COMP3004 D2

Team : 22

Team Members:

Mohammad Makkawi 101311993

Zachary Kushner 101313951

Van Pham 101320140

Michael Boyer 101305518

Date: December 3rd

1. Introduction

This document presents the subsystem decomposition for four selected patron features in the HinLIBS Library Management System. The diagram follows UML package notation and aligns with the layered architectural style discussed. The four selected patron features are:

- Search Catalogue
- Borrow Item
- Return Item
- View Patron Account Details

Based on these features, the system is decomposed into four logical subsystems: **PatronUISubsystem, CatalogueSubsystem, CirculationSubsystem, and PatronAccountSubsystem**. Each subsystem groups its boundary, control, and entity classes according to the analysis and subsystem design principles.

2. System Decomposition Diagram

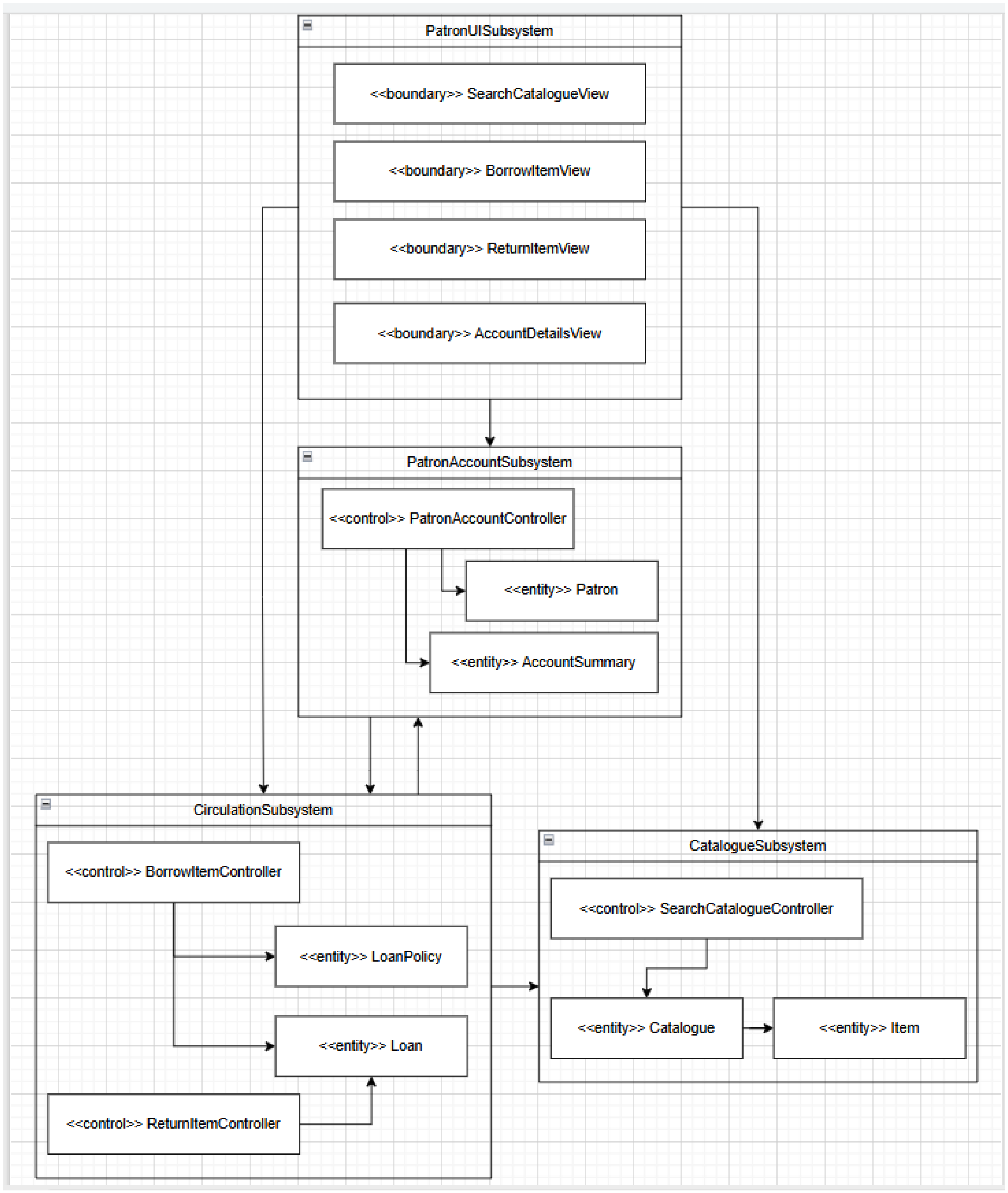
PatronUISubsystem
<<boundary>> SearchCatalogueView
<<boundary>> BorrowItemView
<<boundary>> ReturnItemView
<<boundary>> AccountDetailsView
PatronAccountSubsystem
<<control>> PatronAccountController
<<entity>> Patron
<<entity>> AccountSummary
CirculationSubsystem
<<control>> BorrowItemController
<<entity>> LoanPolicy
<<entity>> Loan
<<control>> ReturnItemController
CatalogueSubsystem
<<control>> SearchCatalogueController
<<entity>> Catalogue
<<entity>> Item

Figure 1: Layered (N-Tier) Subsystem Decomposition of HinLIBS for Patron Features

3.1 Cohesion

The subsystem decomposition is designed to maximize cohesion by grouping classes based on closely related responsibilities. The PatronUISubsystem contains only boundary classes responsible for user interaction. The CatalogueSubsystem, CirculationSubsystem, and PatronAccountSubsystem each focus on catalogue access, borrowing and returning workflows, and patron account management respectively. This grouping reflects strong alignment with the high-cohesion guidelines presented in the course's subsystem design lectures. For example, BorrowItemController, ReturnItemController, Loan, and LoanPolicy are grouped together because they all support the single functional concern of circulation.

3.2 Coupling

Coupling between subsystems is reduced by restricting cross-subsystem dependencies to a few well-defined entity classes. Boundary classes communicate only with their corresponding control classes rather than directly accessing entities. Subsystems expose narrow interfaces and hide their internal implementations, reducing interdependence and enabling safer modifications. Cross-subsystem dependencies are minimized by having controllers interact through a small set of shared entities (for example, BorrowItemController accesses Catalogue and Patron rather than multiple UI or account classes). For example, controllers in the CirculationSubsystem depend only on entity classes in the CatalogueSubsystem and PatronAccountSubsystem, not on UI classes.

3.3 Layered Architecture Alignment

The design follows an N-tier layered architecture similar to The MyTrip example. The PatronUISubsystem forms the presentation layer since it only contains boundary objects that manage interaction with the patron. The CatalogueSubsystem, CirculationSubsystem, and PatronAccountSubsystem form the application logic and domain layers, where the systems main processing and data abstractions live. Entity classes do not depend on boundary or control classes, so dependencies flow in one direction from UI to control to entity, this matches the layered architectural model and separation of concerns guidelines discussed.