北京航空航天大學

软 件 工 程 综 合 实 验

Blade 框架分析

综合实验总结

团队名称	B 组
指导教师	刘 超 任 健
培养学院	计 算 机 学 院

1 项目概述

Blade 项目从 3 月 14 日开始至 6 月 22 日共用时 432.02 个实际工时，如图 1.1 所示，项目初期计划工时为 433.31 个工时，但是在实际完成中用了 430.02 个工时，由于本组内有 3 人上学期有高等软件工程课的上课经验，对软件工程了解的比较多，所以本组总工时与其他组相比少很多，但是这并没有影响到本组作业的质量。

从工时数据来看，工时的计划与实际完成相差不大。但是实际上，在项目进行中，需求的增加和删除，以及完成某项计划的实际时间多于或少于基线时间，最终的结果只是恰巧与计划工时相差不多，而不是因为自己前期计划的合理性较好。

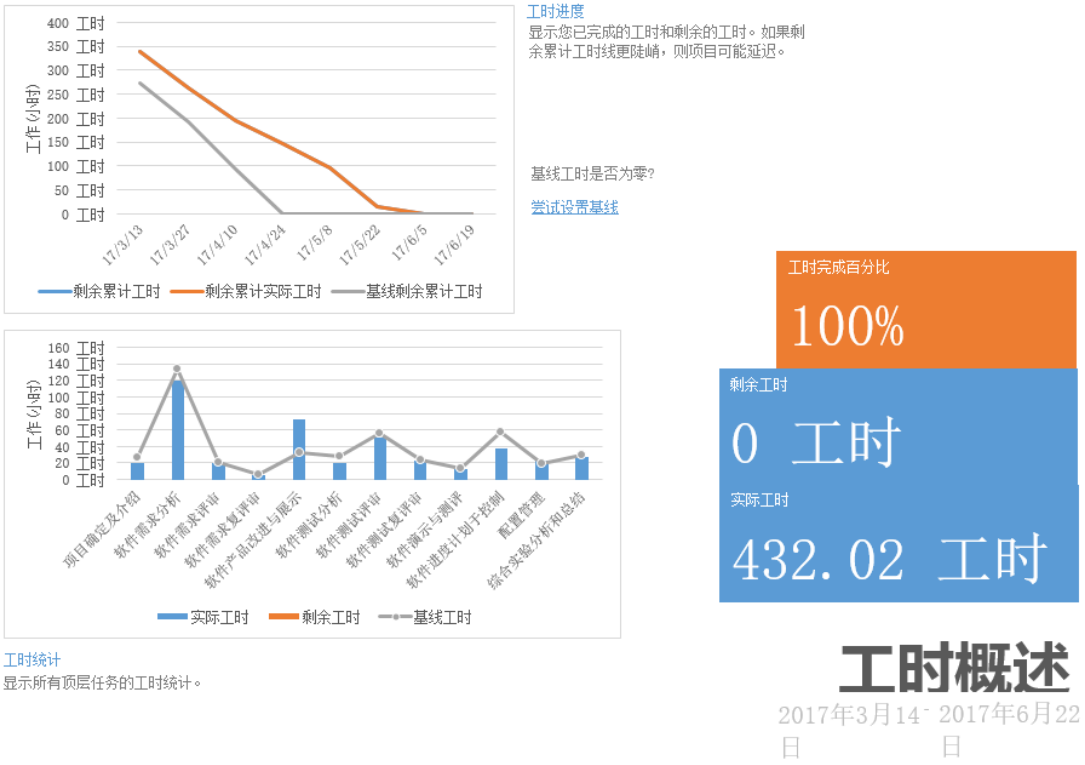


图 1.1 项目工时总览图

对于实验过程的个人工时展示如图 1.2 所示

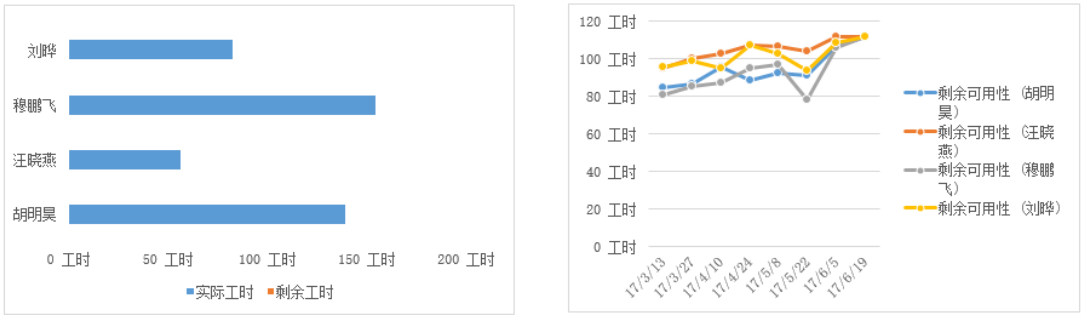


图 1.2 项目个人工时概述

本次课程实验从 3 月 14 日项目确定开始到 6 月 22 日项目完成，共进行了 8 次实验，

由于在实验过程中受到老师的指导，Project 工具的使用，以及组内人员的配合，较好的完成了 8 次实验内容，随着实验的进行，组内不断总结实验过程的经验教训，以便更好的服务下一次实验。针对最终的课程总结，组内人员也分别对实验一到 8 进行了分析，下面一一介绍。

2 实验一 需求分析

2.1 需求分析数据统计

实验一需求分析从 3 月 14 日开始至 4 月 14 日共用时 140 个工时，进行了 7 次版本迭代，完成了 13416 字的需求分析文档。文档撰写过程中，遵循需求文档撰写规范，绘制了包含用例图，类图，时序图，状态图以及其他辅助说明图形。具体统计如图 2.1 需求分析数据统计表所示。

表 2.1 需求分析数据统计表

实验一：需求分析					
需求文档规模	需求项数	其他模型	其他模型中包含元素累积数	版本更新次数	累积工时（含需求修改工时）
13416 字	8	用例图（1）	20	7	140h
		类图（5）	40		
		时序图（8）	134		
		状态图（2）	12		

2.2 需求分析数据分析

在需求分析过程中，由于从确定分析 Blade 项目开始到需求分析文档撰写仅仅 4 天时间，对于 Blade 框架，小组成员在课程之前基本属于零基础，且只有 4 个人员。故由于时间与人员的原因，无法对整个框架进行完整，详尽分析，故对框架进行精炼，找出包括 IOC 模块，配置模块，数据库模块等 8 个 Blade 框架的核心模块进行分析，并对其绘制了相应的用例图并根据分析结果撰写 RUCM。

同时为了更清晰的表达需求内容，从面向对象的角度对需求分析进行分析，为每个模块设计了类图，为了了解每个模块的执行过程，设计了时序图，为了描述系统执行过程中状态的变化，设计了状态图。为了方便用户阅读也设计了 RUCM 与类图的对应表。

在需求分析中的文字部分为 13416 字，但是由于需求文档中的各种模型以及 RUCM 都是以图表形式表现，故无法统计字数，实际字数可能多于 13416 字。

在模型分析上，分析的重点有广度与深度之分，广度就是分析的模块比较多，但是了解的不深，深度就是分析的模块较少，但是分析的深度比较深，基于本小组时间与人员的特点，本小组选择的少而精的分析，故选择了核心的 8 个核心模块，为了更加深入分析各个核心模

块，同时更好的展现分析结果，故做了其他模型的绘制，如时序图，类图，状态图等。

对于版本迭代，由于实验一持续时间较长，且处于实验开始阶段，对实验所做的东西不是很熟练，故修改次数较多，达到了 7 次。同时工时也达到了 140h。

2.3 需求分析文档质量

经过需求评审与复评审对需求说明书的不合理的部分进行了修改，通过刘超老师的意见，为需求说明书添加了类图与 RUCM 的对照表，对前期文档偏设计的部分进行改进。使需求说明书更符合在实际开发中需求说明书的要求。

需求分析作为第一个实验，在前期有许多考虑不周的地方，如改进部分，没有细化软件改进和扩展的部分。同时随着实验的进行，需求不断增加，不断的对需求说明书进行更改，在实验的进行过程中也不断发现问题并对发现的问题进行修改。实验一贯穿于整个实验过程中，所以比实验一较符合实验要求，完成质量较高。

2.4 需求分析有效方法

在需求分析过程中要先对选择开源项目的功能进行梳理，进行用例图的绘制，之后根据组内人员的特点，分别分配相应的功能进行 RUCM 的分析，然后进行 RUCM 的撰写。为了使需求文档更符合规范，采用面向对象的思维进行需求分析，故根据 RUCM 与相应的功能进行类图，时序图，与状态图的绘制。

对于绘制的各种模型图，由于在评审时难以进行准确的评审，同时对于其他人进行阅读有一定困难，故需要设计类图与 RUCM 的对应表，以方便评审与用户阅读文档使用。

2.5 结论与建议

需求分析作为第一个实验，在前期有许多考虑不周的地方，如改进部分，没有细化软件改进和扩展的部分。同时随着实验的进行，需求不断增加，不断的对需求说明书进行更改，在实验的进行过程中也不断发现问题并对发现的问题进行修改。实验一贯穿于整个实验过程中，所以比实验一较符合实验要求，完成质量较高。

在需求分析过程中，一定要把握好需求与设计的差别，要从需求的角度去分析撰写需求文档，而不是以实现角度去撰写需求文档。

对于需求分析而言，要尽量按标准需求分析说明书撰写，为了保证高标准的文档撰写，可以减少模块的分析，对于选择分析的模块，要尽量分析的透彻，除了使用 RUCM 描述模块功能外，也需要使用其他 UML 模型进行辅助说明，从而使文档更完备，能更好的完成需求的描述。

3 实验二 实验总结

3.1 数据统计

表 3.1 需求评审数据统计表

实验二：需求评审（评审+复评审）						
检查单中的检查项数量	互评审中给被评审组提出的问题数（评审+复评审）	接受到的问题数（评审+复评审）	老师的问题数（含各组存在的共性问题）	接受并修改的问题数	评审报告字数	累计工时
15	112	47	51	75	5508 字	25.43 工时

3.2 数据分析与说明

3.2.1 数据获取

评审数据来源于项目下评审表格，在实验末期，我组将所有评审的内容进行了一次汇总，汇总后数据存放在“C-D 评审汇总”、“D-B 评审汇总”、“B-A 评审汇总”以及“B-G 评审汇总”四个 Excel 文档中。

文档记录了评审问题的位置、问题的描述、问题的类别和提出人、严重性等信息用于辅助被评审组修改。

3.2.2 偏差估计

在统计的过程中，部分评审问题我们在修改的时候使用 Excel 样式对“拒绝”，“接受”，“待修改”三种问题进行了标记，但部分文档由于不同人不同的操作方式，并没有对其标记因此存在误差。但评审问题数量均为准确结果。

3.3 有效方法

3.3.1 规范输入

评审问题的表格在大体上对格式进行了统一，但是对问题的描述仍然可以尽量的统一。Excel 提供了候选项，因此可以直接将意见、种类、等级等信息直接制作为候选项，在评审的时候能够做到一个组的输入较为统一。

3.3.2 通过颜色标记问题状态

项目名称	blade分析					
文档名称	需求规格说明书	版本号	1.3			
提交日期	2017/4/6	编制人	汪晓燕			
评审日期	2017/4/9	评审方式	讨论区评价			
序号	问题位置	问题描述	问题类别	报告人	严重性	处理意见
1	大标题，第一页	软件工程综合实 丢了 验 字	文档内容错误	D组李岳檑	高	很严重，第一页
2	3.3	可以多分几个小标题，1）2）后面又一个1）2）	文档版式错误	D组李岳檑	低	
3	全文	图标题加粗	文档版式错误	D组李岳檑	高	建议添加标号和表的说明
4	4.8	图没标题	文档版式错误	D组王春柳	中	添加
5	目录	目录格式有问题	文档版式错误	D组王春柳	中	调整格式
6	第5页	这一页是空白页	文档版式错误	D组谭伟良	低	删除空白页
7	3.5	非功能需求能否写的更加具体	文档内容存在疑问	D组谭伟良	中	建议详细描述非功能需求
8	3.3节	用例截图空白地方不要截取太多，看不清里面的内容	文档内容存在疑问	D组谭伟良	低	建议修改
9	3.9	安全和保密是否可以归到非功能需求	文档内容错误	D组谭伟良	低	修改
10	全文图片	部分图片下面缺少文字说明，未居中排放，部分图片显示不清晰	文档版式错误	C组胡勇	低	修改
11	图25、图26	状态迁移图不规范，圆矩形作为状态，在无判别的情况下，不会出现两条分支	文档内容错误	C组胡勇	中	修改
12	2.4运行环境	JAVA作为跨平台语言，操作系统平台由jvm决定	文档内容错误	C组胡勇	中	说明
13		缺少模板引擎、缓存插件、负载均衡等定义	文档内容错误	C组王益飞	中	新增
14	1.2	1.2文档约定部分表述重复，而且没有明确指出ieee发布的文档约定的版本号	文档版式错误	C组王益飞	低	修改
15	非功能性需求	缺少非功能性需求	文档缺失	C组王益飞	高	新增
16	2.1	Maven、Restful风格并未在术语定义中定义。	文档内容错误	C组武丁淫宇	中	新增
17	2.2	表1名称是 Blade 工程资源结构说明"，而上文描述的是 JAVA 源码各部分参数意义"	文档内容存在疑问	C组武丁淫宇	中	修改
18	2.4.2	需安装以上 1.8 以上版本的 JDK"此处多了一个 以上"	文档内容错误	C组武丁淫宇	低	删去
19	目录	目录缺少第六章索引	文档版式错误	C组郭炜峰	低	新增
20	业务需求	缺少业务需求	文档内容错误	C组郭炜峰	高	新增
21	2.4运行环境	blade作为高级语言框架，为什么会对底层cpu厂商、型号、频率做出限制	ppt内容存在疑问	C组郭炜峰	中	说明

图 3.1 评审问题状态追踪

在评审中，问题基本上处于三个状态，“接受并修改”，“接受待修改”，“拒绝修改”三个状态，因此通过 Excel 的样式标记，可以使用绿、黄、红分别标记，可以快速的在组内跟踪问题的修改情况和剩余情况。

3.4 总结

在实验中，我们对 A 组和 G 组的需求规格说明书进行了评审，在每次评审之后，对方组的文档都有质量上的提升。但是，也发现一些问题，例如我们提出的文档内版本号不统一的问题，在多次评审后仍然存在。

对于评审组提出的问题，我们均加以重视，并通过综合的考虑对需求规格说明书进行了多次的修改。在规格说明书中，我们主要对 Blade 的需求和模型进行了描述，在多次测评后，我们对 Blade 的需求也有了更深的认识。

4 实验三 实验总结

4.1 改进说明

Blade 是基于 IOC 技术的 JavaWeb 框架，其特性就在于开发者无需声明变量即可使用。但 IOC 必须在 Blade 框架完全启动才能生效，JUnit 无法自动生成 IOC 环境，因此无法对基于 Blade 框架编写的应用程序进行测试。这使得 Blade 开发的应用程序变得不安全，我们小组认为易用的 JavaWeb 框架不应该有此缺陷，因此致力于完善 Blade 框架在测试方面的不足。

为了弥补 Blade 框架在测试方面的缺陷，我们结合 Blade 和 JUnit 为 Blade 添加了测试模块，向 Blade 提供了模拟请求和单元测试的可能。

4.2 数据统计

实验三：改进与扩展

设计实现规模	其他模型	其他模型中包含的元素累积数和边的数量	代码行数	问题数	修改的问题数	版本更新	累积工时数
4651 字	RUCM (1) 类图 (2) 时序图(1) 状态图(1) 组件图(1)	元素个数：64 边数量：47	1284 行源码 139 行注释	1	1	2	72.43 工时

4.3 数据分析与说明

4.3.1 数据获取

数据的获取均来自于实验中的即时记录。其中，元素的个数和边数可根据 StarUml 中的模型元素获得，文字个数通过 Word 显示的左下角文字统计获得，代码行数通过 Cloc 进行了统计，问题个数根据刘超老师对文档的批注进行统计，而版本更新数量通过实验七的版本控制可以获得。

4.3.2 偏差估计

本实验的统计数据较为准确，从改进与扩展的角度来看，我们做了如下操作来保证实验的质量：

1） 从需求分析开始严格按照软件工程瀑布模型进行设计与开发

2） 应用适配器模式结合了 Blade 和 JUnit 两个开发库

3） 设计与改进均通过了实验四的测试

从统计数据上看：

1） 使用业内广泛使用的代码计算工具计算了代码行数

但对于问题的追踪存在疑问，在实验的过程中，并没有对于实验三进行评审，因此仅统计了三周的时间里，老师在文档中针对改进部分的批注数量。

4.4 改进测试

4.4.1 测试情况

在测试阶段我们对扩展的需求进行了测试，在改进部分，我们一共提出了 3 个测试点，测试 BladeTest 模块能否正常的工作并满足改进需求。

在测试阶段，小组本该进行单元测试，但由于采用了适配器模式和外观模式将原有的方法均封装在组件内部，因此仅针对测试点使用了集成测试。在测试过程中，一共对 3 个测试点的 4 个测试用例进行了测试。测试覆盖了所有的测试点，并给出了相应输入和输出，详情

课参见实验七的文档。

4.4.2 测试结论

基于测试阶段获得的数据，我们认为 BladeTest 测试模块达到了预期的需求，能够向使用 Blade 开发的应用进行单元测试，但考虑到测试仍然是一个广博的研究方向，因此只能认为通过 BldaeTest 可以对 Blade IoC 管理的模块进行调用并获得特定的输出，该输出与 Blade 正常工作时的结果相同。

4.5 总结与建议

在改进实验中，小组对 Blade 框架内部的实现有了更深理解，并且为了保证 Blade 在不同版本的可用性，尽可能的调用 Blade 内部的方法进行实现，能够保证当 Blade 升级到一个可运行版本时测试模块依旧能够运行。

而从改进和扩展的角度来说，最好在实验开始之前选择一个自己比较熟悉的、体量较小的程序入手，否则可能在实验期间内不足以做出需求上的增加，甚至不能弄懂框架内部的基础原理。

5 实验 4 软件测试总结

5.1 软件测试说明

Blade 是一款简洁易用的 JavaWeb 框架，它抽取了 Spring 的核心功能并重新实现。Blade 在简洁和兼容两者之间选择了简洁，摒弃了繁复的配置，选择了 Java 1.8，以及内嵌的服务器和数据库。它提供了 IOC 容器、MVC 架构支持、模板引擎以及注解功能，并基于 Maven 进行管理。本实验主要对 IOC 容器管理、配置管理、数据库操作、拦截请求、路由转发以及框架并发性能等方面进行测试，主要集中在功能测试和性能测试上。

5.2 数据统计

如下表所示统计软件测试过程中涉及到的测试需求和报告、测试用例、测试数据等相关信息。

表 5.1 测试需求数据统计

实验 4 :测试需求						
测试需求和测试报告等文档规模	测试需求用例数	测试数据（实例）数	测试类型	覆盖率	测试需求文档更新版本数	累积工时

10900	12	44	功能测试、性能测试	100%	4	70h+

如下表列出了实验所有相关的制品。

表 5.2 测试需求实验制品统计表

实验	序号	文件名	最后提交时间	最后提交人
四	1	软件测试说明书	2017/5/18	穆鹏飞
	2	软件测试说明书_v0.5	2017/5/23	刘晔
	3	软件测试说明书_v1.1	2017/5/25	穆鹏飞
	4	软件测试说明书_v1.2	2017/6/1	刘晔
	5	测试模块需求分析文档	2017/5/18	汪晓燕
	6	测试需求规格说明书检查单	2017/5/31	汪晓燕
	7	路由转发模块测试	2017/5/25	刘晔
	8	测试对照表（目录）（10 个文件）	2017/5/31	穆鹏飞
	9	测试检查单（目录）（10 个文件）	2017/5/25	穆鹏飞
	10	对 A 组与 G 组的评价（目录）（ 6 个文件）	2017/5/25	穆鹏飞
	合计	33 个文件		

5.3 数据分析

表 5.1.1 来自于实验四中软件测试说明书小组各成员对各自测试部分的测试用例、测试数据、测试类型等进行的统计归纳。表 5.1.2 来自于 github 上实验四所产生的所有相关制品收集与提交日期、人的整理。

在实验四的计时工具记录的原始工时数据我们可以比较清晰的知道本组各成员在实验四期间的工作记录，不难发现，小组各成员都在积极参与，分别完成实验四需要的各项任务。根据原始工作记录数据，大体上获取到了各成员在实验四的工作时间占比情况以及实验四的总工时数量。通过对 github 中的实验四主要制品：软件测试说明书的内容字数、需求个数、测试用例个数以及版本更迭数等进行汇总以及分成员统计，可以较为简要的判断小组各成员的工作量情况。实验四过程总共产生了 3 3 个文件（当然这部分没有把测试程序源代码文件包括进来）。以上三个方面小组各成员的工作情况进行一些简要的互证，表明了本实验的确实付出了相当的工作量，完成了相当的任务，同时小组各成员都积极参与，较好的完成的实验目标。

在软件测试需求说明书中，我们一共产生了四个版本。初期开始做的时候，计划产出两部分制品，即软件测试需求书和软件测试说明书。有别于软件需求书，软件测试需求书，指的是对软件测试的哪些目标需要进行测试。软件测试说明书，指的是实施的软件测试的过程

的描述说明。之后由于具体实施时，两个制品内容趋同，所以之后只有软件测试说明书。在软件测试说明书中，小组成员共分析了 OC 容器管理、配置管理、数据库操作、拦截请求、路由转发以及框架并发性能等 6 个方面共计 1 2 个测试用例，提供了 4 4 组具体的测试数据，完成了 1 0 个对照检查单和 1 0 个覆盖表。总体而言，软件需求说明书对这 6 个方面做的覆盖是比较全面的。

5.4 总结及有效方法建议

软件测试是软件工程中至关重要的一步。在对 Blade 及改进部分 BladeTest 的测试过程中。初期，小组成员没有对测试用例概念取得共识，经过课堂上与老师以及同学们的交流，小组制定了测试用例规范，在单元测试，集成测试，性能测试等方面对 Blade 的路由，IOC，数据库，处理性能等进行了全面的测试，发现了一些软件缺陷。另外，对于改进部分 BladeTest 也做了较为全面的功能测试。总体上来看，达到任务预期，较好的完成了软件测试的目标。

有效方法建议。在软件测试过程中，一定要弄清楚要测什么，以及为什么要测它这两个问题，因为在一个软件项目中，不是所有点都是必须测试的，考虑到时间的因素，我们往往只需测试软件的关键部分或者重要性能就足够了。在本次 Blade 实验中，小组分别测试了框架的数据库部分，IOC 部分以及配置部分以及框架的并发性能等，这些被测试项都是一个 MVC 框架中极为重要的功能点或痛点。同时，测试时我们要把自己当成“苍蝇”，想尽办法，就算被测试软件是无缝的蛋，也一定要试着看能不能钻进去，何况任何软件都无法保证是完美的。

6 实验 5 软件测试评审总结

6.1 软件测试评审说明

软件测试评审是对软件测试可靠性的第三方验证。我组对 A 组 dexJar 和 B 组 Torch 进行软件测试评审工作。

6.2 数据统计

表 6.1 软件测试评审数据统计

实验五						
检查单中的检查项数量	互评审中给被评审组提出的问题数	接收到的问题数	老师的问题数	接受并修改的问题数	评审报告字数	累积工时
15	25	38	10	40	839	49

表 6.2 实验五制品统计表

实验	小组	评审文件名	问题个数	提交时间

五	A 组	Dex2jar 的测试及优化需求说明书	9	2017/5/24
	A 组	测试覆盖表	4	2017/6/15
	G 组	基于 Torch 平台的神经网络压缩研究与应用测试需求说明书	8	2017/5/24
	G 组	测试覆盖表	4	2017/6/15
	合计	共 2 次评审，4 个评审表	25	

6.3 数据分析

在软件测试评审的两个阶段过程中，小组在第一阶段对 A 组和 G 组的软件测试说明书等进行评审，发现了其中设计文档格式不规范，测试用例说明模糊等共计 2 5 个问题。小组也收到了 C 组和 D 组涉及测试用例规范等共计 3 8 个问题。在第一个阶段过程中，小组发现，对于 B 组 dexJar 本身这一功能性很强很专一的工具，工具本身值得 B 组可测试的模块比较少，所以 B 组专注于对 dexJar 这一整体功能表现的测试上，小组表示认可和理解。在对 G 组 Torch 平台的复评审过程中，由于 Torch 比较偏于神经网络算法理论，理解难度比较大，小组成员努力去发现其测试说明书、覆盖表中涉及内容、格式规范方面的问题共计 12 个。在第二阶段对 A 组的 dexJar 现场测试时，我们在 A 组的协助下，学习了 dexJar 强大的反编译功能，在对测试数据进行测试获得了测试结果，与 A 组结果保持一致，证明了 A 组同学测试结果是值得信赖的。、

6.4 总结及有效方法建议

软件测试评审是整个实验的一个重要部分，是在整个测试阶段对其他组的测试需求说明报告进行详细的评审过程，在都其他组进行评价的同时也接受其他组对我组的评审并进行修改，然后再提价修改后的内容接受老师和同学们的复评审，然后上课时在将这一部分内容体现在 ppt 上，和老师同学们交流。评审可以帮助我们发现团队中疏忽的一些问题，对于软件测试报告的改进还是很有帮助的。

有效方法建议。在软件测试评审及复评审过程中，建议前期多多了解前两组同学的项目。在本次测试评审及复评审过程中，确实感觉到前期缺乏了解造成测试评审过程中的无法深入，只能透过表面文字，在其他小组协助下才能顺利完成软件测试评审工作。

7 实验六 项目计划

7.1 项目计划数据统计

项目计划从项目准备开始，直到项目最终总结供 17 周用时 38.22h，对实验一到 5 每次实验都在前期做了详尽的计划，以便在具体工作时，可以更好的完成计划。项目计划数据统计如表 7.1 所示。

表 7.1 项目计划数据统计表

实验六：项目计划				
项目计划书、小组会议记录累积字数	项目中任务项数	项目计划耗费实际工时数	项目计划更新次数	项目累积实际工时
9309	208	38.23h	16	430.02h

7.2 项目计划数据分析

在第一次项目计划完成后，在每次实验开始前，都要开会进行讨论，进行新的项目计划的撰写，以便能够更好的完成相应的任务，在所有实验完成后对实验过程中的项目计划进行总结，并形成总结分析报告，这两项一共 9309 个字，由于前期初学 Project 的使用，对计划做的比较细，在后期意识到这不符合计划的规范，故对计划方式进行修改，项目计划数偏多共 208 个。

由于前期对 Project 工具不熟练，前期 Project 计划中用时较多，随着实验的进行，Project 熟练度的提高，Project 的使用时间逐渐缩短，在后期的计划中用时较少，故 Project 计划中所用工时为 38.23 h。

课程总共分为 16 周，从第一周开始每周都进行 Project 的计划，以便更好的完成工作，故项目更新供进行了 16 次。

7.3 项目计划质量

项目初期计划工时为 433.31 个工时，在实际完成中用了 430.02 个工时如图 5.1 所示，由于本组内有 3 人上学期有高等软件工程课的上课经验，对软件工程了解的比较多，所以本组总工时与其他组相比少很多，但是这并没有影响到本组作业的质量。

从工时数据来看，工时的计划与实际完成相差不大。但是实际上，在项目进行中，需求的增加和删除，以及完成某项计划的实际时间多于或少于基线时间，最终的结果只是恰巧与计划工时相差不多，而不是因为自己前期计划的合理性较好。

根据图 7.2 可以看出，项目整个过程中基线工时与实际工时基本相符，这也表现了，项目在前期 Project 计划的合理性。对于软件产品改进与展示大大超出基线工时，是由于对软件的改进难度估计不足，由于为了使改进部分能够有更好的鲁棒性，对软件改进部分做了部分测试，这在之前的计划中也没有考虑。

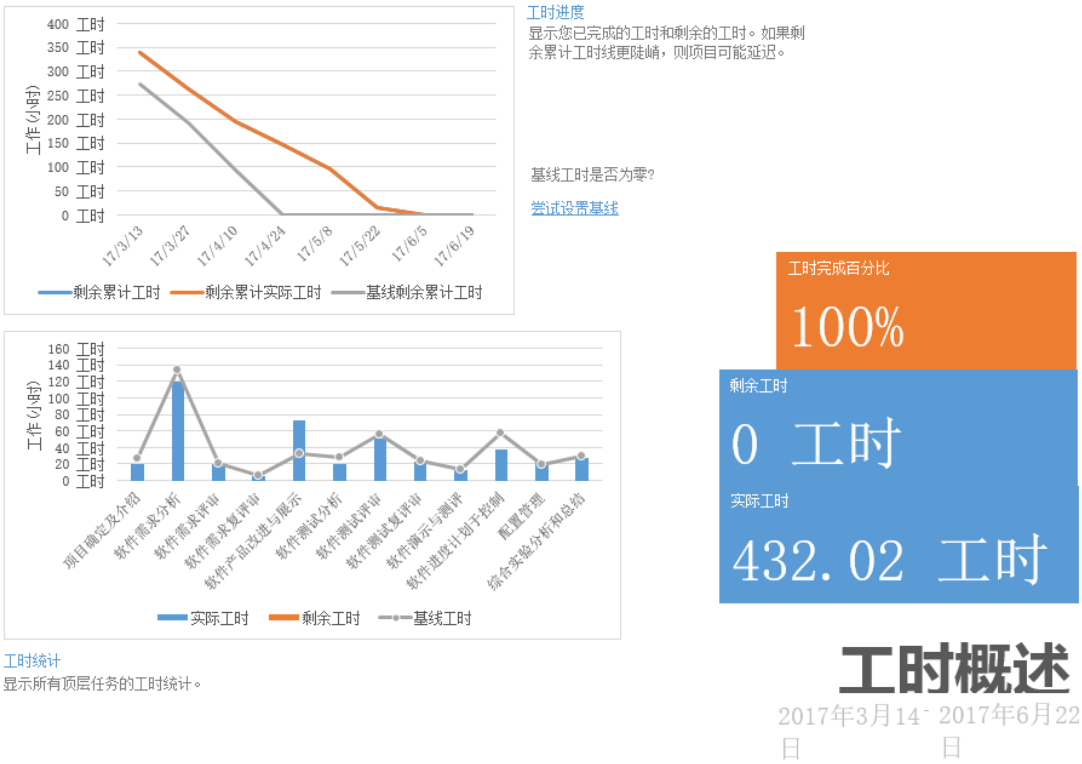


图 7.1 项目工时总览图

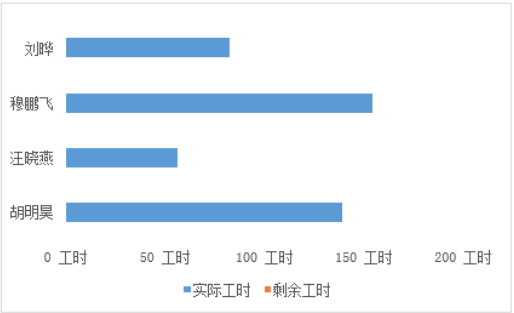
对于软件进度计划与控制而言，由于前期过程对软件不熟悉，对老师要求的内容理解不透彻，导致前期在 Project 计划中用时较多，这也导致了对后期计划的制作造成影响，从而导致基线工时偏多。但是 Project 工具的使用，熟练度的提高，随着项目的进行，Projec 的使用时间逐渐缩短，这就使软件进度与计划控制时间偏多。

实验六中的个人工时如图 7.3 所示，从图中可看出，个人工时分配很不均衡，这主要由于以下三个原因

1）由于在计划执行过程中，有的人未使用工时记录工具，

2）记录的时间由于个人因素不是很准确

3）对工具的使用不熟练，对作业内容不熟悉。如穆鹏飞同学在实验六过程中，前期对 Project 工具使用不是很熟悉，导致前期 Project 计划中浪费的时间较多。在配置管理过程中，随着文件的增加，胡明昊同学在整理文件时用了很多时间，最后为了更好的完成配置管理工作，还开发了相应的工具，这也归属在实际工时中。



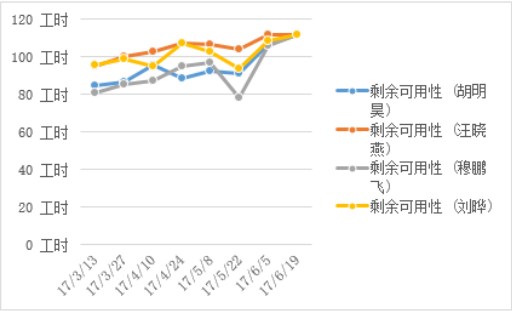


图 7.2 实验六个人工时概述

7.4 项目分析有效方法

对于实验六，前期一定要考虑各类异常情况，如病假，事假等，预留出流动时间，从而能使组内人员更好的完成计划。

对于任务分配，一定要遵循按人员特点，任务属性进行任务的分配，这样能最大化每个人的能力，也能更快更好的完成相应的任务。

7.5 项目计划结论与建议

对于计划准确性而言，对于可能使用到工具的地方，包括自己开发工具，使用别人的工具，一定要预留出相应的时间，同时对于使用别人的工具而言，要考虑到随着工具的使用，熟练度的增加，使用时间会逐渐缩短，这些都需在前期计划中进行考虑。只有充分考虑这些内容，才能让计划更有效，更准确，从而达到更好完成项目的目的。

对于 Project 的计划，在初始时，一定要按照老师的要求，组织 Project 中的相应列，同时对于发现的问题应该尽快修改，否则随着错误的积累，到后期错误会越来越多，不但导致错误修改繁杂，同时也会影响对后期 Project 分析的准确性。

对于每次实验内容，以及老师新增的需求部分要及时与老师交流，明确老师的要求，从而能够更准确的组织 Project 中的内容。

对于人员分配而言，要根据每个人的特点，按特点分配任务，同时要多开会多交流，这样能更快更好的完成项目。

在课程前期要保证 Project 的正确性，否则对后期的计划分析有很大影响，对于各项实验内容，可以详细列出实验内容，以方便 Project 的准确制作。

8 实验七 实验总结

8.1 配置管理

Blade 是基于 IOC 技术的 JavaWeb 框架，其特性就在于开发者无需声明变量即可使用。但 IOC 必须在 Blade 框架完全启动才能生效，JUnit 无法自动生成 IOC 环境，因此无法对基于 Blade 框架编写的应用程序进行测试。这使得 Blade 开发的应用程序变得不安全，我们小

组认为易用的 JavaWeb 框架不应该有此缺陷，因此致力于完善 Blade 框架在测试方面的不足。

为了弥补 Blade 框架在测试方面的缺陷，我们结合 Blade 和 JUnit 为 Blade 添加了测试模块，向 Blade 提供了模拟请求和单元测试的可能。

8.2 数据统计

表 8.1 实验七数据统计

实验七：配置管理			
版本更新次数	配置管理报告累计字数	GitInfo 工具源码行数	累计工时
337	8741	1455 行源码 247 行注释	22.4 工时

8.2.1 提交次数随周数变化情况

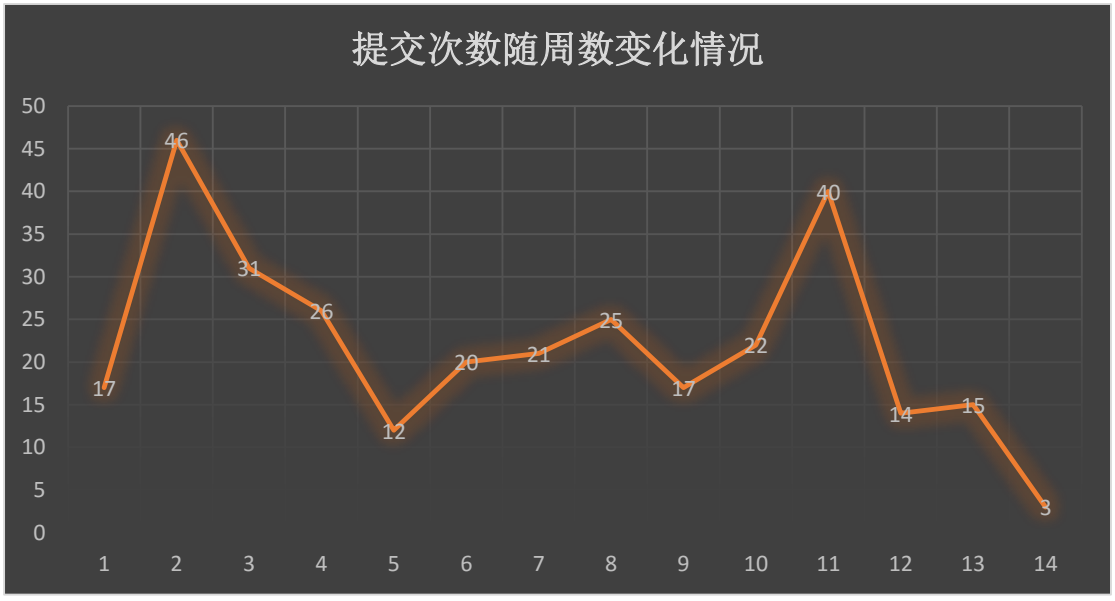


图 8.1 提交次数随周数变化情况

图 8.1 是组员提交的次数随周期变化的情况，从图中可以看出，在实验后期组员的提交次数少有下滑，而在需求和测试的部分显得较为活跃。分析认为，需求阶段提交频繁、测试阶段提交频繁的原因在于该阶段零散文件较多，提升了提交的次数。而后期提交的次数变少的原因在于，对于二进制文件的更新需要极为小心不能产生覆盖，因此均在本地完成后由组员统一提交。

8.2.2 提交次数随天数变化情况

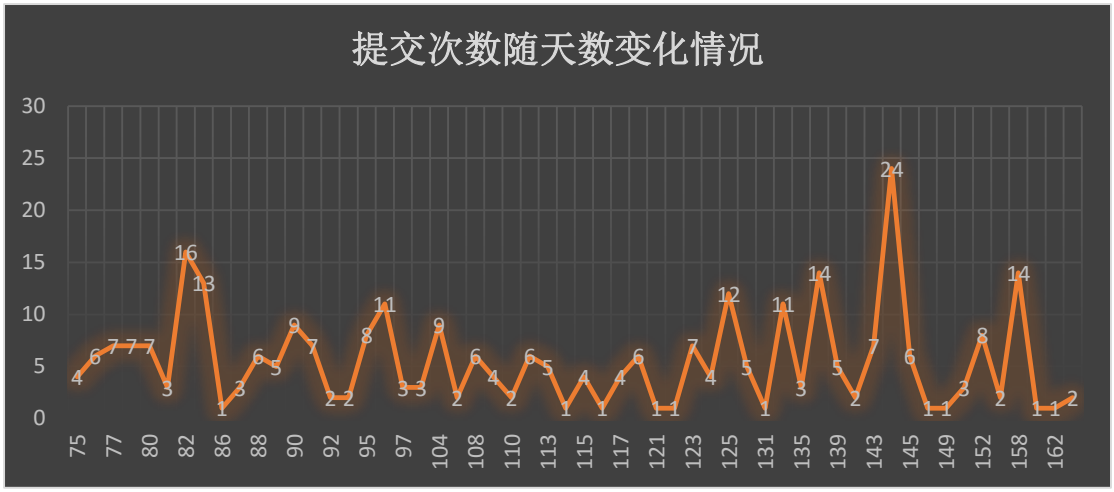


图 8.2 提交次数随天数变化情况

8.2.3 各个关键词提交次数

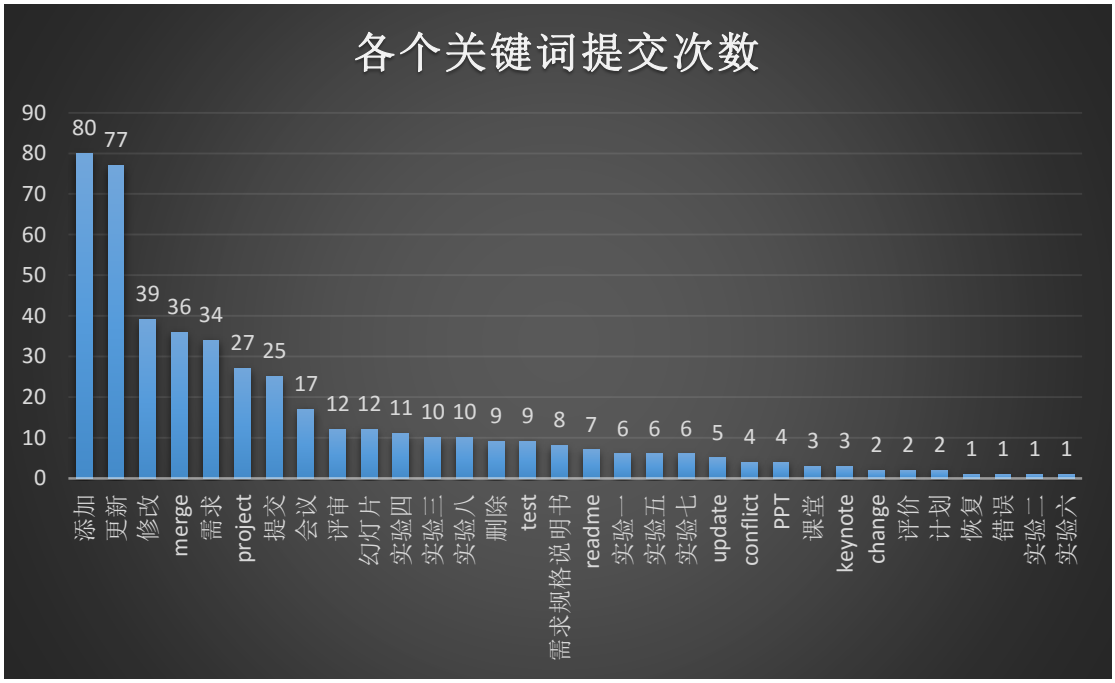


图 8.3 各个关键词提交次数

在关键词分析时，我们发现，在所有的关键词中【添加】、【更新】的出现频率最高，可见在平时的实验中，一直都有新的产物出现，并且对原有的产物也一直保持着更新。但仍然可以确认有些产物提交之后并没有修改行为。

在 merge 中出现了 36 次说明在同一个分支下出现提交不同步的概率较高，因此之后仍然应该在不同分支下进行提交。conflicat 出现了 4 次，说明对于任务的划分较为合理，之后的提交只需稍稍注意即可。

8.3 版本控制方法

8.3.1 Git 版本标识

本组的版本标识共分为两类：

- 提交注释

采用上节所示的关键词进行标记，表示是对文档的更新、添加或整理来表达文档的版本变化情况。

- 提交文档

本组对不同的文档命名方式略有不同。

1）预见会存在多版本的文档：采用后缀 vX.Y 的形式递增版本，若没有结构上发生变化，仅更新内容则仅变更小版本号，若发生结构性的变动，则更新大版本号。

2）与日期相关的文档：评审文件与评审日期相关度较大，因此对于评审文档采用后缀日期的形式进行版本标识。

3）与前一版本关系不大的文档：项目文档 mpp 文件、会议记录文件，每次版本均与上次内容关系较小，因此采用第 X 版的形式进行编号。

8.3.2 优点

1）较为准确的行为记录

在提交的过程中，我们小组使用了助记词汇进行提交注释，在提交后，能够较为准确的分析组员提交的内容、动作和所属实验。

2）较为完善的项目描述

在 Github 主页的 Readme 文档中，我们小组描述了系统的基本原理、Blade 的官方主页、项目涉及的文件类型的说明以及工具的使用说明。

除此之外，小组在 Readme 文档中提供了所有的文件的路径展示，因此，在查看中可以直接搜索到需要查看的文件所属的文件夹。

3）工具支持

采用了工具 GithubInfo 辅助分析提交信息，能够快速的分辨出组员的提交行为，并对提交的行为按日期、周数的分布情况进行的整理，有助于分析组员的提交行为。

8.3.3 缺点分析

1）提交与文件不能完全对应

我们在分析的过程中发现，提交了很多次，但文件、标题并不能反映出内容的更新。例如，实验六的项目计划文件的更新，在系统内记录有 27 次提交，但文件仅有 5 个版本。因此在版本管理中，虽然能够随时恢复至提交前后的状态，但无法直观的获得提交版本的更新、

内容的变化。应当单次提交只提交一个文档，并注明文档更新的内容，更新前后文件名变化和版本号变化。

2）文件归属的不统一

我们对于一个系列的文件采用了文件夹的形式存放，例如对于测试覆盖表，我们在实验文件夹下建立了新的文件夹用于存放，能够保证一次性浏览全部文件，也不至于因为文件名排序而混杂到其他文件中。但对于评审文档来说，我们采用文件夹的形式散落在各个实验文件夹下，虽然对应至各个实验的评审，但不能够一次性获得所有的评审文档，显得稍有紊乱。应当建立总的评审文件夹，下分实验一~8 文件夹存放评审文档。

8.4 总结与建议

在配置管理期间，由于组员均对 Git 工具和 Github 网站有较长的使用时间，因此在实验中，没有发生文档覆盖、无法解决冲突的情况。

采用了基于关键词的注释使得提交显得较为清晰，并且在提交期间，大多数提交均能够按照约定提交，能够在后期进行自动化分析。

在自动化分析的阶段，通过 GitInfo 节省了组员人工分析的过程，由于提交备注较为粗糙，无法获得足够的有效信息，但对于提交行为的分析依然有正向的帮助。

9 实验 8 工作量统计总结

9.1 工作量统计说明

本实验通过对实验项目的跟踪和记录，统计出每个成员的工作量和情况，分析成员工作量差异及其原因。为后续的任务分配、调整、控制和最终的确定成员贡献度提供支持。

9.2 数据统计

表 9.1 实验八数据统计

迭代版本数：	工具：	统计分析项数：	报告字数：	累计工时
4	1	17	6552	50

表 9.2 实验八制品信息

实验	序号	文件名	最后提交时间	最后提交人
四	1	工作量统计估计与分析 v1	2017/ 4 /18	刘晔
	2	工作量统计分析 0417	2017/4/19	汪晓燕
	3	工作量统计分析 519	2017/5/19	穆鹏飞
	4	工作量统计分析 0615	2017/6/15	刘晔
	5	工作量统计分析 0620	2017/6/20	刘晔
	6	计时工具 softTime-1.1	2017/ 4 /19	胡明昊
	合计	5 个文件+1 个工具		

9.3 数据分析

实验八的工作量统计对实验项目从实验一到实验八的实验过程中，小组成员所产生的工作记录以及制品数据等做了较为详尽的收集，分析和总结，得到了各个实验的工作量数据以及各个实验小组各成员的工作量情况，最后，汇总得出总的项目的工作量数据和小组各成员的工作量占比情况。从表 9.2 中的 5 个文件加 1 个工具中能够看出，各成员积极参与本次实验，持续的更新工作量统计报告。比较好的印证了表 9.1 中的工作量统计情况。

在实验八的统计过程中，分别从三个不同的方面以及两个不同的时间段来进行统计工作。三个不同的方面分别指的是：

1） 分实验工作量估计

2） 计时工具统计

3） github 实验详细统计

两个不同的阶段分别是：

1） 4 月 20 日之前的时间段

2） 4 月 20 日至 6 月 12 日这一时间段

特别说明的是，时间阶段选取两个是有原因的，因为 4 月 20 日之前，使用的是 mpp 文件，而初期 mpp 文件有一定的缺陷，对工作量估计误差比较大。4 月 20 日以后，使用小组胡明昊同学的计时工具完成了对工作记录的方便记录和追踪。这非常有利于准确统计小组成员的工作量。另一点需要说明，由于实验六七八最后同期进行，所以六七八的工时需要额外加入整个项目总工时的计算中去。

9.4 总结及有效方法建议

工作量统计是软件项目开发中的重要一环，能够帮助项目团队主管了解当前的软件开发工时，最重要的是能够决定着项目内各成员的贡献率分配和项目的总工时估计，这对调节控制软件开发成本，以及之后的类似项目报价而言非常重要。

有效方法建议。工作量和贡献率之间关系不确定。需要注意的是，虽然实验八采用的数据是真实而且可靠的，数据能在一定程度上反映各成员的贡献率，但是由于小组各成员的分工不同，任务和人的特性有所区别。所以在计算最后的贡献率时，小组需要达成共识，给出一致认可的贡献率占比。