

软件工程课程项目实践报告

(SE224 微电子学院 2019 春)

“二手书交易平台”小组

项目研究报告

作者：杨文曦

目 录

1 绪论	1
1.1 研究目的.....	1
1.2 研究内容.....	1
1.3 本文的组织结构.....	1
2.1 需求建模的目的.....	3
2.2 需求建模的方法.....	3
2.3 需求建模的项目实践.....	4
2.4 需求建模的实践心得.....	11
3.1 设计建模的目的.....	12
3.2 设计建模的方法.....	12
3.3 设计建模的项目实践.....	13
3.4 设计建模的实践心得.....	16
4 质量管理.....	17
4.1 质量管理的目的.....	17
4.2 质量管理的方法.....	17
4.3 质量管理的项目实践.....	18
4.4 质量管理的实践心得.....	19
5 软件过程与项目管理.....	20
5.1 软件过程与项目管理的目的.....	20
5.2 软件过程与项目管理的方法.....	20
5.3 软件过程与项目管理的项目实践.....	21
5.4 软件过程与项目管理的实践心得.....	22
6 总结与建议	23
6.1 工作总结.....	23
6.2 建议.....	23
参考文献	24
致谢	24

1 绪论

1.1 研究目的

本课程的研究目的是学习和实践在设计和构建软件时的各种规范和工程化方法，通过软件项目的建模和相应的开发实践，对软件工程的整个开发流程进行了解和学习。学习的重点在于对软件过程的理解以及对需求建模和设计建模的掌握，并将理论与实践结合，在实践过程中运用质量管理和项目管理的方法。

1.2 研究内容

本组开发的项目是基于 React 架构的“二手书交易平台”，属于 Web 应用。项目开发的初衷是建立一个 Web 端的二手书交易平台，既可以购买二手书，也可以销售二手书，从而解决校内二手书线上/线下交易不便的问题。

在实现上，本组计划利用 MVC 架构编写一个具有前后端，并使用 MySQL 和 MongoDB（最后仅成功运用了 MySQL）等外部系统。而在设计约束上，本项目利用 Java/JavaScript 语言编写前后端，并分别使用 Webpack 和 Maven 进行前后端打包管理，使用 GitHub 进行项目版本控制，并使用 MySQL 数据库等外部系统。

1.3 本文的组织结构

本文的组织结构主要分为六个部分。下一页的图 1 用树状图形式进行了直观的逻辑架构展示。

第一部分为需求建模部分，介绍了需求建模的目的和需求建模的方法和一些相关的关键概念，然后展示了个人的项目实践，即所建立的本用例的需求模型，分为总体描述、功能评述、外部接口需求和其他非功能需求四个部分，最后附有实践心得；

第二部分为设计建模部分，组织结构与第一部分相仿，仅在项目实践部分变为了架构设计和构件设计两部分，并介绍了一些使用到的 GoF 设计模式和实现方法，与第一部分稍有不同；

第三部分为质量管理部分，第四部分为软件过程与项目管理部分，两部分组织结构均与前两部分相似；

第五部分是本项目的工作总结以及关于本项目和本课程的一些建议；

第六部分列出了本项目研究报告的参考文献及致谢。

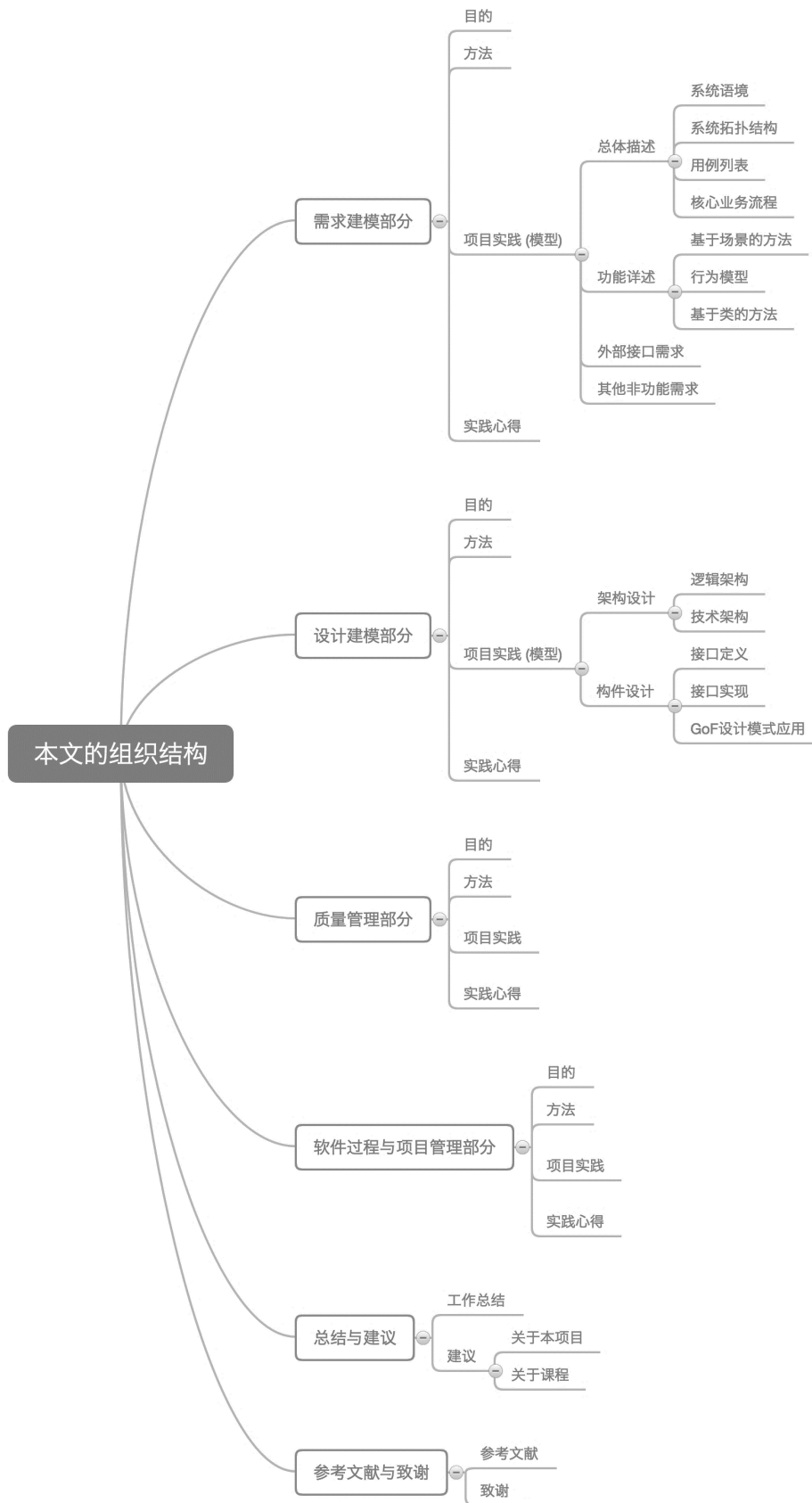


图 1 本文的组织结构

2 需求建模

2.1 需求建模的目的

需求建模的目的是以相对容易理解的方式描述需求，创建各种表现形式，从而使软件开发工作更为具体，同时为开发者和用户提供评估软件质量的手段。作为软件开发人员对本项目的共同理解，需求模型可以帮助团队更清楚地认识到需求是否正确、完整和一致，从而可以更好地开发、管理和维护本项目。

具体地说，需求模型需要描述客户的需求，包括问题的范围、运行的整体目标、优先级顺序、已知功能需求和系统将处理的对象，为软件设计奠定基础，并定义在软件完成后可以被确认的一组需求。需求模型是联系系统级表示层和软件设计模型的桥梁^[1]。

2.2 需求建模的方法

2.2.1 概述

需求建模分为传统的结构化分析方法和当前主流的面向对象分析两种方法。结构化分析是一种考虑数据和处理的需求建模方法，其中处理将数据作为独立实体加以转换，而数据对象建模定义对象的属性和关系，操作数据对象的处理建模则需要表明数据对象在系统内流动时如何处理和转换数据。

而另一种建模方法。即面向对象分析的方法，关注于定义类和影响客户需求的类之间的协作方式。本次课程项目的需求建模主要方法就是面向对象分析的方法，但对结构化分析方法也有一定的运用。

2.2.2 基于场景的方法

UML 需求建模需要从开发用例、活动图和泳道图形式的场景开始。基于场景的方法可以从用户的视角描述系统，从而更好地理解用户如何与软件交互，以及发现没有覆盖到的利益相关者所需的主要系统功能和特性。基于场景的建模方法的首要任务是获取开始编写用例所需要的基础信息，这一步骤称为起始和导出，这是需求工程的包括需要解决的问题的范围、整体的运行目标、优先级顺序和所有已知的功能需求^[1]。

在定义了需求后，需要进一步细化初始用例。在这一步骤中，需要对主场景中的各个步骤进行评估，精确定义和修改基本需求，从而创建一组次场景，例如一些附加操作和一些可能引起报错的操作及相应的处理。在此之后则需要编写正规的用例，并将较复杂的场景进行图形化，从而更好地、更准确地刻画需求特征，完成更有针对性的分析和设计模型。

2.2.3 基于类的方法

相比基于场景的建模，基于类的建模表示系统操作的对象。要从需求模型开发的使用场景中识别出可分类和定义的物理对象，需要运用分析类，即对用例的处理叙述进行语法分析，分离出潜在的类，再从中选择合理属于类的对象。而要识别分析类，基于类的建模使用从基于场景和面向流的建模元素中导出信息。

类-职责-协作者建模方法则可以识别和组织与系统或产品需求相关的类。前述的类可以拓展分类为实体类、边界类和控制类。实体类是从问题说明中直接提取出来的、保存在数据库中、贯穿应用程序的数据实体；边界类用于创建用户可见的、在使用软件时交互的接口；

而控制类全程对整个工作单元进行管理。此外，各种 UML 建模方法还可以定义类之间的层次、关系、关联、聚合和依赖，从而为整个系统提供了更好的管理。

2.2.4 行为模型

行为模型会将前述建立的需求模型静态元素转换成系统中的动态行为，显示了软件如何对外部事件或激励做出响应。在行为建模的场合下要考虑系统执行其功能时每个类的状态和从外部观察到的系统状态，其状态具有主动和被动两种特征。行为模型可以通过状态图呈现主动状态和相应导致状态变化的触发器，也可以通过顺序图，以时间函数的形式表明事件如何引发从一个对象到另一个对象的转移。

2.3 需求建模的项目实践

2.3.1 总体描述

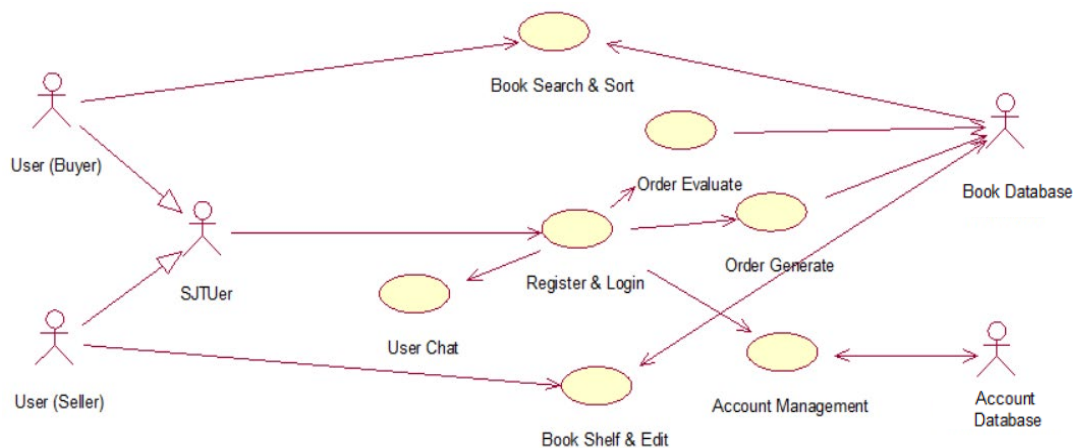


图 2 系统语境图

上图 2 为本项目的系统语境图，是一个 UML 用例图，明确了本系统的边界。可以看到，本系统有买家和卖家两类用户，而系统中包含了七个用例。其中，注册/登陆为所有用户共有，登陆后可以使用订单生成、订单评价和购物车等用例，除此之外还需要实现用户管理功能。而书籍上架/编辑为卖家独有用例，书籍搜索/排序为买家独有用例。

而外部系统方面，分为了用户账户数据库和书籍数据库两类。用户账户信息库用于存储用户账户信息，而书籍数据库用于存储书籍信息和对应的订单信息。在实际的实现中，将较简单的文本信息存储在 MySQL 数据库中，而将书籍封面等较复杂的信息存储于 MongoDB 数据库中。

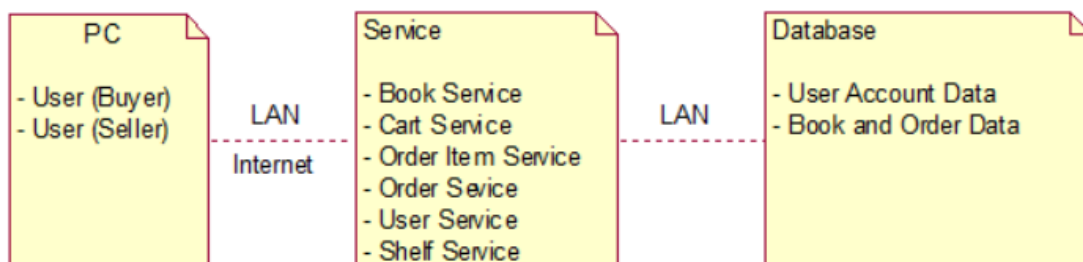


图 3 系统拓扑结构图

上图 3 为本项目的系统拓扑结构图，明确了本系统的部署方式。可以看到，用户在 PC

端通过互联网访问服务，而服务所需要的书籍信息会从数据库交换得到。其中，用户分为卖家和买家两类，服务则有书籍服务、购物车服务、用户服务和订单服务等多项服务，数据则分为用户账户数据和书籍及订单数据两类。

下表为本系统中包含的所有七个用例及其相关信息。

表 1 用例列表

<p>用例名：用户账号注册/登陆；</p> <p>用例概述：本用例实现用户的注册账号和登陆账号等功能。用户可在初始页面上选择要进行的操作。若选择注册，则跳转至注册页面，用户须填写用户名、密码、重复密码、手机号/邮箱，系统会实现相应的格式检验，若格式错误则会显示提示信息；在实现用户信息管理中的封禁功能之后，若被封禁，页面会显示相关提示信息；</p> <p>优先级：基础功能之一，高优先级。</p>
<p>用例名：商品增删/编辑；</p> <p>用例概述：本用例实现商品的增删和编辑功能。用户选择上架商品操作后，跳转至注册页面，用户需要填写书名、作者、ISBN 号、库存量、价格和上传封面照片，提交后系统将用户所填写的信息存入数据库；</p> <p>优先级：基础功能之一，中优先级。</p>
<p>用例名：商品检索/排序；</p> <p>用例概述：本用例实现商品的检索和排序功能，实现商品关于特定属性的检索和排序。在搜索页面，可选择按属性的顺序/倒序排序。而选中某本书后，利用 Ajax 方式显示详细信息；</p> <p>优先级：基础功能之一，中优先级。</p>
<p>用例名：用户问答；</p> <p>用例概述：本用例实现用户针对书籍的问答功能。在书籍详细信息页面下会有窗口，用户可以对该商品进行讨论。</p> <p>优先级：基础功能实现之后，低优先级。</p>
<p>用例名：订单生成/处理。</p> <p>用例概述：本用例实现订单的生成和处理。用户可以将书添加至购物车，然后在购物车页面查看要购买的书籍清单，点击购买后则生成相应订单，可在订单页查看；</p> <p>优先级：基础功能之一，高优先级。</p>
<p>用例名：订单评价；</p> <p>用例概述：本用例实现交易双方对订单的相互评价。点击订单详情后，跳转至评价页面，交易双方可以对该订单进行评价；</p> <p>优先级：中优先级。</p>
<p>用例名：用户信息管理。</p> <p>用例概述：本用例实现对用户信息的管理，主要功能为可以禁用特定的用户账号，从而禁止该用户登陆。</p> <p>优先级：基础功能之一，高优先级。</p>

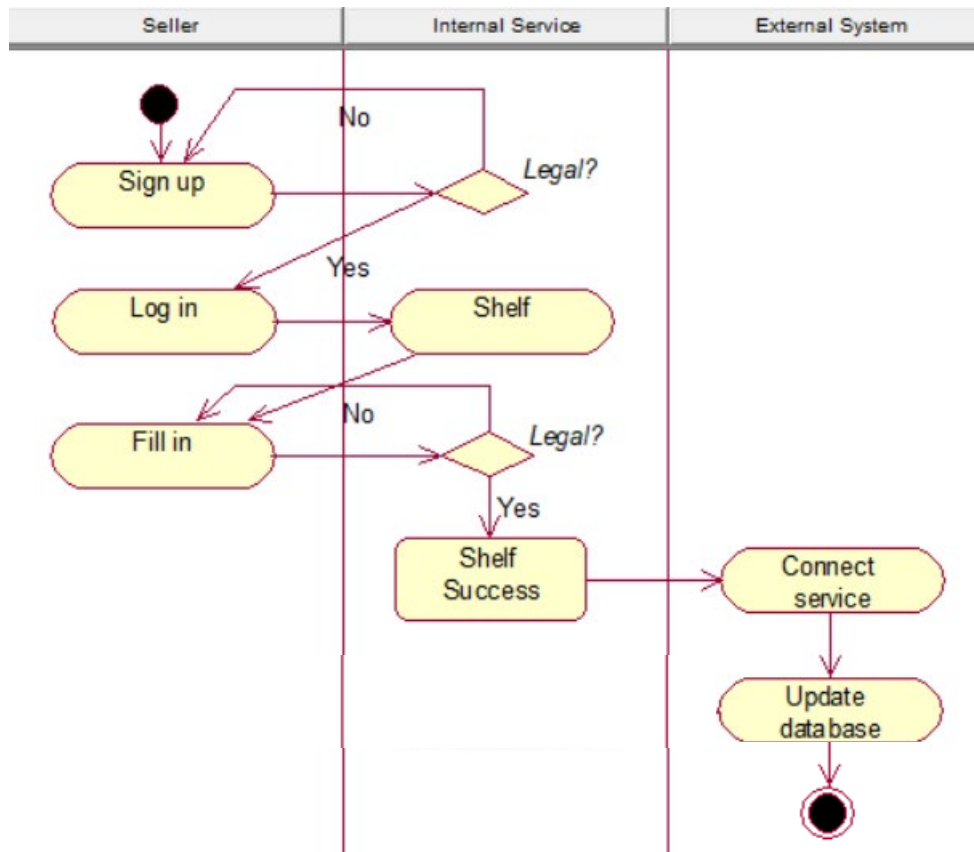


图 4 带泳道 UML 活动图：买家核心业务流程

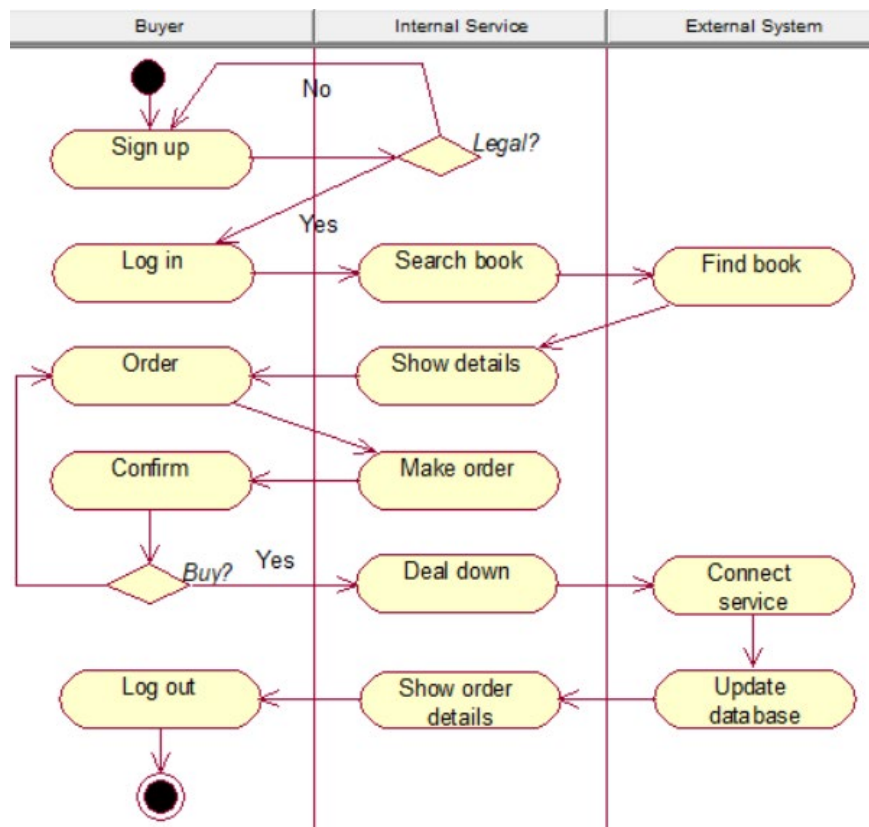


图 5 带泳道 UML 活动图：卖家核心业务流程

以上两张带泳道 UML 活动图（图 4、5）分别明确了买家和卖家的核心业务流程：买家注册，登陆，搜索书籍，查看详细信息后下单，确认后完成购买；卖家注册，登陆，上架书籍，填写表格，填写信息无误后提交即完成其核心业务流程。

2.3.2 功能详述

下表 2 为本用例的用例详述，依照正规用例模板描述了用例的典型描述要点。

表 2 商品增删/编辑：用例详述

<p>用例：商品增删/编辑：利用数据库实现管理（上架/下架/更改/删除）商品</p> <p>范围：Platform 二手书交易平台</p> <p>优先级：中等优先级</p> <p>何时可用：第一个增量</p> <p>使用频率：中等频率</p> <p>主要参与者：卖家</p> <p>涉众及其关注点：</p> <ul style="list-style-type: none"> - 卖家：希望能够便捷地上架/下架/更改/删除商品，并且能展示商品的关键信息； - 买家：希望能够清晰及时地了解到商品的关键信息。 <p>前置条件：卖家必须登陆账号</p> <p>后置条件：存储更新后的商品信息以便检索</p> <p>主成功场景：</p> <p>场景 1：上架商品</p> <ol style="list-style-type: none"> 1. 卖家点击上架商品按钮； 2. 系统开始一次上架操作； 3. 卖家选择图书分类； 4. 卖家填写商品信息（名称、作者、照片、品相、描述、ISBN 等）并提交； 5. 系统审核后记录信息，将信息记入商品信息库； 6. 显示上架成功页面，上架成功。 <p>场景 2：编辑商品信息</p> <ol style="list-style-type: none"> 1. 卖家点击编辑商品信息按钮； 2. 系统开始一次编辑操作； 3. 卖家更改商品信息（有一些信息需要锁定，无法更改）； 4. 系统审核后更新信息，并更新商品信息库； 5. 显示编辑成功页面，编辑完成。 <p>场景 3：下架商品</p> <ol style="list-style-type: none"> 1. 卖家点击下架商品信息按钮； 2. 显示确认提示信息； 3. 系统更新商品信息库； 4. 显示下架成功页面，下架完成。 <p>扩展：</p> <p>场景 4：必填信息空缺</p>

1. 卖家点击上架商品按钮；
2. 系统开始一次上架操作；
3. 卖家选择图书分类；
4. 卖家填写商品信息并提交，但必填信息有空缺；
5. 系统审核后发现必填信息空缺，提示报错；
6. 返回商品信息填写页面。

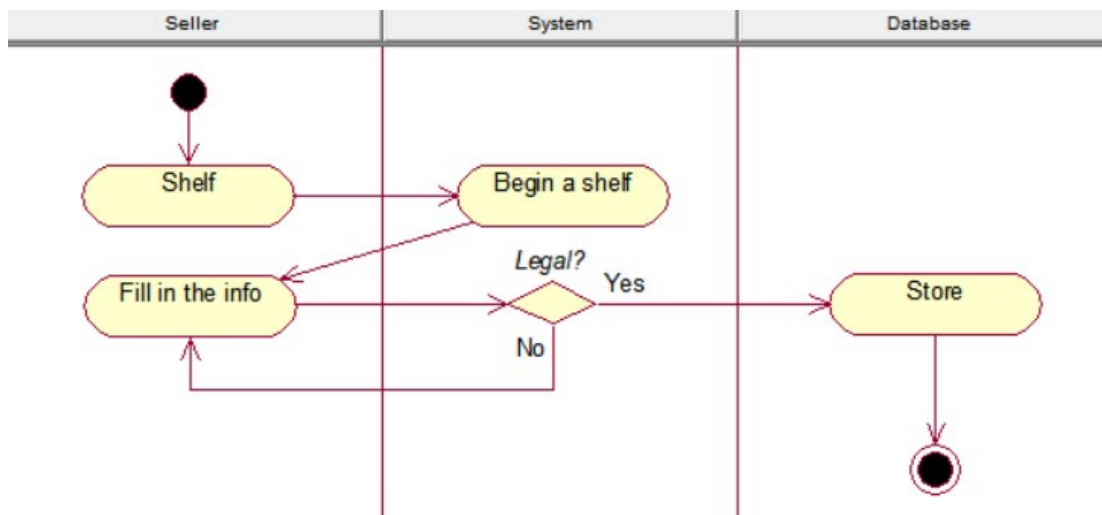



图 6 带泳道 UML 活动图：核心场景

上图 6 为展示本用例核心场景的带泳道 UML 顺序图。本用例的主要参与者为卖家。卖家请求上架操作，系统随即响应，开始一次上架操作，随后卖家填写上架书籍信息并提交。若信息填写无误，则系统将其提交至数据库存储，上架成功；否则将表单返回给卖家重新填写，直至信息填写无误。而下图为本用例的用户界面原型。



Drag and Drop or Upload Avatar Image

图 7 用户界面原型

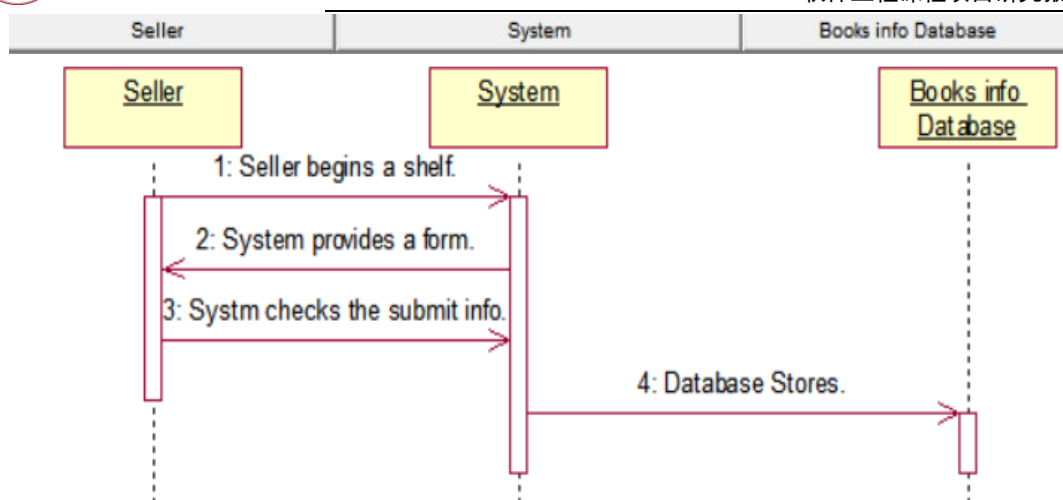


图 8 系统顺序图

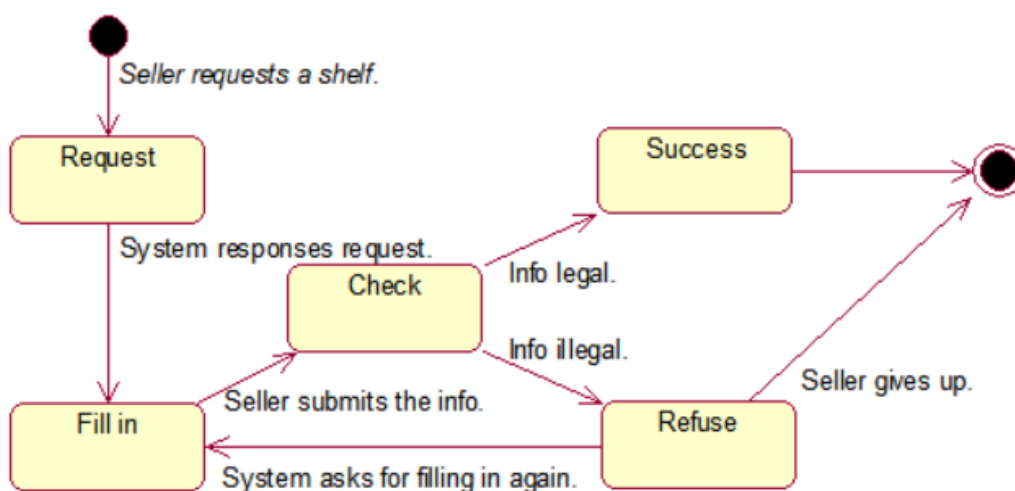


图 9 状态机图

以上两图（图 8、9）为本用例的行为模型，包括识别用例事件的系统顺序图和描述系统状态变化的状态机图，其内容与前述核心场景一致。

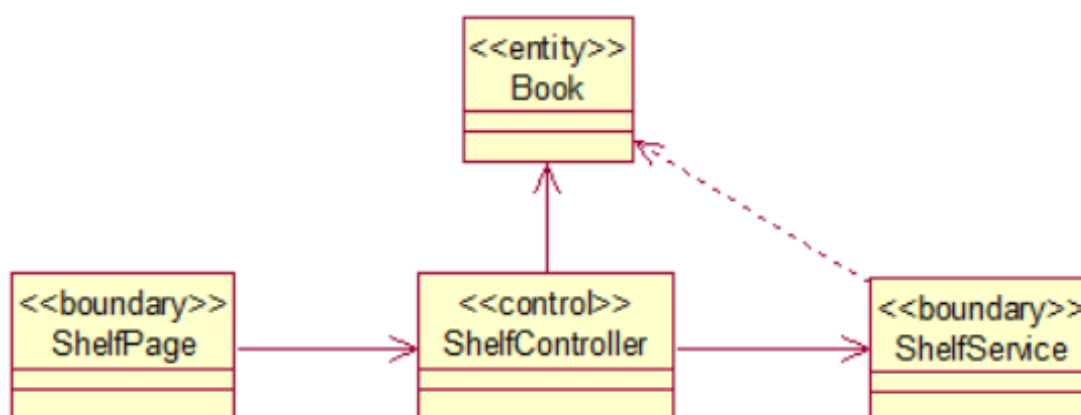


图 10 分析类

上图 10 为本用例所识别出的分析类，其中包含两个边界类，分别负责与用户和与外部系统交互；而有一个实体类，即上架的书籍，有一系列属性和对应操作；有一个控制类，负责整个用例活动的控制。

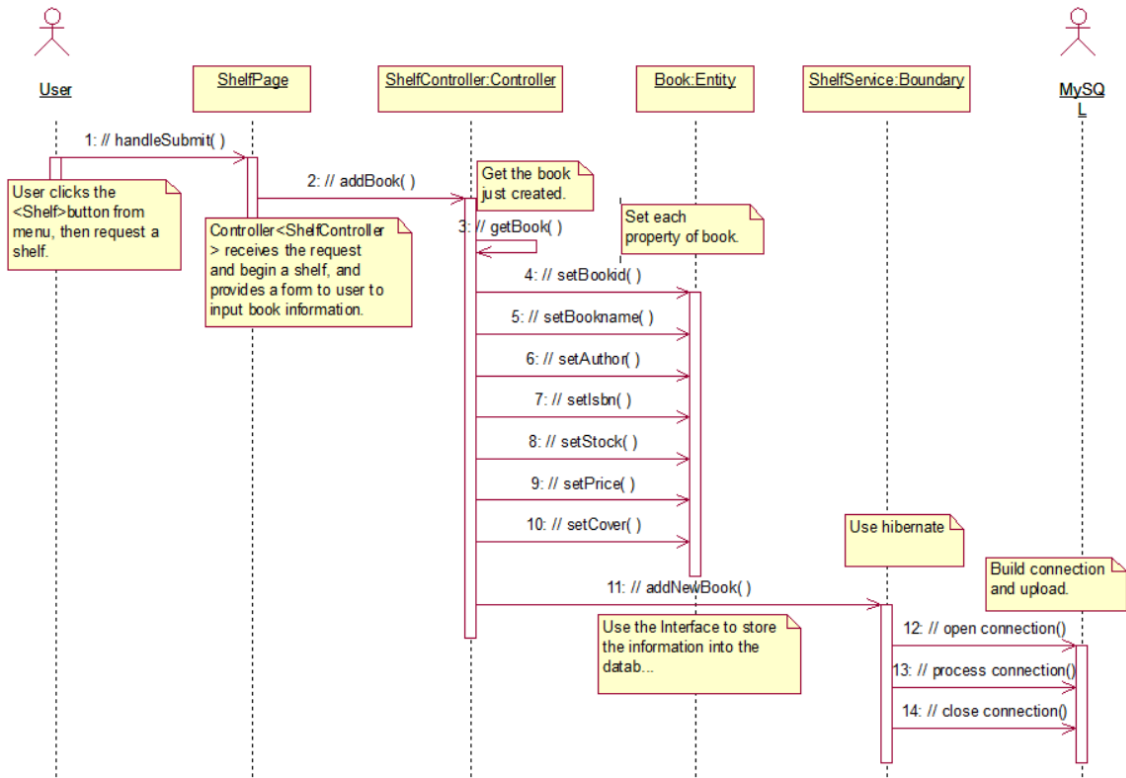


图 11 UML 顺序图

而上图 11 为用例实现的 UML 顺序图，从行为方面描述了以上分析类如何协作实现核心场景。由上图可以看出，两个边界类 ShelfPage 和 ShelfService 分别负责与用户进行交互，而 ShelfController 则是核心的控制类，负责发起商品上架操作，创建新书籍并提交信息。上图各处有相应注释。

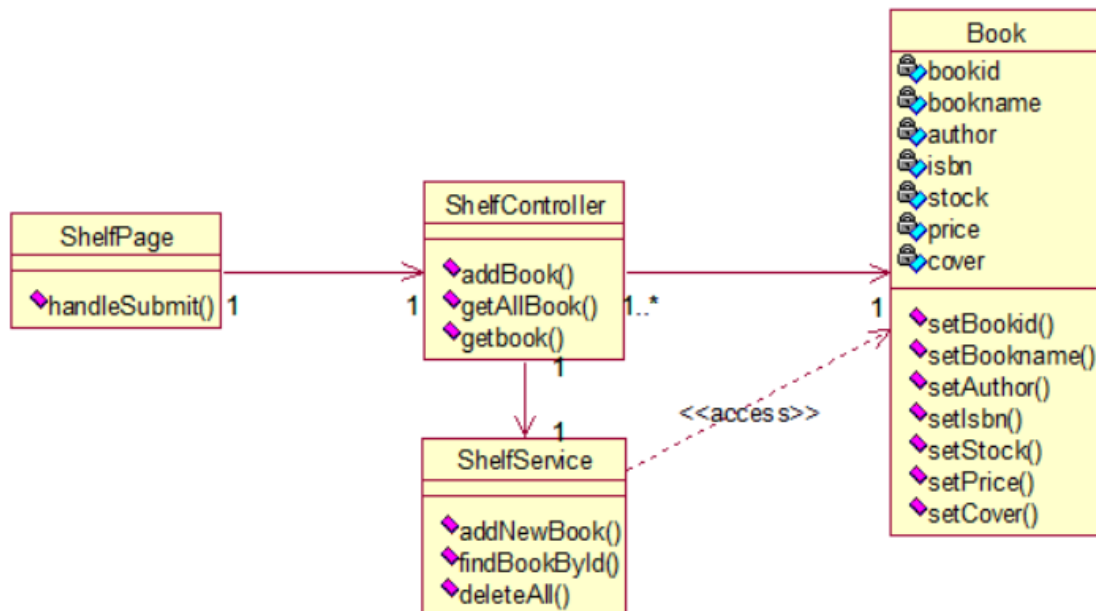


图 12 类图

而上图 12 为由分析类细化出的类图，从结构方面描述了分析类如何协作实现核心场景。其中四个类与图 10 中的分析类相对应，并添加了一系列相应操作。而在各类之间相互关系

方面，关联展示出了多样性，而 **Book** 类和 **ShelfService** 类展示出了依赖关系。

而在外部接口需求方面，本用例由于用到了 MySQL 数据库，故运用了 Java 的 MySQL API，即 JDBC。而本系统并未使用外部设备硬件接口。

在其它非功能需求方面，本用例开发方面并没有考虑诸如终端数支持、并行操作用户数支持和动态数值需求等性能需求；在安全需求方面，本系统仅考虑了强制访问控制；软件质量属性方面，本系统考虑了友好性（操作界面和报错信息从用户角度考虑）、可移植性（软件需使用 react 框架和 npm 管理系统，前端采用 Webpack 打包管理，后端采用 Maven 打包管理。数据库间的转换暂不考虑）和可拓展、可修改性（顶端有抽象接口设计的类层次结构，代码结构有充分逻辑，有一定注释且可塑性较强）。

2.4 需求建模的实践心得

本学期所有课程项目建模中，由于是初次涉及建模相关的知识和操作，所以需求建模部分花费的时间最多。先列写一下自己需求建模中遇到的主要问题：

1) 关于用例的命名：在最初开始导出用例时，本组的多个用例命名都不合规范，直到开题答辩后才进行了更正，从动宾结构短语改为了主谓结构短语。这样的命名是容易理解的，属于细节上的问题；

2) 关于分析类：在建立分析类时遇到的问题大概是需求建模部分遇到的最大困难，主要问题在于所建立的分析类和后续 UML 顺序图和类图的不匹配。究其原因，是对基于类的方法的需求建模理解不够深刻，尤其是对边界类的作用和意义理解不够；

3) 各部分操作的一致性：中期答辩之前所建立的需求模型中，行为模型与结构模型中的操作也有一些并不匹配，这属于建模时的疏忽，没有将模型中的各部分组织为一个统一的整体。这一部分在中期答辩中进行了修改；

4) 关于模型与代码的一致性：最初建立的模型中的逻辑较为简单，但在后续代码实现中遇到了困难，按照原逻辑无法实现或较难实现原定功能，故改变了实现方式。本报告在代码实现后对原有的需求模型进行了相应的修改；

5) 关于建模工具的使用：除上述需求建模具体内容的问题外，在最初使用建模工具时也出现了一些问题。最初使用的建模工具是 Enterprise Architect，但由于操作较为繁琐而改用了 IBM 开发的 Rational Rose。但最初使用 Rational Rose 时对整个软件并不了解，出现了诸如删除了图标却没从模型中删除，类之间关系标注错误等问题。在熟悉建模工具后解决了这些问题。

总的说来，通过需求建模这一部分的学习和本项目的对应实践，基本掌握了需求建模的方法。用这样工程化的方法来理解需求，在进行大型项目时能够梳理出非常明晰的逻辑，这为大型系统提供了行之有效的管理方法，对于整个项目的开发和质量评估非常有益。另一方面，这种需求建模的思想和其中的方法在软件工程之外的各种工程都可以适用，为今后各种工程或项目的开展提供了一种很好的思路。另一方面，要最大化需求建模的作用，还应该紧密围绕所建立的需求模型进行软件开发，这一部分本用例做得不是太好，并且由于最初需求建模中存在一些问题，导致最终的代码实现与需求模型有一些不匹配之处，在最终撰写本报告时又对需求模型进行了一些修改使之与软件代码相匹配，这样的情况在之后的建模和实现中应该避免出现。

3 设计建模

3.1 设计建模的目的

设计建模在需求工程的首次迭代得出结论时开始，其目的是利用一些引导的开发原则、概念和实践来实现高质量系统或产品开发，其目标是创建可以正确实现所有客户需求的、最适合项目利益相关者需要的软件模型。设计模型是需求模型的继续，是将需求变换为用于构建软件的“蓝图”。与需求模型注重描述所需数据、功能和行为不同的是，设计模型提供软件的体系结构、数据结构、接口和构件的细节。而软件设计在进行需求建模时就已开始，是建模活动的最后一个软件工程活动。软件设计在软件工程过程中处于技术核心，其应用与软件过程模型无关^[1]。

3.2 设计建模的方法

3.2.1 设计建模的核心概念

首先概括一些设计建模的核心概念。

设计模型不同于需求模型，是站在开发者角度而非用户角度看待问题，提出解决问题的方案。而设计模型包括有四种元素，这里讨论三种，分别为体系结构设计元素、接口设计元素和构件级设计元素。体系结构设计提供待构造系统的整体视图，描述软件构件的结构和组织、构建的特点以及构件之间的连接，注重早期的设计决策，包含多种不同的体系结构风格和模式，定义如何集成构件以构成系统的约束条件，使设计者能够理解系统整体特性的语义模型。

而接口设计元素包括用户界面 UI 设计和与外部已有系统或设备的接口设计。在本用例中即为 Web 页面设计与与数据库的接口设计。在这个阶段，全部的数据和软件的结构都已经建立起来，因此需要把设计模型转化为运行软件。接口提供关于通信和协作的重要信息

构件级设计过程包含一系列活动，这些活动逐渐降低描述软件的抽象层次，最终在接近于代码的抽象层次上描述软件，采用了可复用的软件构件和设计模式，细化时有包括开闭原则、依赖倒置原则以及耦合性和内聚性等设计原则和概念。这些设计原则和概念在本项目的设计建模中也有所体现。构件级设计涉及的概念有构件、接口、依赖于继承。其中，构件指的是已经被确定为体系结构模型一部分的，建立了命名约定并需要进行进一步的细化和精化的设计类；而接口提供关于通信和协作的重要信息。在将分析类中的边界类转化为设计模型中的构件时，需要给构件添上接口；而构件之间的依赖关系通过接口来表示，遵照 OCP 的思想，使得系统更加易于维护。

3.2.2 设计建模的方法

接下来对设计建模的基本方法和关键方法进行概括。

考虑问题的模块化解决方案时，要运用抽象的方法，给出不同层次的抽象级，主要分为过程抽象和数据抽象。过程抽象是具有明确有限功能的指令序列，而数据抽象是描述数据对象的冠名数据集合，而过程抽象会利用数据抽象属性中所包含的信息。

关注点分离是一个设计概念，其表明任何复杂问题如果被分解为可以独立解决和优化的若干块，该复杂问题就更容易解决。关注点是指一个特征或行为，被指定为软件需求模型

的一个部分。而模块化是关注点分离最常见的表现，即将软件划分为独立命名可处理的构件。模块数量应该控制在一个适当的范围内，此范围称为最小成本区域。模块数量小于该区域则会导致工作难度较大，而超出该区域则会导致总体软件成本较高。

信息隐蔽意味着通过定义一系列独立的模块可以得到有效的模块化，独立模块相互之间只交流实现软件功能所必需的信息，以此加强对模块内过程细节的访问约束和对模块所使用的局部数据结构的访问约束。

对应于分析类，在设计建模中会运用到设计类，这些设计类会促成类的实现，并实现支持业务解决方案的软件基础设施。设计类应该完整地封装所有可以合理预见到的存在于类中的属性和方法，而和一个设计类相关的方法应该关注于实现类的某一个服务，一旦服务实现，就不应该再提供完成同一件事情的另一种方法。同时设计类还应该具有高内聚性和低耦合性，即职责集合应该小而集中，设计类之间的相互协作应保持在一个可接受的最小范围内。

3.3 设计建模的项目实践

3.3.1 架构设计

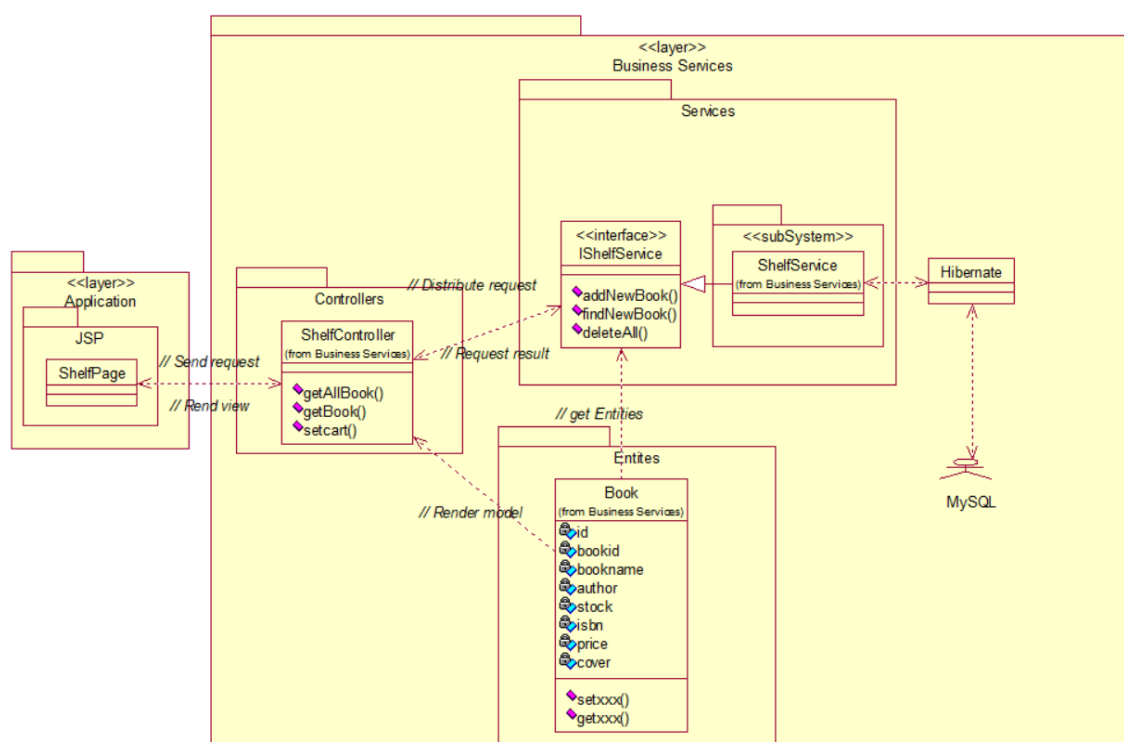


图 13 UML 包图描述的逻辑架构

上图为 UML 描述的分层的逻辑架构，采用了 MVC 分层架构（模型—视图—控制器分层）为应用层和服务层两层。其中，展现层，即左边的应用层展示了包 JavaScriptPage，其中包含本用例涉及的前端页面；而应用程序逻辑层，即右边的服务层中包括了控制器 Controller、数据模型实体类 Entities、业务逻辑 Services 三个包和数据持久层 hibernate 框架以及 MySQL 数据库管理系统，核心是控制器，模型关注数据处理，视图关注数据显示和报表处理，控制器负责协调模型和视图。

上述架构中，展现层的 ShelfPage 向 ShelfController 发送请求，而 ShelfController 分发请求到 ShelfService; ShelfService 从 Book 处得到实体并响应请求返回结果到 ShelfController，其中 IShelfService 为构件 Service 的接口，负责提供关于通信和写作的重要信息。Book 也对模型进行渲染，而 ShelfController 对视图进行渲染。在与外部系统（MySQL）相连部分，Hibernate 对 JDBC 进行一次轻量级封装，自动生成 SQL 语句将文本类书籍信息存储到 MySQL 数据库管理系统。

所以本项目分为了三层架构，即包含 JSP 的表示层（Web）、业务逻辑层（Service）和数据持久层（DAO）。这一三层架构在随项目答辩文档提交的代码里容易看出，表示层位于 react-front/src/ShelfPage.js，业务逻辑层位于 backend/src/main/java/com/example/demo/Service/ShelfService，而数据持久层利用了 hibernate 框架，相关配置文件位于 backend/src/main/resources/hibernate.cfg.xml。

3.3.2 构件设计

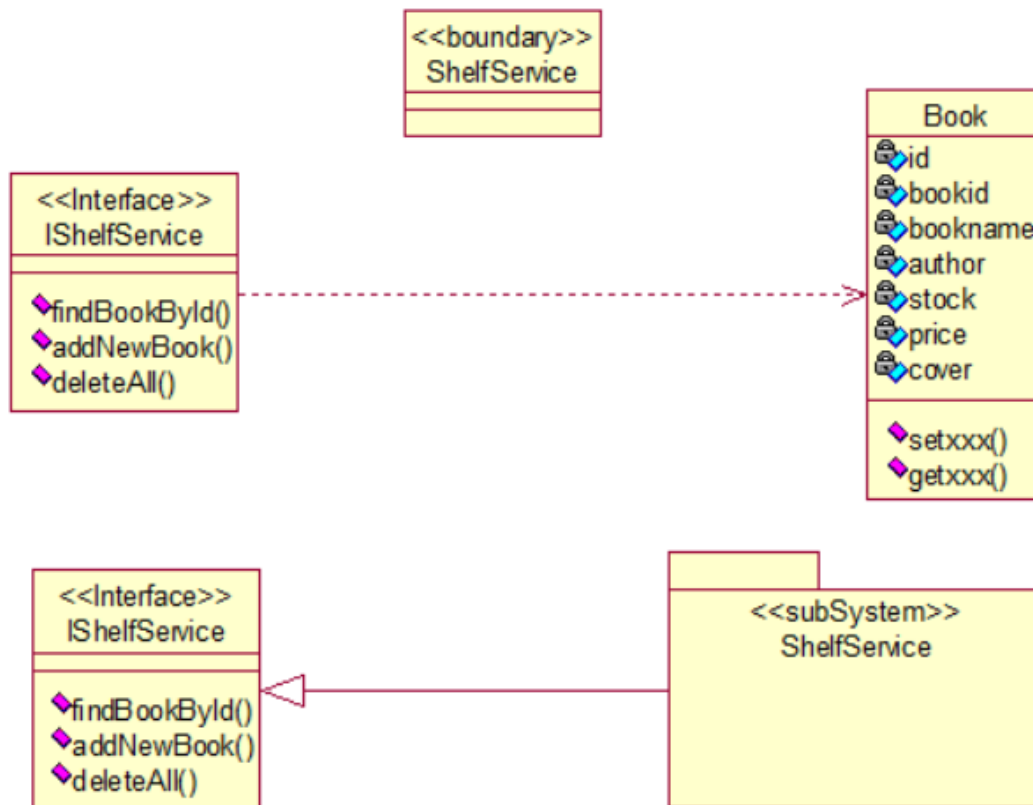


图 14 构件/子系统的接口定义

构件和构件接口定义如上图所示。其中最上侧为分析类，其下两图为定义接口。其中，shelfService 为构件，负责与外部系统，即书籍数据库交互；而该构件依赖于对应的 ShelfService 接口，该接口与数据模型实体类 Book 关联。

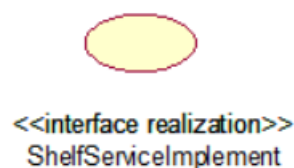


图 15 构件/子系统的接口实现

而接口的实现如上图所示。其下为子系统 ShelfService 的功能实现：

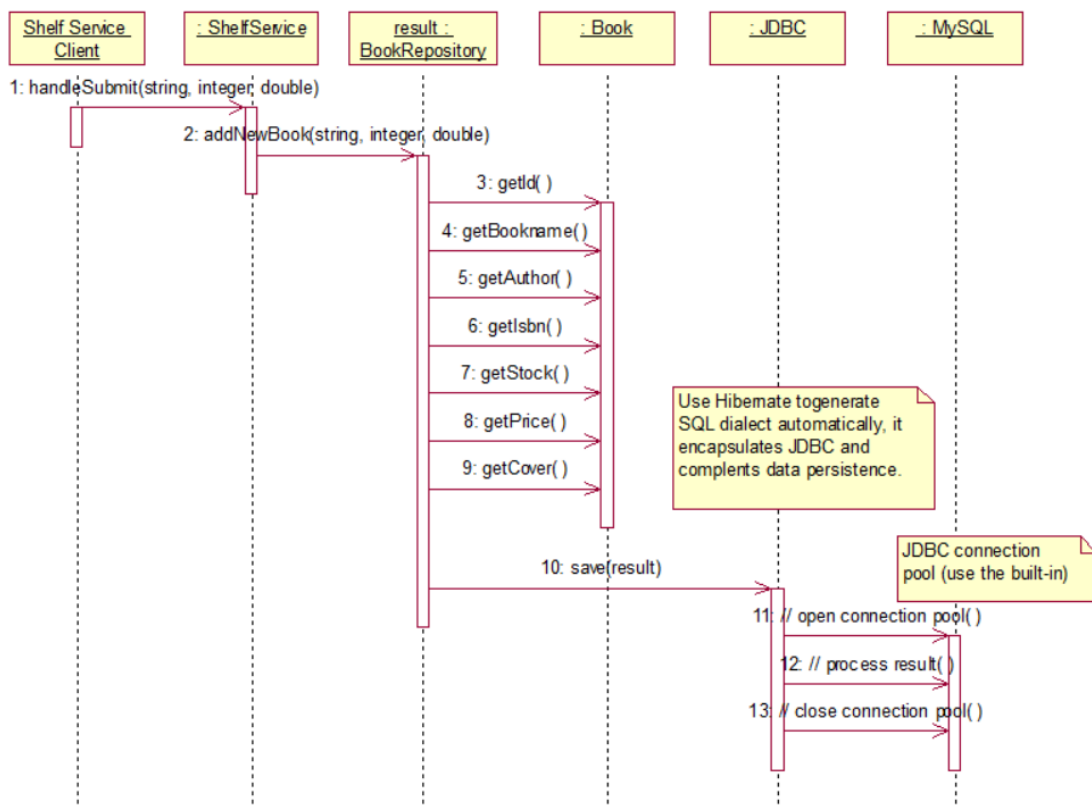


图 16 描述接口实现动态行为的 UML 顺序图

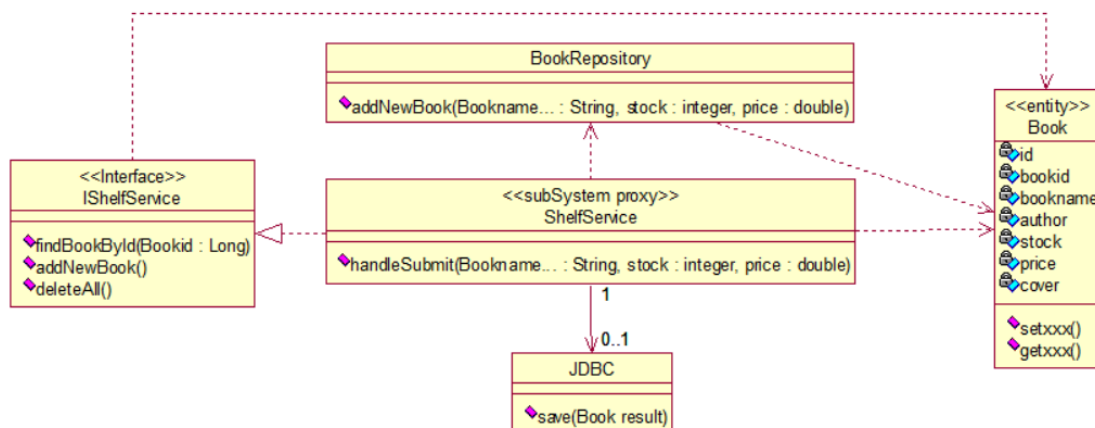


图 17 描述接口实现静态结构的类图

而上图所示即为所对应的描述接口实现动态行为的 UML 顺序图和描述其静态结构的类图。从 Shelf Service Client 开始描述动态行为，子系统 ShelfService 收到上架操作的信息，发出创建新书籍实体的函数，而后获取书籍的属性后将其整合为 result 存储至数据库。其中，JDBC 为用于执行 SQL 语句的 Java API，而数据持久层框架 hibernate 对其进行了轻量级封装。而类图中从静态结构方面对其进行了描述。

而下图为描述子系统和其它模块之间依赖关系的包图。子系统 ShelfService 与外部系统中的 External Interface 和应用实体中的 Book 相关联。

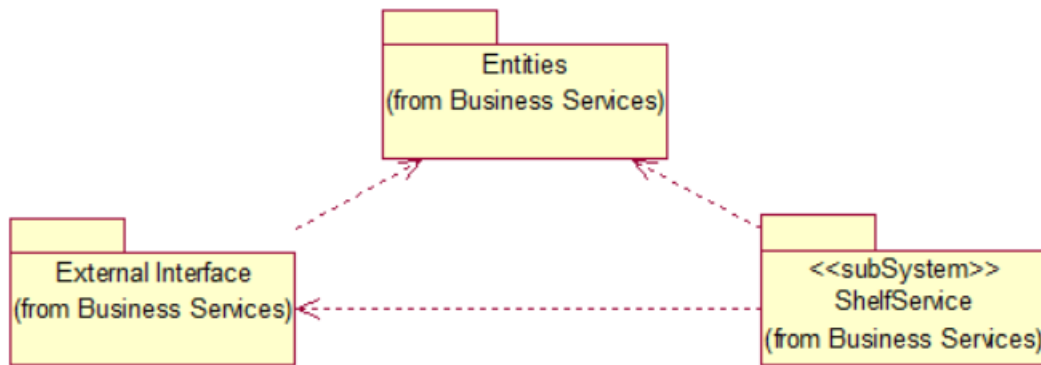


图 18 描述子系统依赖关系的包图

而在 GoF 设计模式的应用方面，创建型选择了抽象工厂。所谓抽象工厂模式，即创建多个工厂类，后续再增加新的功能只需要增加新的工厂类即可，比起普通的工厂方法，遵循了闭包原则，拓展性更好。

项目中也用到了一些设计原则，如利用抽象工厂遵循了开闭原则，在程序拓展时不需要修改原有代码，即热插拔。这一效果使用了接口和抽象类，实现这一原则的关键步骤是抽象化。除此之外还用到了里氏替换原则，比如在后面的拓展功能，即浏览书籍信息部分，复用了书籍增删/管理的功能，而只有当衍生类可以替换掉基类，软件单位的功能不受到影响时，基类才能真正被复用。

3.4 设计建模的实践心得

设计建模过程中遇到的较大的问题有以下几点：

- 1) 关于逻辑架构：初次用包图描述逻辑架构时，将层内的包也加上了<<layer>>的标签。这是对层的理解不到位造成的。每个层次抽象程度应相同，每层各自完成操作；
- 2) 关于接口实现：初次建模时没有画出接口 IShelfService 的功能实现，接口实现不完整。这也是对接口实现的理解不到位造成的；
- 3) 最初的接口命名不符合规范导致代码与模型不匹配，且由于名称原因混淆了一些类，导致最初建立的模型较为混乱。之后修改了接口名称，按照命名规范将所有接口名称进行了更新；
- 4) 最初建立的设计模型中运用了外观设计模式，而之后在编写代码时并未用上。而本构件涉及的仅有一个接口，因此没有进行封装；
- 5) 前已述及，本系统使用了 Java 的 API，即 JDBC。而频繁打开和关闭数据库会无限制地占用系统资源，故采用了数据库连接池技术，从而提高系统的性能和稳定性。除此之外，本系统还使用了 Hibernate 数据持久层框架，对 SQL 和 JDBC 的涉及较少。

概括地说，设计建模部分由于建模时已经对建模过程有一定程度的了解，同时也熟悉了建模工具，软件项目开发也有相当程度的进展，所以比起需求建模要完成得更顺利。除此之外，设计建模的确对软件项目的开发有相当程度的帮助，诸如 3.3.2 节最后几段提到的 GoF 设计模式，对项目的结构以及后续的编程都有很大的引导作用。

4 质量管理

4.1 质量管理的目的

要讨论软件质量管理的目的，首先要了解软件质量的定义。软件质量的定义应该强调有效的软件过程、有用的软件产品和能提供的明显价值。而软件质量管理的目的在于，如果团队在所有软件工程中都强调质量，就可以减少很多不必要的工作量，其结果是可以降低成本并缩短项目完成时间。

4.2 质量管理的方法

4.2.1 质量管理核心概念

首先简要地叙述一些有关软件质量管理的核心概念。

软件质量可以定义为“在一定程度上应用有效的软件过程，创造有用的产品，为生产者和使用者提供明显的价值”^[1]。前已述及，这一定义强调了有效的软件过程、有用的软件产品和能提供的明显价值三个方面。而质量维度的概念包括性能质量、特性质量、可靠性、符合性、耐久性、适用性、审美和感知。这些质量维度在软件质量考虑时可以使用，但并非全部必须。这些评判标准大多数只能主观考虑，因此需要有一系列较为客观的质量因素，即质量因素。

质量因素侧重于产品运行、产品修改和产品转移三个方面。细化来讲，这三个方面又包括产品运行中的正确性、可靠性、易用性、完整性和效率，产品修改中的可维护性、灵活性和易测试性，产品转移中的可移植性、可复用性和互操作性。这些质量因素的直接测度是困难的，但间接度量这些因素，使用这些因素评估应用系统的质量可以真实地反应软件质量。

而无论选择什么方法，质量都是有成本的，而缺乏质量也是有成本的。这里引出了质量成本的概念。质量成本包括追求质量过程中或履行质量有关过程中引起的费用，也包括质量不佳引起的下游费用等所有费用，可以区分为预防成本、评估成本和失效成本。

4.2.2 质量管理方法

软件质量管理需要的四大管理和实践活动是软件工程方法、项目管理技术、质量控制活动和软件质量保证。软件工程方法即需要创造一系列符合需求的设计，这也是之前的两个部分，即需求建模和设计建模一直讨论的问题。如果合理完整地理解了需求，进行了综合性设计，为构造活动打下了基础，那么就更容易创造出高质量的软件，这也是需求建模和设计建模的意义。

而合理的项目管理也能够对软件质量起到积极的影响。这意味着需要合理安排项目进度计划，明确进度依赖关系并严格执行，并进行合适的风险规划。

而要保证软件质量，还需要一套关于质量控制的软件工程活动，在测试开始前评审模型，检查代码，对每一部分进行分块的测试以保证其达到了预期的功能。

而除上述四大管理和实践活动外，软件的质量管理方法还涉及软件的评审和软件的质量保证。评审的目的在于找出错误和发现可能带来负面影响的问题，而软件质量保证包括对方法和工具有效应用的规程，对技术评审和软件测试等质量控制活动的监督，变更管理规程，保证符合标准的规程以及测量和报告机制。而具体的软件测试策略不再在此展开叙

述。

4.3 质量管理的项目实践

本项目的质量管理实践包含下列几个方面。首先，项目是在需求建模和设计建模的基础上进行的，需求模型的正确性、完整性和一致性以及设计质量的属性为构造活动打下了基础，对软件的质量有相当程度的提升。而本组的开发计划也基本遵循初期设定的甘特图，保证了按照进度依赖关系进行开发。

质量评审的进度安排也纳入到了软件增量的项目计划之中，而在用例完成后也进行了单元测试。本用例的单元测试模块如下：

表 4 单元测试模块：商品上架

开发项目名称		Web 二手书交易平台		项目第一负责人		杨文曦		
单元名称		商品上架	责任人	杨文曦	开发周期	2019/5/11-2019/6/1		
代码测试检查								
代码测试内容		测试结果				备注		
路径测试		所有可执行语句均执行				无		
声明测试		声明有效				无		
循环测试		循环执行正常				无		
边界测试		无法正常处理极端情况				未考虑数据结构边界		
接口测试		数据交换正常				无		
界面测试		易理解，风格一致，页面元素可用				未进行美化		
代码走查		通过				和组员进行讨论		
功能测试								
序号	功能名称	操作方法			结果		建议	
1	商品上架	通过菜单栏进入上架页面,输入测试信息,提交,在数据库中查看提交的测试信息			运行基本正常,但未处理异常输入		可在后续修改时增添对异常输入进行处理的功能	
测试结论		基本实现预期功能			测试日期		2019 年 6 月 1 日	

其中代码测试检查部分在边界测试中出现了问题。在设计中，本用例被设计为可以储存 1024 组数据，但并未考虑如果超出如何处理，未考虑边界；而在功能测试部分，原本每种属性都规定了数据类型（例如书名 `bookname` 应为 `String` 类型，而价格 `price` 应为 `double` 类型），但在上架页面中输入时并未对输入数据类型进行检测。除此之外本用例基本实现了预期功能。除此之外，小组内规定了代码的统一风格和变量名的命名规范，这样代码更加易读且便于更改，更易于维护。

在系统测试方面，本组进行了恢复测试，即在出现错误后人工干预进行初始化，手动刷新页面，平均恢复时间在可接受的范围之内。在性能测试方面，本系统响应特征时间应在 0.1s 以内，除此之外并未考虑足够终端数支持、足够并行操作用户数支持和动态数值支持。除此之外，软件只在 Windows10 系统中运行过（包括丁维德的 Windows 虚拟机），并未进行部署测试。

除此之外，本组的项目版本控制还使用了 GitHub。最终的项目代码以及每一次修改，每一次答辩的材料和项目文档各部分的各个版本，以及每一次组会的日志都在 GitHub 上有所记录。

4.4 质量管理的实践心得

本组个人的软件质量管理中遇到过如下问题：

1) 统计软件质量保证：如果运用统计软件质量保证方法，可以发现本组质量管理部分问题较为突出的部分是设计模型，其中主要问题是构件接口不一致和设计逻辑出现错误。在每个人的报告中对这一部分都已经进行了修改；

2) 风险规划：项目预先进行了风险规划，并且风险规划的确起到了作用，例如 Karen 的缺席虽然影响到了他本人的用例实现，但并没有影响其他组员用例的实现；

3) 软件质量与成本的平衡：本组的项目进度设计实际时间较为紧迫，因此需要承认在很多细节上的软件质量并没有得到充分保证。并且，本组的正式技术评审并不到位，组员基本没有进行事先准备，以至于正式技术评审的效果并不良好，在最终代码提交后仍然发现了软件的一些缺陷。在最后几次小组会议时，会议的主要内容都偏向于设计模型的修改和代码基本功能的实现，而代码基本是通过走查的形式互相讨论，且并没有按照严格的 FTR 方式做出统一的评审决定。在各组员完成各自用例，在组长处整合完成后，也没有尽早通过会议的形式让全体组员对整个项目的各部分代码进行审查，虽然没有影响到基础功能的实现，但的确对本组项目的质量管理产生了负面的影响；

4) 个人模块质量管理：在个人模块质量管理方面，对本人负责的模块进行了单元测试，测试结果即前述表 4 所示。由于本模块只强调一个功能，因此比较容易预见和发现错误。同大多数软件一样，最主要的错误行为出现在边界测试部分，而这一错误是有考虑到但由于开发周期等因素而并没有进行限制的；

5) GitHub 的使用：虽然很早就拥有了自己的 GitHub 账号并上传了一些代码，但本次项目开发是第一次利用 GitHub 进行项目版本控制。初次运用 GitHub 进行项目管理还是遇到了不少问题，但总体来说达到了项目版本控制的目标。

综上所述，本组软件过程中的质量管理得到了落实，但在后续的正式技术评审以及单元测试后对出现问题的解决方面还有欠缺。总的说来，在软件质量管理部分了解并实践了一些软件质量管理的方法，基本达成了本部分的目标。

5 软件过程与项目管理

5.1 软件过程与项目管理的目的

5.1.1 软件过程的目的

软件过程指软件开发过程中遵循的一系列可预测的、需要完成的工作活动、动作和任务的集合。遵循软件过程进行产品的开发和系统的构建有助于及时交付高质量的产品，可以做到强调灵活性、可延展性和开发速度。另一方面，合适的良好的软件过程还可以鼓励组员和利益相关者之间的相互沟通和协作。

5.1.2 项目管理的目的

而软件项目的管理涉及对人员、过程和所发生事件的策划和监控，需要制定和遵循一定的项目计划，该计划应当定义将要进行的过程和任务，安排工作人员，确定评估分享，控制变更和评价质量的机制。项目管理的目的是协调业务和团队成员之间的关系，从而更加有条理地，高效地从宏观上统筹项目的开发。

5.2 软件过程与项目管理的方法

5.2.1 软件过程

通用过程框架定义了五种框架活动，包括沟通、策划、建模、构建以及部署，这些框架活动有一系列软件工程动作构成，每个软件工程动作由任务集合来定义，明确工作任务、工作产品、需要的质量保证点以及表明过程状态的标志。其中，任务集定义了为达到一个软件工程动作的目标所需要完成的工作。除此之外，而项目跟踪控制、风险管理、质量保证、配置管理、技术评审以及其他活动贯穿软件始终。

而过程流也是软件过程的重要方面之一，其描述了在执行顺序和执行时间上如何组织框架中的活动、动作和任务。过程流分为线性过程流、迭代过程流、演化过程流和并行过程流。

过程模式描述了软件工程工作中遇到的过程相关的问题、明确了问题环境并给出了针对该问题的可证明的解决方案，即提供了统一描述问题解决方法的模板。

而惯用的过程模型有瀑布模型、增量过程模型、演化过程模型和协同模型。本组采用的是演化过程模型。在开发过程中，产品需求经常会发生变化，故采用了这一专门应对不断演变的软件产品的过程模型。这一过程还涉及原型开发、螺旋模型等概念。

原型开发范型可以在需求较为模糊时帮助软件开发人员和利益相关者更好地理解究竟需要做什么。组员和利益相关者在项目开始时进行沟通，定义软件的整体目标，明确已知的需求，并大致勾勒出以后会进一步定义的东西，然后迅速策划一个原型开发迭代并进行建模。在原型不断调整以满足各种利益相关者需求的过程中，采用迭代技术，同时也使开发者逐步清楚用户的需求。概括地说，演化过程模型即先明确核心需求，快速设计出原型，并根据需求迭代丰富功能，故会经历多次迭代，可以做到强调灵活性、可延展性和开发速度。

而敏捷方法是为克服传统软件工程中认识和实践的弱点而形成的。普遍存在的变化是敏捷的基本动力，但敏捷并不仅仅是有效地响应变化，还能鼓励团队内的沟通和协作。

本组在敏捷开发方面选择了极限编程，这是敏捷软件开发中使用最为广泛的一个方法。

极限编程使用面向对象方法作为推荐的开发范型，包含了策划、设计、编码和测试四个框架活动的规划和实践。其中，策划是需求获取活动，该活动使团队技术成员理解软件的背景以及充分感受要求的输出、主要特征和主要功能；而在设计方面，极限编程鼓励简洁而非复杂的表述，不鼓励额外功能性的设计。而设计可以在编码开始前后同时进行，重构则意味着设计随着系统的构建而连续进行，其目的是控制由于提出“可以根本改进设计”的小设计修改而造成的代码改动；完成编码后，所有人的代码会在通过单元测试后集成起来。这种集成作为团队的日常工作实施。而单元测试在编码开始前就已经建立，而所建立的单元测试应当使用一个可以自动实施的框架。

5.2.2 项目管理

而在项目管理方面，首先要明确项目管理的范围。有效的项目管理集中于人员、产品、过程和项目四个方面，其顺序并非任意。项目初期人员间的深入交流可以避免为错误的问题构建模型；而注意过程可以避免技术方法和工具失效，而在建立可靠的项目计划后再开始工作有利于项目有条不紊地进行。

在前面的各部分中不止一次提到了“利益相关者”这一概念。参与软件过程的利益相关者可以分为高级管理者、项目技术管理者、开发人员、客户和最终用户，而每个项目都有上述人员的参与。组织实现最大限度地发挥每个人的技术和能力，可以获得高效率。这也是团队负责人的任务。

而软件项目管理的第一项活动就是确定软件范围。软件范围通过项目环境、信息目标和功能与性能来定义。软件项目范围在管理层和技术层都必须是无歧义、可理解的。而软件需求分析的核心活动是问题分解。在确定软件范围的过程中，要分解其中的两个主要方面，即必须交付的功能和内容，以及所使用的过程，而不需要完全分解问题。

而过程部分与 5.2.1 中所叙述的软件过程有很高的重合度，故不再在此赘述。

在所有软件项目中，最关键的因素是人员。可以通过多种协调和沟通技术来支持团队的工作。一般而言，技术评审和非正式的人与人交流对开发者最具有价值。

项目计划活动是软件项目管理的重要组成部分，而进度安排是计划活动的首要任务。进度安排始于过程分解。根据项目特性，为将要完成的工作选择适当的任务集。以进度为指导，项目管理者可以跟踪和控制软件工程过程中的每一个步骤。

5.3 软件过程与项目管理的项目实践

由于项目开始初期的需求还较为模糊，故本组在软件过程方面采用了演化过程模型，先明确了核心需求，以帮助所有成员更好地理解需要做的任务。而在后续每周三的例行组会上，组内成员对软件的整体目标进行了讨论，明确了已知的需求，并在需求模型建立的过程中就开始了原型开发迭代。而在开题答辩、中期答辩和期终答辩之间，本组对需求进行了足够的细化，也在开发过程中对原有需求进行了一些更改，并在其间完成了第一次迭代，并在后续编码中根据需求丰富了软件功能。

而在敏捷方法方面，本组选择了极限编程，即遵循策划、设计、编码、测试的路线进行了项目的开发和管理，最终基本完成了预期任务。

而本组在项目开始初期就进行了项目计划活动，安排了项目进度计划。本组的项目进

度计划甘特图如下：

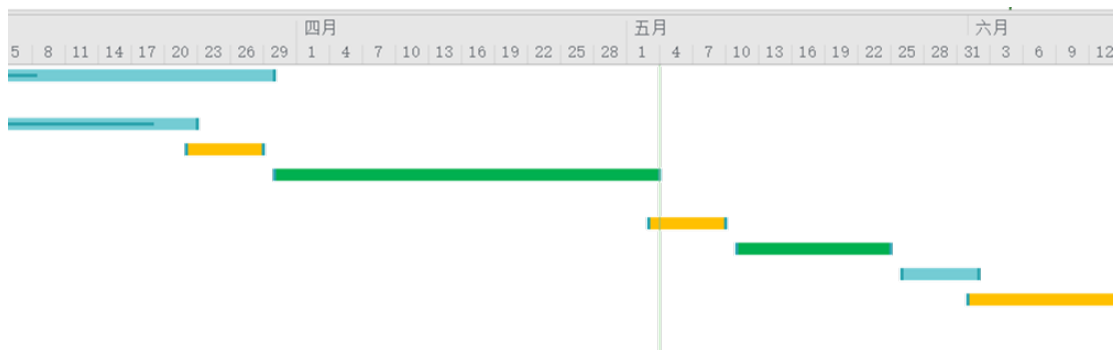


图 19 项目进度计划甘特图

对应的项目进度表如下：

表 5 项目进度计划表

任务名称	工期	开始时间	完成时间
学习相关编程语言	21 个工作日	2019/03/01	2019/03/29
需求调研	16 个工作日	2019/03/01	2019/03/22
准备开题答辩	5 个工作日	2019/03/22	2019/03/28
开展项目并讨论	26 个工作日	2019/03/30	2019/05/03
准备期中答辩	5 个工作日	2019/05/03	2019/05/09
完成各自剩余内容	11 个工作日	2019/05/11	2019/05/24
测试完善	7 个工作日	2019/05/26	2019/06/01
准备期末答辩，结题	11 个工作日	2019/06/01	2019/06/14

而本组的项目基本遵循了以上的项目进度计划。

在团队协作方面，本组固定于每周三开展组会，交流项目开发进度和其中遇到的困难。本学期共开展了 14 次组会，其中新图书馆小组学习室 10 次，东上院自习室 2 次，东中院 1 号楼讨论区 1 次，第二餐饮大楼西餐厅 1 次。各次组会的情况都记录在日志中，可以在 GitHub 上查看。

5.4 软件过程与项目管理的实践心得

作为团队的组长，在软件过程和项目管理部分还是有很多感受：

1) 直观的感受：首先，通过本次项目的开发和管理，我的确感受到了软件开发的难度和利用工程化方法进行软件项目开发的优点。很显然这样的工程化方法更为正规，也更加适合团队项目，尤其是大型团队项目的开发。本次项目开发中运用到的方法在今后的各种团队项目中都可以运用；

2) 关于模型和方法的选择：实践情况证明本组的软件过程模型选择演化过程模型、以及敏捷开发选择极限编程都是较为合适的，而项目基本按照甘特图进行也证明了本组和项目进度计划是基本合理的。

6 总结与建议

6.1 工作总结

首先要确认的是，本次软件项目开发基本实现了预期功能，而在需求和设计模型建模以及质量管理、软件过程与项目管理方面也基本达到了预期的目标，虽然最终开发出的软件较为简陋且未完善所有功能，但基本可以认为本次项目开发是成功的。

在项目的开发过程中，除了对本课程的内容，包括需求和设计建模、质量管理方法、软件模型选择和项目管理方法等方面的学习外，还对 Java/JavaScript 语言进行了学习，对 Java Web 应用的框架和具体实现方法进行了了解和学习，同时还对数据库相关的知识有了进一步的理解。从这一方面来说，我对自己的学习情况还是比较满意的。

而作为团队负责人，有点像 Pressman 在书中所引用的 Edgemon 的话：“很不幸却经常是这样，人们似乎碰巧落在项目经理的位置上，也就意外地成为项目经理。”当然，负责整个团队和整体布局项目的开发对于个人是非常好的锻炼机会，而我也确实在这个过程中学到了项目开发之外的很多项目管理方面的东西，也借这一机会和原本并不熟悉的几个同学成为了朋友。虽然作为一名团队负责人，对于整个项目的管理和开发的作用不能和同班的乔田伟、王珏一样出色，但能够基本完成项目管理和开发的任务也就算达到了目标，而且相比所开发的软件项目，更重要的是这个过程中收获的知识和经验。

6.2 建议

对于本项目的建议有：

1) 性能可优化：目前代码的运行速度，尤其是初始化的速度很慢。虽然基本实现了预期功能，但客户不会喜欢性能如此差的软件。后续如果有可能继续此项目，可以考虑优化本项目的性能；

2) 运行环境可简化：本项目除前端运用了 React 架构之外，还用到了诸如 MySQL 和拟运用的 Mongo DB 等外部系统，环境搭建较为复杂（最初组内成员的环境搭建就花了两次组会的时间），可以考虑进行简化；

3) GitHub 的运用：本项目的版本控制主要还是依靠小组会议，对 GitHub 的使用频率并不高。GitHub 完全可以发挥更大的作用；

4) 可以进一步提高封装性和编写 css 文件美化前端页面；

5) 代码注释较少，可读性不强，可以适量增添注释。

而对于本课程的建议有：

1) 需求建模的周期较长，在学生能力可及的范围内可以适当缩短需求建模时间，从而为后续的设计建模和测试等留出更多的时间；

2) 大家平时对教材的阅读较少，可以将平日的一些作业改为阅读教材并完成章节后的习题和思考题。

参考文献

- [1] Roger S. Pressman, Software engineering: A practitioner's approach (7th Edition) [M], McGraw-Hill Companies, Inc., 2011
- [2] netoxi, 设计模式笔记: GoF 设计模式汇总[J/OL], 博客园, <https://www.cnblogs.com/netoxi/p/7259863.html>
- [3] Hibernate, hibernate-orm[Z], GitHub, <https://github.com/hibernate/hibernate-orm>

致谢

首先要感谢饶老师这一学期以来的精彩授课和对问题的耐心细致的解答。然后希望老师多注意身体, 很多次都看见老师午餐只吃几个包子……这样对身体不好。

然后要感谢助教孔璐和为我们讲解 Git/GitHub/单元测试的分布式计算实验室的黄汇学长, 你们的工作对于我们的课程学习以及项目开发都有很大的帮助。

接下来要感谢所有组员本学期的付出。我自知我这个团队负责人在很多方面做得都不怎么到位, 非常感谢组员的配合以及大家对项目开发的贡献。杜培栋上课听得很认真, 在设计模型部分为组员们提供了很大的帮助; 丁维德作为交换生也对本项目非常认真, 对项目的很多代码实现都进行了讨论。而最终项目开发能够顺利完成是每个人的功劳。

除此之外, 还要感谢王珏同学在本组设计模型建模部分提供的帮助。除了设计模型刚开始时的帮助外, 本组最终的设计模型也反复参考了王珏的模型。虽然答辩时大家的设计模型都存在一些问题, 但相信最终的报告中大家都完成了正确的设计模型。另外还要感谢一些软件学院的同学在本项目的环境搭建和编程过程中提供的帮助, 这些帮助让本组的软件开发少走了很多弯路。

感谢对我个人和整个团队的学习和项目开发提供帮助的所有人!