

F.A.F酒店预订系统

项目评价文档

2017-1-1

NJU F.A.F 万年杰 田原 曾虎双 陆茹茹

V1.0



目录

[一、 项目管理评价 2](#_Toc439364410)

[二、 产品评价 2](#_Toc439364411)

[1.产品在项目生命周期中的变化 2](#_Toc439364412)

[2.产品返工情况 3](#_Toc439364413)

[3.开发工具 3](#_Toc439364414)

[4.产品规模 3](#_Toc439364415)

[5.产品质量 3](#_Toc439364416)

[6.产品度量数据及分析 4](#_Toc439364417)

[三、 团队评价 5](#_Toc439364418)

[1. 团队总体评价 5](#_Toc439364419)

[2.个人自我评价 5](#_Toc439364420)

[3.团队会议记录 7](#_Toc439364421)

## 修改记录

|  |  |  |
| --- | --- | --- |
| 修改人 | 修改时间 | 修改原因 |
| 万年杰 | 2015/1/1 | 初始版本 |
|  |  |  |

## 项目管理评价

在本次“酒店预订系统”的开发过程当中，项目管理整体较为良好。项目所使用的过程是迭代式开发，每一个阶段都能有部分产品产生，极大地鼓舞了团队士气。但是，实际的过程和当初的设想有所不同，在当初的设想中，从设计阶段开始，每一次的迭代都应该产生可运行的代码；然而在实际开发过程当中，从构造阶段开始才有了可执行的代码。

在过程当中，由于经验不足的原因，需求阶段和体系结构设计做的不是很好，需要在下次的项目当中强化和改进；详细设计阶段设计的较为出色；构造阶段分工并不是很合理，导致每个人的工作量不是很均衡，好在后来将提前完成自己任务的队员帮助尚未完成任务的队员完成任务，才将整个项目的构造阶段顺利完成，但构造阶段总体时间花费较多，改动太大。

## 产品评价

### 1.产品在项目生命周期中的变化

在项目的需求、体系结构、详细设计阶段，产品主要以文档为主，包括需求阶段的用例文档、需求文档、需求度量文档，设计阶段的体系结构文档和详细设计文档、测试用例文档等，此外包括少量的代码，主要是桩和驱动，单元测试用例等。在项目的构造阶段，代码量发生了巨大的变化，产品基本成型，但还存在不少缺陷，界面也不够美观。在测试阶段，我们修复了找到的缺陷，并美化了界面，同时写了产品测试报告，产品基本达到交付标准，在此阶段我们准备了产品的部署工作。

### 2.产品返工情况

因为项目经验不足，在这次项目开发过程中，我们出现了不少返工情况。问题主要集中在以下方面：

1. 在项目设计阶段发现对需求的理解不足，需求阶段一些关键需求的具体内容没有能细化，导致设计阶段有些功能无法设计，要回到需求文档落实细化部分需求。
2. 在构造阶段，我们发现设计阶段定义的接口不够稳定，出现过一些修改，由此导致层与层之间的冲突，出现集成失败的情况。需要修改设计阶段定义的接口。

### 3.开发工具

在项目开发过程中，我们使用Word编写文档，用ProcessOn画设计图，使用Github进行配置管理，用maven进行项目构建，使用Jenkins进行项目的持续集成，使用eclipse进行软件构造，Junit进行单元测试，使用Metrixs进行软件度量。我们使用大量的工具，成功提高了项目的开发效率和产品质量。

但是在项目开发过程中，我们在Github上遇到一些问题，但我们学习使用Git Shell解决问题，并学习使用gitignore来忽略一些东西，从而使commit更有效，让问题保持在可控的范围内。

总体来说，我们使用的工具支持产品的制造、维护和测量。

### 4.产品规模

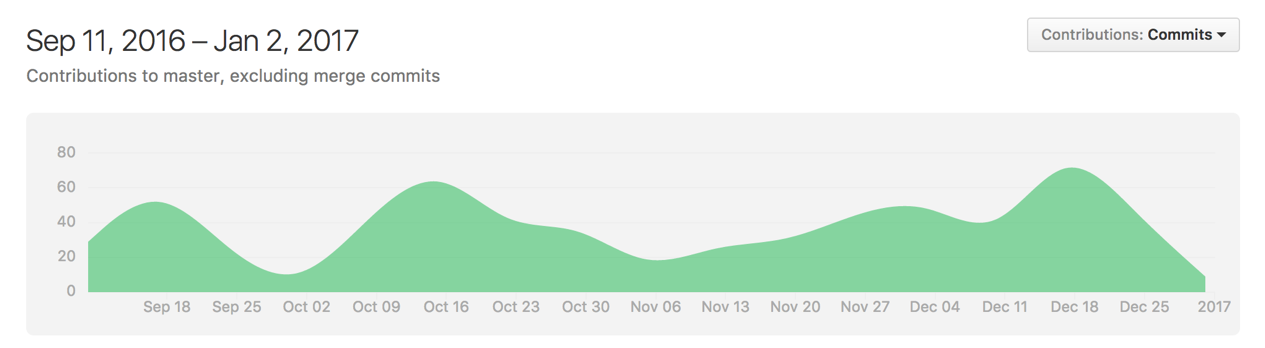
最终产品共有22907行代码。

### 5.产品质量

因为有前期大量的文档支持和采用了良好的设计模式和编码规范，项目的产品质量较之前的项目有很大的提高。但因为开发经验不足和开发能力有限，产品难免存在缺陷。我们在开发过程中已经尽力提高产品的质量，尽我们最大的努力使产品达到交付标准。在测试阶段，我们约有30多个记录的产品缺陷，并修复了这些问题。

### 6.产品度量数据及分析

#### 产品的增长情况



#### 产品在每个里程碑上的测量

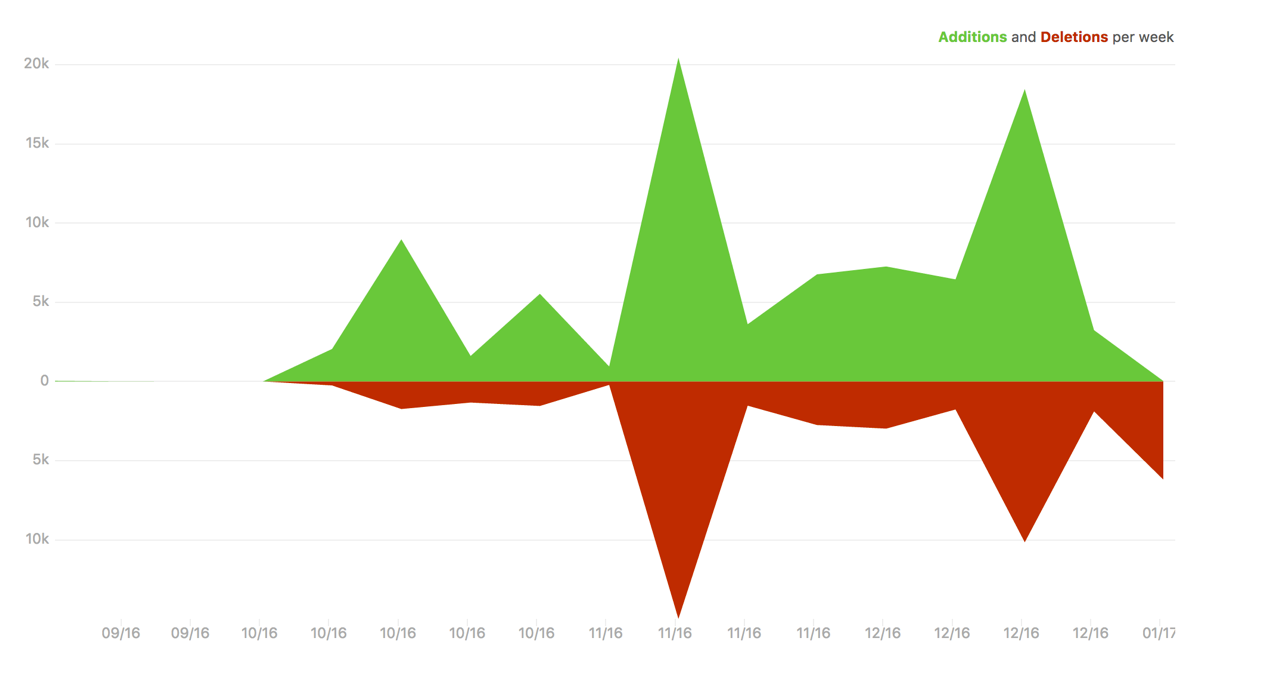
|  |  |  |
| --- | --- | --- |
| 里程碑 | 时间 | 成果 |
| 需求阶段 | 2016/9/18——2016/10/2 | 软件需求规格说明书、软件需求度量文档 |
| 体系结构设计阶段 | 2016/10/3——2016/10/17 | 体系结构设计模型、体系结构设计文档、体系结构原型代码（含桩和驱动） |
| 详细设计阶段 | 2016/10/18——2016/11/13 | 详细设计模型、详细设计文档、详细设计的集成测试代码和驱动代码 |
| 构造阶段 | 2016/11/14——2016/12/11 | 可执行程序（功能实现率86%） |
| 测试阶段 | 2016/12/12——2016/12/19 | 系统测试用例和测试报告文档（功能实现率95%） |

#### 产品复杂度和内容的测量

代码量为22907行，共有373个类。该产品全部实现了酒店预定系统的基本功能需求，实现了酒店预订操作的一体化。

#### 过程和工具对产品的影响

本项目采取迭代式开发，每次迭代周期为2个星期。开发过程的频率如下图。



采用Jenkins、maven进行构建和集成，提高了集成的效率。利用junit进行测试驱动开发，减少了代码的缺陷。通过git进行项目的同步和备份，使小组成员可以随时获得项目的最新进度，也有助于项目开发过程中的回退。

## 团队评价

### 团队总体评价

团队总体评价良好。管理当中常见的问题有组内人员空闲时间不统一，导致难以确定开会和集体开发的时间；当出现争执时不能够冷静地通过讨论优缺点来提出解决方案，容易情绪化。主要优点体现在定好了会议时间以后，大家都能较为准时地参加；大家都能以团队项目为重，能够意识到“团队的失败就是每个人的失败”，当出现问题的时候都能够停下手中的工作来修复项目问题；但在布置任务之后常常不能按时完成，这是态度上需要改进的地方。

在团队过程当中，风险集中出现在构造阶段，由于对项目代码量的没有准确的把握和最后期限的限制，导致组内对于能否按时完成构造产生了一种怀疑的态度，好在随着过程的进行，组员们渐渐意识到项目能够按时完成，有了充足的信心。相信这样的问题能随着经验的上升慢慢改进。

在团队管理当中做出的最重要的决定应该是构造阶段的分工决定。由于构造阶段较为重要，分工方式又有很多，但是考虑到我们的体系结构是分层式的设计，拥有并行开发的特点；又考虑到界面层要学习很多并不掌握的UI知识，还要承担美工的任务；最后，考虑到我们做这个项目仍然处于一个学习的阶段，我们的决定是让每一个组员尽量多的参与项目的每个过程，所以万年杰、曾虎双、陆茹茹承担了界面层的所有任务，和逻辑层的大部分任务，田原承担了逻辑层的部分任务和数据层的所有任务。

### 2.个人自我评价

**万年杰——**在这次项目实践的过程当中，我第一次作为队长带领这个团队，说实话，压力还是很大。这些压力主要表现在每一次分工的过程当中，要充分考虑组员的技术优缺点，作业量的不同，最为合理，最为公平地分配任务，这也就要求你首先对自己的组员有一个比较全面的了解，但这是我们第一次合作，相互的了解并不多，这也就导致之后一些问题的出现。在看法不一致的情况下，我都希望能通过沟通和分析达成一致，但是难免有谁也没有办法说服谁的情况，这时候只能听组长(我)的安排，有时候组员会很抵制，但是之后还是会完成任务。

在这次的项目实践中，我主要承担了UI层和订单逻辑的工作，有时也要去看看数据层的代码，因此，我也学习了数据库的相关语句操作，包括建表、增删改查等。在UI的构建过程中学会了用JavaFX来构建UI的实现，以及了解了FXML和Controller的对应机制，JavaFX的反射机制，CSS的映射机制。为了Github不发生很多冲突从而正常运行，我学会了如何利用git shell，如何编写.gitignore文件。在工具使用方面，我学会了如何用Junit进行单元测试，如何用用Metrixs进行软件度量。

总之，我们组最终还是实现了全部需求，产生了可执行的文件，组内关系融洽，建立起了很好的友谊，这也是这次大作业除了技术方面我获得的最重要的东西吧。 学到了很多，也感受到了很多，更多的是看到了和别人的差距，日后还需努力，以此共勉。

**崔浩——**在这次大作业过程中，我在需求设计等阶段和组员合作完成文档的编写，在构造阶段主要负责界面层架构的设计，并和组员一起完成界面层的编写。我能积极参与团队的每次会议，但在开会过程中，我经常和组长和组员有一些冲突。但最终我们都能做出让步并互相妥协。在这次大作业的过程中，我增进了对组长和组员的了解，意识到团队的每个人都是项目不可或缺的一部分，一个良好的项目一定是团队的所有人合作完成的。尽管冲突不可避免，但我们学习掌握处理冲突的方法才是最重要的。

**杜天蛟——**在完成这次项目的过程中，我学习了java的反射机制，通过反射构建类的对象，通过类名和方法名执行方法，减少了大量重复代码。我还学习了xml文件的配置和读取，利用第三方jar包实现，把界面的所有组件写在xml文件里，通过读取xml文件和反射的结合来创建组件，所有的位置、大小以及其他信息都写在了xml文件里，避免了硬编码现象。

这次项目也培养了自己面向对象的思维，也想了一些办法减少代码的圈复杂度，比如增加类，通过类名执行方法等。

但是在这个过程中也发现了自己的不足，比如考虑问题不周全，没有养成防御式编程的习惯。还有项目即将结束时，自己设计项目框架的能力还非常弱，对整个项目没有整体把握，只能在小范围内实现一些具体的功能。

在详细设计的过程中，自己写了项目所有的po和大部分接口、vo，但是在写代码的过程中却发现很多设计都非常不合理，也给自己带来了很大麻烦。相信吸取了这次的经验后，下次设计会有一些进步。

**高露——**在做大作业的过程中，我学到了不少知识。首先自然是对分层的体系结构风格、模块化等有了更深一步的了解，对各层之间的分工、协作以及其之间的关系也有了了解。在逻辑层的实现中，操作基本上实现vo，po的相互转换，在实现的过程中，有时也要去看看数据层或界面层的代码。在此过程中，我也学习了数据库的相关语句操作，包括建表、增删改查等；也了解了一些界面层与数据层用到的反射的机制，这对我的学习无疑是一大提高。当然在完成大作业的过程中，我也发现了自己的很多不足。比如逻辑层的异常处理机制做的不好，只是随意的抛出异常并做可能并不合适的处理；有一些代码重复的现象；有时候只是为了实现一些功能，而不考虑与其他模块的联系，忽略内聚耦合等等。而正是有了这些问题，我也学到了一些知识，比如异常处理的方法、抽象工厂模式等。经过一次大作业的锻炼，相信以后我也会不断提高自我。

### 3.团队会议记录

在前期的合作过程中，我们组主要由万年杰负责会议记录，后来发现这种统一记录的方式效率过于低下，于是后来我们采用了各自记各自的“分布式会议记录法”，但总体的问题仍然由万年杰记录，发现效果确实不错。



