基于企业级系统升级与重构的实训课设计

白玉琪
河北石油职业技术大学
cdpc_byq@cdpc.edu.cn

骆磊
河北石油职业技术大学
cdpc_ll@cdpc.edu.cn

摘要 Java 企业级开发课的岗前实训课程设计中普遍性的难题是如何设计在系统复杂度、开发流程与活动各个方面接近真实职业场景的课程结构。本文提出一种岗前实训课程设计方法，它以具有真实复杂度的成熟的企业级应用系统为起点，通过一次完整的升级周期，历经功能业务与技术的重构、设计、编码、测试、上线活动，充分达成岗前实训课程的目标。

关键词 岗前实训 系统复杂度 领域驱动 重构

计算机软件本科课程中，《Java 企业级应用开发》不同于一般语言类课程，是一门理论性与技能性兼备的高度综合的课程，直接面向企业的技术人才需求；它首先要使学生在开发语言与技术的掌握上达到能够开发真实业务系统的要求，同时还要对设计模式、系统架构、最佳实践、领域模型等方法论问题有基本认知，并且对软件系统生命周期中的活动与环节有清晰概念。而与这门课程相对应的岗前实训课则是要对以上要求进行全面落实、强化和检验，为入职后的开发实战做好准备，如何设计这门实训课，是教学中极为重要的一环。本文提出的实训课设计能够在最大程度上达成以上教学目标。

1. 当前实训课程设计存在的问题

目前这类实训课的基本方式是：选取某个行业的一种业务，从头开发一个应用系统，在开发过程中通过讲解和学生参与完成一些简单功能的开发。

这种课程设计存在以下问题：

1. 以实训课的课时安排，所完成的还是一个玩具系统，与现实中真正的企业级系统相去甚远，学生不能获得企业级开发的现实感，失去了实训之实。

2. 由于是从头开发，即使是一个玩具系统，开发量仍然不易控制，这样就使得单纯的编码活动几乎成为全部内容，而失去了软件生命周期的全貌。

2. 基于企业级系统升级与重构的实训课设计

本实训课程的思路是：准备一个已经完成的、业务功能丰满完备的开源电商的系统，提出新的功能模块扩展需求，在实训课上完成一次版本升级，包括

1. 分析完整业务模型和现有代码
2. 分析扩展模块需求，完成扩展模块设计。
3. 如有必要，对现有代码重构。
4. 开发新模块，测试，升级部署上线

这个设计有如下几个突出优点：

1. 面向一个真实项目，有足够复杂度，突出实训之“实”
2. 一个模块升级从开发量上合适。
3. 一个完整的需求分析，设计，编码周期。
4. 突出重构这一核心开发活动。
5. 新员工在大部分情况下，入职之后的工作都是扩展开发现有系统

对这个设计我们从以下几点做进一步的分析

2.1 真实的系统复杂度是重要的

软件系统，尤其是企业级应用系统，都是在支撑一种行业的业务，而业务的复杂性是系统复杂性的根源，它是模式、架构、方法论处理的核心问题，几十年企业级系统开发实践与理论都是围绕这一核心展开。Brooks 的《no silver bullet》[1]早在 70 年代就断言，软件系统无论技术如何进步，解决的都是表现的复杂性，而不是实质的复杂性，实质复杂性来自于行业、业务等领域问题，这种领域行业的复杂性会一直存在，此后几十年的发展史都在证明这一断言。《domain driven design》[2]在软件方法论历史上具有里程碑意义，提出了业务领域模型是软件构建活动的核心驱动力，一直在应用类系统开发中有持续的重大影响，今年有关微服务的探讨中，领域驱动理论中的 bounded context[3-5]仍然是核心议题。设计模式、架构方法、最佳实践从根本上都是在应对和处理这种业务复杂性问题，对于一个脱离真实复杂业务的玩具系统，设计模式、架构方法、最佳实践等方法论工具就失去了用武之地，而作为实训项目也没有达成教学目标的可能。

软件系统在其全部生命周期会经历一系列版本升级，而每次升级都是一次需求收集与分析、设计、开发、测试的完整周期。这个结构为我们的实训课设计提供了一个方式：不是从新建应用系统开始，而是从应用系统的成熟期切入，截取一个完整的升级周期作为实训内容，这样一种方式使得学生马上面向一个具有真实业务复杂度的系统，为实训之“实”提供了基本条件。

2.2 重构是最核心的开发活动

开发方法论的理论和实践中，重构（Refactoring）[6]理论有重大影响，其核心思想是，由于应用系统中高度的业务复杂性和人的理性设计能力的局限性，开发活动的重点不是事前设计，而是事后的代码结构的重构，业务模型重构、系统架构重构、代码级重构（战略性重构与战术性重构）遍布系统生命周期的全部，每次升级在功能性开发活动之外，都有可能、有必要伴随重构活动，重构活动是系统可控性、可扩展性等所有重要品质的根本保证，重构能力是开发人员的核心能力。

因为重构开发活动的重要性，实训课设计必须突出这一安排；但是对于缺乏复杂度的系统，重构的必要性无法凸显，即使课程设计中勉强安排，会出现为重构而重构的情况，而本实训课的设计安排则能够充分展开重构教学和训练。

2.3 契合学生职业生涯的场景

绝大部分情况下，新员工入职软件开发马上面临的任务，不是参与一个新启动项目的从头开发，而是参与现有系统的升级开发，主管给你代码库，要求阅读代码，熟悉业务和所用技术，然后参与升级功能分析、技术设计、现有代码重构以及新功能开发、测试、升级上线等工作，而我们实训课的课程设计高度契合这一入职场景，对于“岗前实训”无疑是相当合适。

2.4 其他课程设计特点

1. 充分模拟、复现软件周期的流程环节，业务部门、产品部门、开发部门、测试部门、运维部门、高层管理决策的职责与配合。
2. 在授课时对设计与编码中的方法论、设计模式、最佳实践充分讲解、探讨、实操，引导学生对理论课的内容加深理解，而不是一味埋头编码，使得学生入职后参与团队问题讨论时头脑有所准备。

3. 课程概要方案设计

本实训课 24 课时，本文每 4 课时一小节予以介绍

3.1 实训项目讲解(4 学时)

（1）电商平台业务和行业背景、需求与功能分析(40 分钟)

1. 讲解电商业务知识，电商类平台的总体功能。
2. 关于单商户系统和多商户卖场在技术上的不同要求

（2）技术选型与架构讲解(40 分钟)

有关技术选型问题：

1. 数据库选型，关系型还是文档型？
2. 持久层技术，JPA还是mybatis？
3. 有没有搜索功能？用的什么全文检索技术引擎？

（3）三．核心代码分析(80 分钟)

对核心模块、关键技术的代码做出全面深入的分析。这部分工作是升级开发的前提条件。阅读的过程中带着几个任务：

1. 通过代码反推业务模型，并从业务模型的抽象角度对平台的核心质量做出评估，包括模型的扩展能力，实体关系构建是否合理，组件服务的责任划分是否具备通用性。
2. 对平台技术构成全貌充分把握，包括是否用到异步消息？是否有批处理作业？有哪些部位用到第三方服务？支付支持了哪些支付方式?
3. 评估代码质量，识别有可能需要重构的架构失衡的部位，代码大量冗余的部位，对象责任分配错位的部位，等等

总之，阅读和分析代码是开发人员最重要的技能之一，本讲安排了较长的课时，充分展开讲解。

注意的要点：

1. 在支付系统方面，多商户系统的复杂性提高很多
2. 开发人员入职第一个工作就是阅读分析已有系统的代码
3. 识别设计缺陷和代码缺陷并迅速给出解决思路需要充分经验和高度技巧，这一点对

初学者不可能做到，要用启发式方法使学生有所思有所悟

3.2 升级需求分析与设计

（1）讲解平台生命周期管理、版本升级的一般流程(40 分钟)

从生命周期全局的角度来讲解和剖析一次版本升级的可能目标和任务构成。这里要充分结合理论课相关内容，归纳几个讲解重点：

1. 版本升级的构成问题：功能性的升级，技术性的升级
2. 功能性升级可能引发技术性升级
3. 模型的进一步扩展未必一定支持具体的功能性升级要求
4. 模型扩展优化、技术架构与代码质量优化未必仅仅支持本次功能升级，而是要为未来可能的功能升级做充分准备
5. 如果模型、架构、代码面临恶化失控，即使没有功能升级要求，安排重构升级也成为必要

（2）版本升级需求分析(40 分钟)

1. 首先对本次升级的功能内容做出具体分析
2. 考察业务实体模型要做哪些扩展，现有库结构和实体结构是否支持这种扩展，是否需要重构
3. 考察现有架构和代码有哪些问题有必要在本次升级中解决

以上分析建立在前面课时充分的代码分析基础上

（3）现有业务模型的扩展、架构和代码的缺陷，分析重构问题，提出重构内容(40 分钟)

现有系统是否需要重构，是否安排在本次升级需要综合考虑，这里需要详细讲解如何考虑权衡折衷各种因素，最后确定本次重构内容：日志和异常处理。

（4）升级技术选型和技术设计(40 分钟)

在分析了升级需求，确定升级内容之后，分析以下问题：

1. 原有技术板块构成是否支持本次升级？否则要做技术扩展，选择新的开发包
2. 选型之后讨论确定技术实现的思路、方案、核心设计模式和关键技术。
3. 技术实现的思路、方案、核心设计模式等问题，必须考虑跟已有实现的继承关系，该留则留，该变则变

注意的要点：

1. 一次升级内容，其中的安排需要考虑的条件因素非常多，一般是由高层和销售运营部门、开发部门、产品部门中层联合讨论决策。
2. 重构任务很多时候技术部门独立决策消化，但在时间和资源紧张时可能需要高层的支持安排。
3. 某些技术环节，例如一些通用组件，在设计和编码上要求很高，也属于关键技术

3.3 架构与代码重构

（1）用 AOP 对异常处理和日志的的重构开发(50 分钟)

本次升级的技术性重构内容主要是异常和日志处理中的大量代码冗余问题，原项目在每个 cotroller 方法中黏贴复制了大量高度冗余的异常和日志处理代码，造成的后果是：

1. 大量与业务无关的冗余代码
2. 异常和日志处理方式不同一
3. 如果要改变处理方式或要优化处理逻辑，需要在无数地方修改代码
4. 新扩展的功能模块继续黏贴复制，代码质量进一步恶化

因此本次升级决定采取重构行动，清理冗余代码，用 AOP 技术开发横切组件，统一处理日志和异常

（2）二．业务模型重构开发(50 分钟)

本次升级的功能内容暂安排为物流跟踪，集成第三方物流 API，针对这个功能做以下分析：

1. 需要增加哪些主要的业务实体，与已有业务实体有何关系？例如订单是否需要扩展字段支持物流状态？平面扩展还是嵌入物流状态信息实体对象？
2. 字典等基础设施信息，例如城市等，是否已经具备？
3. 订单处理流程逻辑中，增加物流支持后，现有服务组件的责任分割是否失衡，是否需要调整？

然后对需要调整的部分安排任务进行重构开发。

（3）其他重构点，安排指导学生尝试重构开发(60 分钟)

1. 日志数据量极大,考虑mongodb库存储日志
2. 其他技术性重构

注意的要点：

1. 日志数据量极大，是否考虑mongodb库？
2. 重温理论课AOP的横切、正交、非侵入式等方法论概念
3. 让学生体会，重构之后，切换mongodb数据库只需要很少的开发支持

3.4 服务端升级开发

（1）技术实现方案分析和设计（40 分钟）

前面的实训课完成了升级功能需求分析，又通过重构为升级功能的技术设计和实现做好了准备，本讲就是具体的设计实现。有以下几个任务：

1. 阅读分析第三方物流API,找出与实现方案有关的部分，比如下单，结算，物流跟踪消息的通知等等。
2. 根据物流跟踪功能设计controller层的api结构，找出需要调整逻辑的现存组件，比如支付之后调用物流API上门取货并启动跟踪。
3. 书写设计文档

（2）服务器端开发（120 分钟）

这部分的教学采取这样的方式：

1. 讲解者根据设计生成有结构、无实现逻辑的空组件，这样的组件形成了功能的结构性框架
2. 讲解者实现一些方法逻辑
3. 课堂上学生实现几个方法逻辑

3.5 客户端升级开发

（1）前端技术介绍(20 分钟)

介绍主流的前端开发技术：

1. VUE、React、Angular
2. Android开发技术

（2）RestfulAPI 设计和约定(20 分钟)

RestfulAPI 在后端开发中已经由后端开发人员形成初版，UI 开发阶段对这个初稿由前后端开发人员协作调整、修改。关于这个开发协作，有几点需要介绍：

1. 后端负责初版是合理的
2. UI与后端的开发实际上是并行的，UI开发在界面基本完成后，开始前后端集成才开始。
3. 前后端经过紧密合作协商，修正API，API基本稳定之后，应该定版本，后续再有调整需要有一定控制。这个定版的时机不同项目、公司有差异，但在启动测试前总要有一版。

（3）UI 代码开发(120 分钟)

本部分教学集中在“集成”的逻辑部分而不是 UI 显示部分：

1. UI显示界面的部分应该已经全部完成，讲解者介绍其设计实现的关键环节
2. 调用后端API的部分未实现
3. 采用讲解者现场编码实现一部分

学生实现一部分

3.6 测试与部署上线

（1）测试基础知识（40 分钟）

1. 测试流程与类型：功能测试、性能测试、集成测试、准生产测试、回归测试的概念
2. 测试工具的使用

（2）模拟测试流程（40 分钟）

做一次“准生产测试”：

学生分成测试组和开发组

1. 测试人员提交bug
2. 开发人员修定bug

（3）上线准备和演练（40 分钟）

1. 编写部署操作手册
2. 按照手册模拟、演练上线流程

（4）上线操作过程（40 分钟）

完成打包、部署、回归测试的系列活动

注意的要点：

本讲重点是通过模拟测试和上线流程对其中的各种人员、活动、环节形成清晰理解。

真实的上线活动还包括客户通知等内容

4. 结论

本文提出的课程设计思路和设计方法用于作者岗前培训课程教学，取得了良好效果。

参考资料

1. Brooks, F. and H. Kugler, *No silver bullet*. 1987: April.
2. Evans, E. and E.J. Evans, *Domain-driven design: tackling complexity in the heart of software*. 2004: Addison-Wesley Professional.
3. Merson, P. and J. Yoder. *Modeling Microservices with DDD*. in *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*. 2020. IEEE.
4. Shadija, D., M. Rezai, and R. Hill. *Towards an understanding of microservices*. in *2017 23rd International Conference on Automation and Computing (ICAC)*. 2017. IEEE.
5. Thönes, J., *Microservices.* IEEE software, 2015. **32**(1): p. 116-116.
6. Fowler, M., *Refactoring: improving the design of existing code*. 2018: Addison-Wesley Professional.