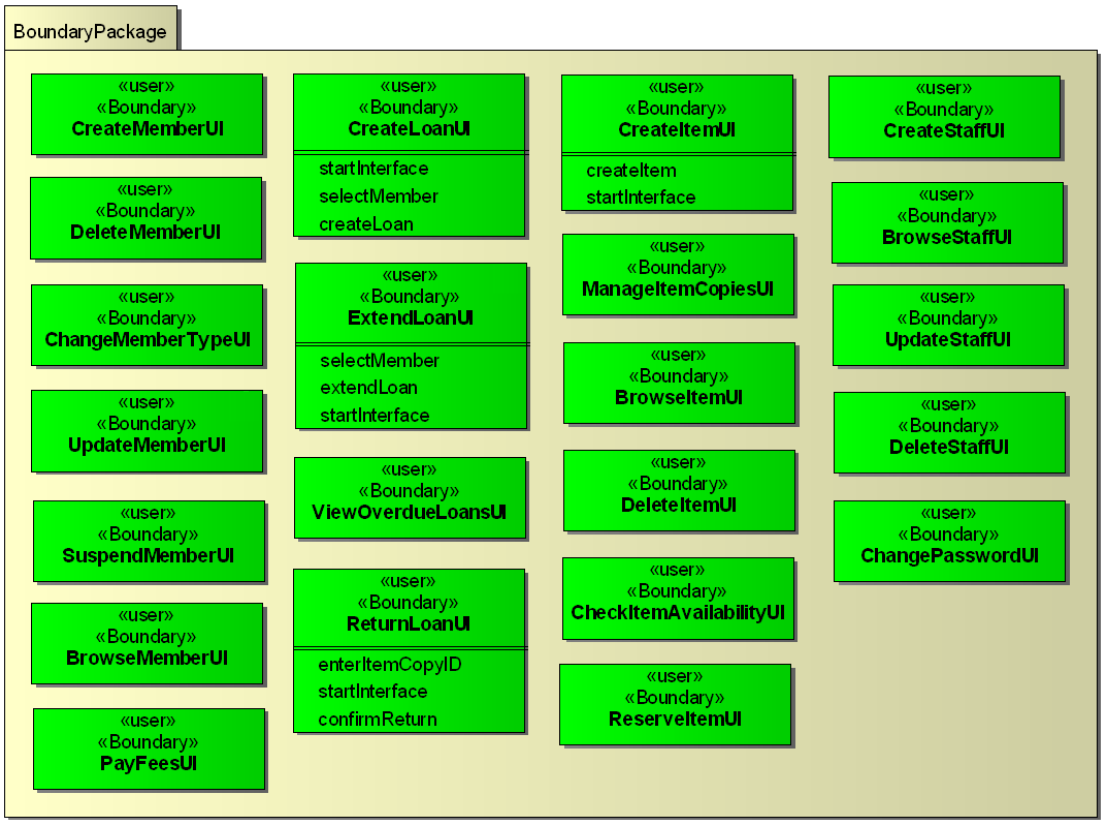


Figure 19. Boundary Package Diagram

8.9 Return Loan Sequence Diagram

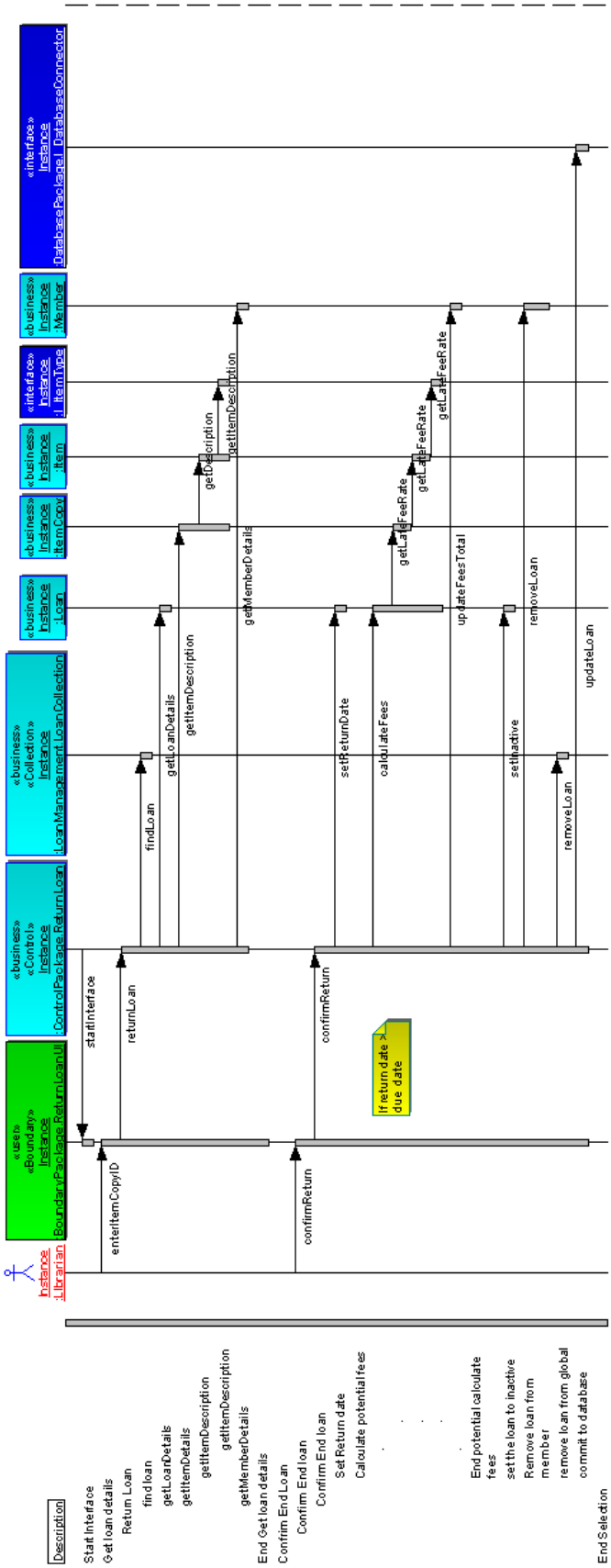


Figure 20. Return Loan Sequence Diagram

8.10 Return Loan Communication Diagram

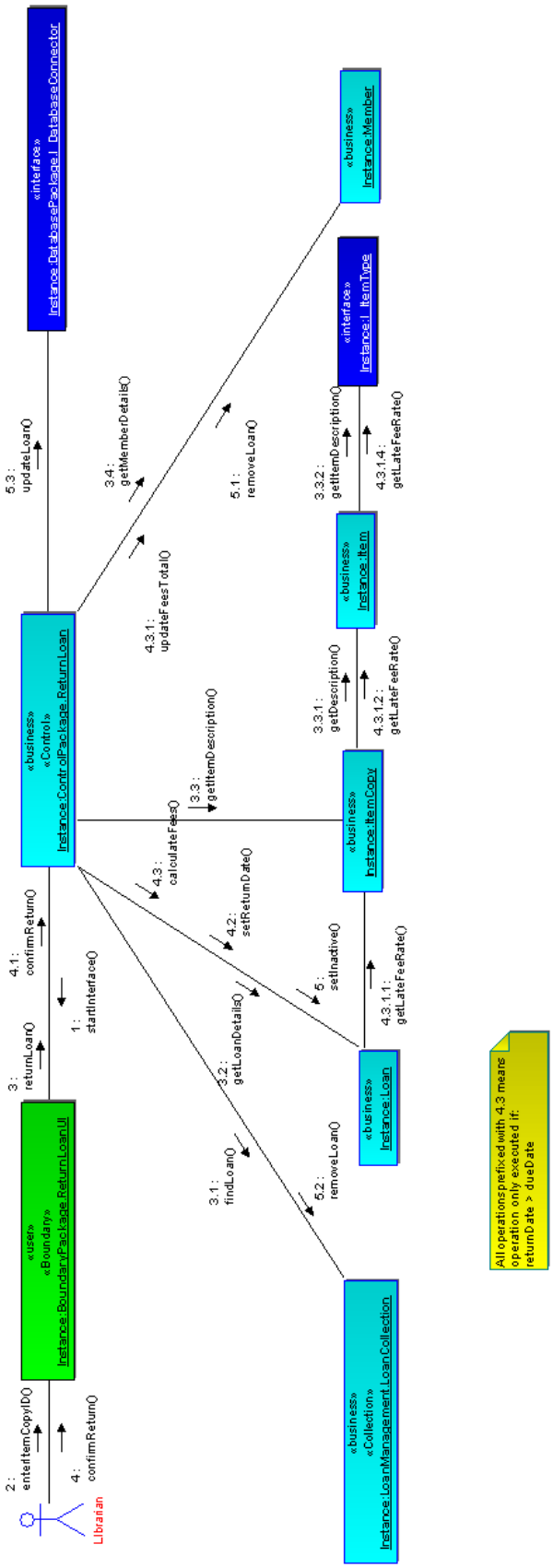


Figure 21. Return Loan Communication Diagram

8.11 Create Item Communication Diagram

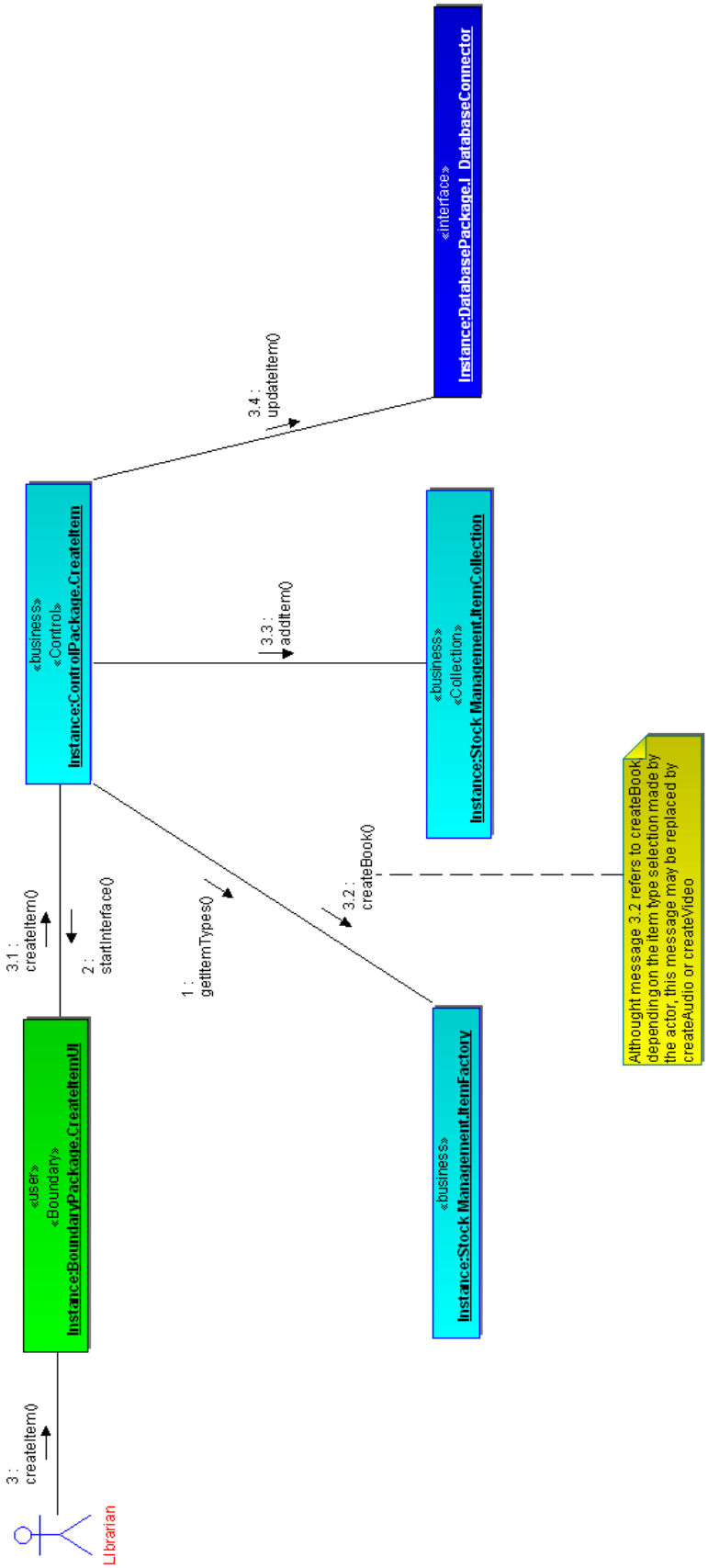


Figure 22. Create Item Communication Diagram

9. Design Discussion

9.1 Design Patterns Used

9.1.1 State Pattern

The state pattern is a behavioural pattern which allows an object to appear to change its class at runtime through an internal state change.

Consequences **Taken from Design Patterns – Elements of Reusable Software by Gamma, Helm, Johnson, Vlissides*

- State behaviour is localized and the behaviour for different states is separated.
- State transitions are made explicit.
- Where a state object has no attributes relevant to a specific Context object it may be shared among the Context objects.

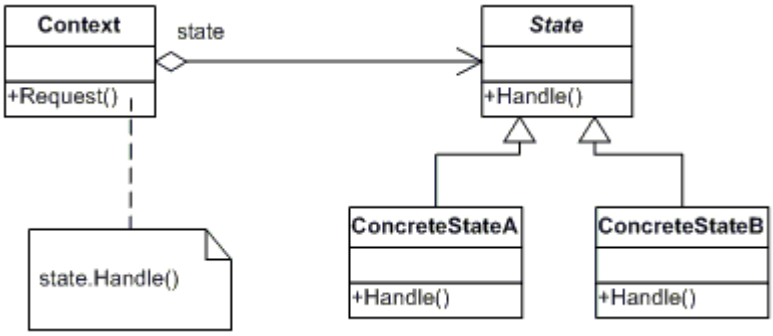


Figure 23. State Pattern Class Diagram

(Image: http://www.dofactory.com/Patterns/PatternState.aspx)

The state pattern is used in the library system in two packages. The first use allows a library member to be made a particular type of member. A library member includes a reference to a member type interface which can be one of the three member types: Adult, Staff, or Child. Each of these have different attributes such as the maximum loans allowed and the multiplier used to calculate fees due. The second use of this pattern is to assign a type to an item. An item can have a reference to a Book, Audio, or Video type which can return descriptions and late fee rates.

9.1.2 Factory Method Pattern

The factory method pattern is a creational pattern which encapsulates the code to create new objects inside public methods and removes the need for clients to access constructors.

Consequences **Taken from Design Patterns – Elements of Reusable Software by Gamma, Helm, Johnson, Vlissides*

- Factory methods eliminate the need to bind application-specific classes into your code.
- Clients might have to subclass the Creator class just to create a particular ConcreteProduct object.
- Creating objects inside a class with a factory method is always more flexible than creating an object directly.
- Clients can find the factory methods useful especially in the case of parallel class hierarchies.

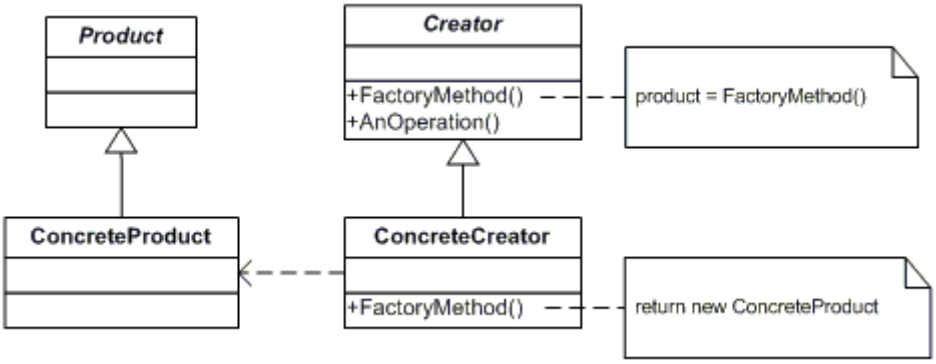


Figure 24. Factory Method Pattern

(Image: http://www.dofactory.com/Patterns/PatternFactory.aspx)

This pattern is used in the system to abstract the creation of members and items into methods. As both these types use the state pattern, they both require the creation of two objects. By using the factory method, an operation can be included in the factory class to create each possible type that the state pattern allows which means that items or members can be created through a call to one method rather than a number of calls to create objects and set the state reference. This pattern also removes the need to allow clients access the constructors for these classes.

9.1.3 Singleton Pattern

The singleton pattern is a creational pattern which ensures only one instance of a class can exist.

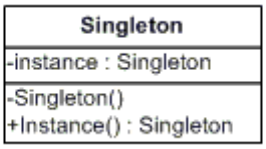


Figure 25. Singleton Pattern Class Diagram

(Image: http://www.dofactory.com/Patterns/PatternSingleton.aspx)

The singleton pattern is used to ensure only one object of each of the three member type classes are created. This is used as each of these classes only contain static attributes so only one instance of each class is needed and this can be assigned in the relevant creation method in the factory method pattern mentioned above.

Consequences **Taken from Design Patterns – Elements of Reusable Software by Gamma, Helm, Johnson, Vlissides*

- Controlled access to sole object.
- Reduced name space.
- Permits refinement of operations and representation.
- Permits a variable number of instances.
- More flexible than class operations.

9.3 Operation Specification Using O.C.L.

Item::removeReserve(member: I_Member) :boolean	
pre:	member->exists reserveList->find(member) == true
post:	reserveList->find(member) == false

Loan::endLoan( ) :boolean	
pre:	member->exists itemCopy->exists active == true
post:	active == false returnDate == currentDate

Loan::extendLoan( ) :boolean	
pre:	member->exists itemCopy->exists active == true itemCopy->isAvailableToExtend == true member->canExtend == true
post:	None

Member::addLoan(nLoan: Loan) :boolean	
pre:	numOfLoans < memberType->MAX_LOANS suspended == false

Member::addLoan(nLoan: Loan) :boolean	
post:	numOfLoans == numOfLoans[previous] + 1 nloan->member == this nloan->startDate == currentDate

MemberFactory::createChildMember( ) :I_Member member	
pre:	None
post:	member->exists member->id == UNIQUE member->memberType == Child

9.4 Architectural Pattern Considered

9.4.1 Model View Controller Architectural Pattern

The Model View Controller (MVC) pattern separates an application into three major types of components:

- Models that comprise the main functionality
- Views that present the user interface.
- Controllers that manage the update to views.

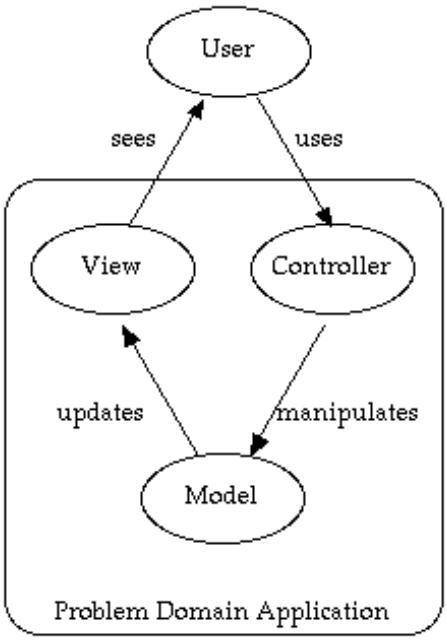


Figure 26. MVC Diagram

(Image: http://cristobal.baray.com/indiana/projects/mvc.html)

The MVC ensures the user sees the latest data even if other users have updated it since this particular user opened this view. As the Portumna Community Library will only consist of one computer used by one user at a time, this will never occur so the MVC pattern is not applicable to the community library system. However, the class diagram fragment and sequence diagram on the following pages show how the system would be modified to include the MVC architecture.

9.4.2 Model View Controller Class Diagram Fragment

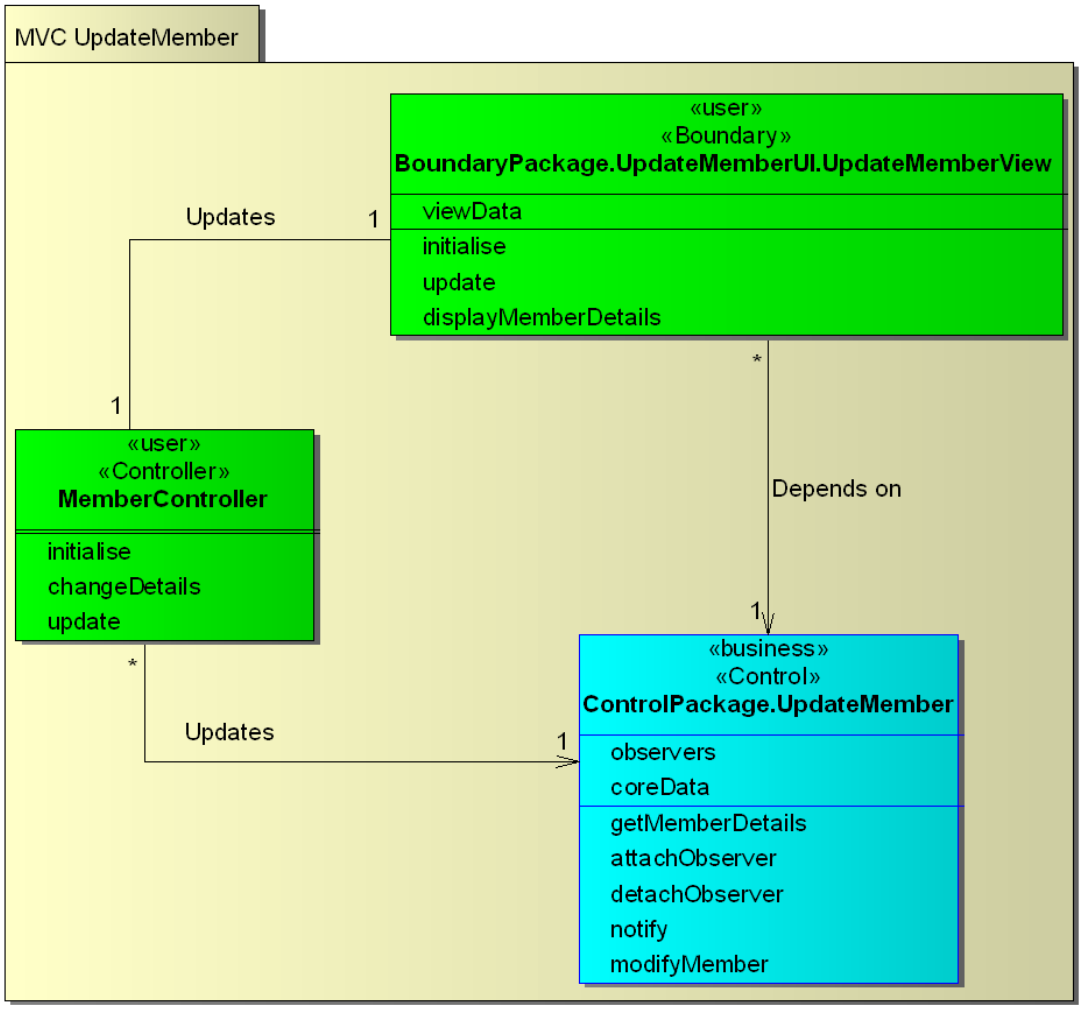


Figure 27. MVC Class Diagram Fragment

9.4.3 Model View Controller Sequence Diagram

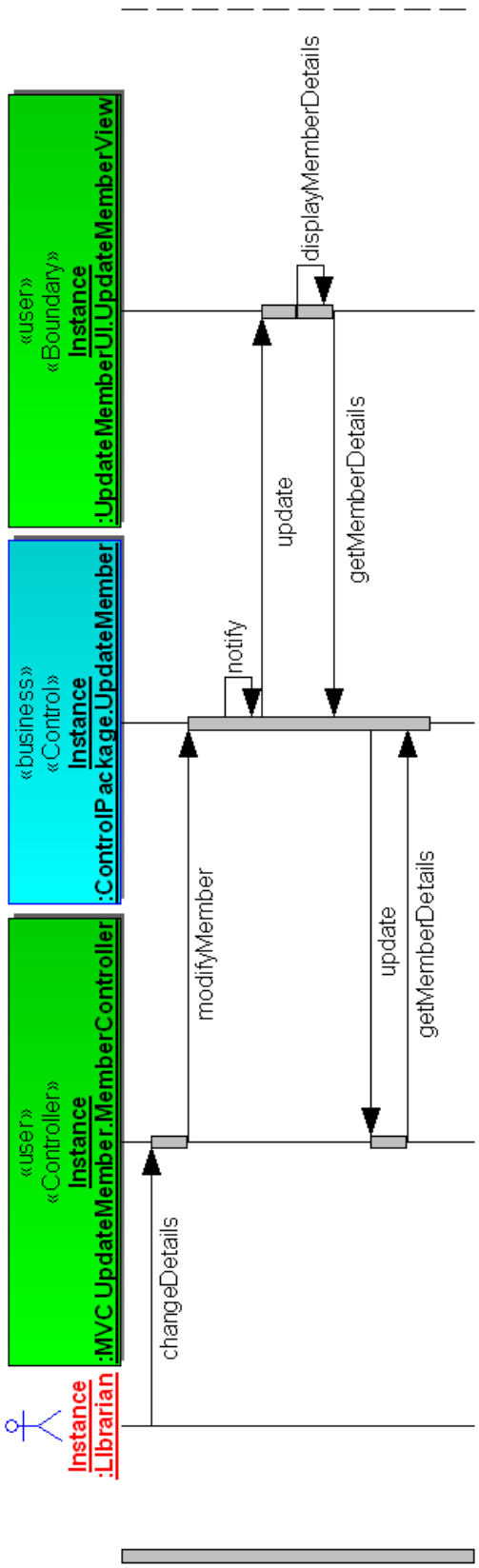


Figure 28. MVC Sequence Diagram

9.5 Architectural Considerations Summary

Following the analysis phase, interfaces were placed in each package to provide the necessary externally visible operations. This ensured conformance to the "Program to Interfaces, not Implementation" paradigm.

Considerations for possible future expansions were mainly based around a distributed system that networked many libraries together. The *client* in our system can be thought of as the *boundary* and *control* packages. Together, these connect to the underlying business tier through specified interfaces. Therefore the client-server architecture is already in place, meaning transition problems for a networked solution would be minimal. In such a transition the system may also be required to be distributed amongst several servers to share the workload. This is one reason why coupling between packages is undesirable. For example if one package was not dependent on any other package then this could be maintained easily on a different server. However this is not always possible and in those cases scaling at a local level is more appropriate. This would involve upgrading local machines to accommodate for greater computation ability.

Code-reuse also becomes an issue when talking of package dependencies. This is especially true for component based development (CBD) where the software is developed by combining and integrating different packages or components. These components are ideally self-contained and do not require any external software or code in order to function correctly. Even though efforts were made to cater for this style of development, in some cases it was necessary to have links between packages, e.g. *LoanManagement* and *MemberManagement*. However it may be that in later iterations these dependencies get removed.

Modifiability was an important factor from the beginning when the client stated that changes to the system were likely to be needed in the future. This is intrinsic in the multi-tier architectural approach to designing systems. It decouples the business logic of the system from the user interface and the database. Interfaces are provided for both of these connections so if one of these aspects was to change then a new implementation would be given without affecting the business layer. In future modifications the member types and item types are entities that would most likely be ones that get changed or added to. Therefore the intention was to extract this data and allow other types to be added post deployment at run-time. The *state-pattern* was used to achieve this. It essentially provides a unified interface to different implementations. For example the permissible member types are *child*, *adult* and *staff*. A future modification may be the desire to add a new type, such as *senior*. This can be easily be done because the client connects to the type only through an interface.

The issue of reliability and security is persistent amongst systems that maintain people's records and details. Security is not a major issue here because the system is locally deployed with administrator oversight on system level operations. If this was to become a problem in the future it is most likely that a third party solution would be attained. Reliability on the other-hand is a big concern. Luckily the DBMS will provide transaction support allowing the system to make atomic updates to the database. The data in the application that maps to the database is committed after every successfully completed operation sequence. If, before this, an error is

encountered or the system crashes then none of the data being operated on will be preserved. Instead, the system will rollback to a previous stable state. Take for example the *OverdueLoan* sequence of operations. It goes through all operations, updating the member and item accordingly, then commits this newly changed data to the database through the database layer.

Our system largely took the standpoint of modifiability over performance and on a number of occasions, modifiability has been chosen over performance. For instance the decision to use the state pattern extracts type differing attributes of the members and items. This provides an extra function call to retrieve this specific data, however the benefit is that different types may be added at runtime. Performance has also been affected by the use of collection classes. These collections provide an easier way of managing the many objects in the system but the result will be : 1) the system to take longer to load as the collections are all populated when the system initially starts and 2) the operations to manage instances are delegated to a new class.

10. Data Dictionary

Model: PortumnaLibrarySystem
Date: Thursday, April 23, 2009
Time: 13:10:10

Report: Item details (brief)
This report contains a list of all dictionary items (of selected types) and their details which contain information.

Name: LIbrarian
Type: Actor
Description: A librarian is a staff member of the library, they can be full time or part time. They carry out all interaction with the system on behalf of the libarary members. Certain staff (e.g. full time staff) will have administrator rights to the system which allow them to maintain staff account records. All staff are allowed change their own password.
Constraints:

Name: member
Type: Actor
Description: A library member can be one of 3 types: Child, Adult or Staff member. Each of these types can have different item maximum values and associated late fee values (via the feeMultiplier values).Each library member has a unique ID within the system.
Constraints:

Name: Member
Type: Class
Description: A member class holds attributes and behaviour to manage a library member details, associated loans, & associated overdue fees.
A member can be of type Adult, Child or Class.

Name: Loan
Type: Class
Description: Loan class holds all attributes and required behaviour of a item copy loan to a libraray member life cycle. It's period attributes are used to decide on fees required on return of loan item.
The loan is referenced by a member ID and an item Copy ID.

Name: Item
Type: Class
Description: A library item, it can be either of book, audio or video item type which will have 0..* item copies associated.
Also contains behaviour to control the availibility of the associated item copies and the assocaited member reserve list who wish to loan one of the related item copies.

Name: ItemCopy
Type: Class

Description: A unit of a library item with a unique item copy ID. All related item data is stored within the item class. ach item class has a access to a item copy collection for efficient to its related copies.

Name: Book
Type: Class
Description: Book is one of 3 current item types stocked by the library, it will have a related late fee rate potentially different to the other types which are currently Audio and Video.

Name: Audio
Type: Class
Description: Audio is one of 3 current item types stocked by the library, it will have a related late fee rate potentially different to the other types which are currently Book and Video.

Name: Video
Type: Class
Description: Video is one of 3 current item types stocked by the library, it will have a related late fee rate potentially different to the other types which are currently Book and Audio.

Name: Child [Member Management.]
Type: Class
Description: The member type "Child" refers that those library members whom are under 18 years of age. They may have lower loan capacity and fee terms than "Adult" members. The other member types being "Adult" and "Staff".

Name: Adult [Member Management.]
Type: Class
Description: The member type "Adult" refers that those library members whom are over 18 years of age. They may have higher loan capacity and fee terms than "Child" members. The other member types being "Child" and "Staff".

Name: Staff [Member Management.]
Type: Class
Description: The member type "Staff" refers that those members who are also library staff (part time of full time librarians). These will in general have favourable capacity and fee terms. The other member types being "Child" and "Adult".

Name: ItemCopyCollection [Stock Management.]
Type: Class
Description: A collection class holding all references to item copy object associated with an item class.

Name: LoanCollection [LoanManagement.]
Type: Class
Description: Loan Collection class will hold all references to active loans within the library system to improve throughput efficiency.

Name: Administrator [StaffPackage.]
Type: Class
Description: Administrator refers to the librarian with security rights to the system which gives allows them to maintain theirs and other staff records including system access password.

Name: CreateLoan [ControlPackage.]
Type: Class
Description: Control class that facilitates the successful creation by the librarian of an item loan to a library member. Functionality will include:
Validating member and item copy
Showing any member outstanding fees and links to fee payment
Qualifying that member has not exceed their max loan limits
Qualifying that item copy is available to loan (i.e. not reserved)
If successful loan created, removing member from item reserve list if relevant.

Name: ExtendLoan [ControlPackage.]
Type: Class
Description: Control class that facilitates the successful extension of a loan for a library member by the librarian. The functionality will include:
Validating member and item copy on loan to member.
Showing any member outstanding fees and links to fee payment
Qualifying that item copy is available to loan extension (i.e. not reserved).

Name: ReturnLoan [ControlPackage.]
Type: Class
Description: Control class that facilitates the successful return of an item loan by a library member. Functionality will include:
Validating loan and item copy
Calculating late fees on returning loan & links to fee payment
Successful removal of loan from system and updating of item copy availability.

Name: CreateItem [ControlPackage.]
Type: Class
Description: Control class that facilitates the successful creation by the librarian of a new library item of either book, audio or video type onto the system.

Name: MemberCollection [Member Management.]
Type: Class
Description: A collection class to hold references to each active member within the system to facilitate efficiency.

Name: ItemCollection [Stock Management.]
Type: Class
Description: A collection class to hold references to valid item objects witrhin the system. Increases system effiency.

Name: StaffCollection [StaffPackage.]
Type: Class

Description: A collection class which holds references to the staff object within the system. Used to facilitate efficiency.

Name: I_ItemFactory [Stock Management.]
Type: Class

Name: ItemFactory [Stock Management.]
Type: Class
Description: A constructor class for creation of items of different item types (book, audio and video).

Name: MemberFactory [Member Management.]
Type: Class
Description: A constructor class for creation of members of different member types (Adult, Child and Staff).

Name: I_MemberFactory [Member Management.]
Type: Class

Name: Staff [StaffPackage.]
Type: Class
Description: Librarian that does not have administrator rights to the system. Generally part time staff will have "staff" rights, allowing them all access to library functions except for ability to maintain staff records and loan capacity and fee parameter values.

References

Object-Orientated Systems Analysis and Design – 3rd edition
[Bennett, McRobb & Farmer]

Using UML – Software Engineering with objects and components – updated edition
[Stevens & Pooley]

Design Patterns - Elements of Reusable Object Oriented Software
[Gamma, Helm, Johnson, Vlissides]