综合实验总结报告

小组成员：

刘少凡

宋昱材

吴沂楠

黄飞

版本变更记录

版本	变更时间	修改人	审核人	备注
1.0	20170621	刘少凡 宋昱材 吴沂楠 黄飞	刘少凡 宋昱材 吴沂楠 黄飞	初稿

目录

1 前言

1.1 项目概述

本次软件工程综合实验过程在遵循软件工程开发过程的基础上，进行了适当的调整和创新，具体过程包括：软件项目计划阶段、软件需求分析阶段、软件需求评审阶段、软件产品改进与展示阶段、软件测试阶段、软件测试评审阶段。除此之外，还进行了软件进度计划与控制、工作量估计与统计分析、配置管理这些工作，贯穿在整个项目执行的过程之中。

具体到本组的话，本组项目名为：基于 Lire 的分析与扩展，即通过对 Lire 的学习与理解，对 Lire 进行业务需求、功能需求和非功能性需求等分析，在需求分析的基础上，本项目扩展了 Lire 支持的特征算法，并基于此开发了一个简单的 CBIR 系统，最后对 Lire 的各种需求与开发的 CBIR 系统的功能与性能进行了测试分析。

1.2 文档概述

接下来的内容为对本项目的总结报告，主要内容有：

第二章基于数据统计，说明各个实验的工作量；

第三章对实验 6~8 统计分析数据的完整性和准确性进行说明；

第四章对软件需求、测试需求和改进与扩展的质量水平进行说明；

第五章对两个有效方法进行详细说明；

第六章总结经验并提出建议。

2 工作量统计

本章进行工作量统计，并做简单的说明，详细的说明分析请见《工作量统计分析》文档。

2.1 需求分析

需求文档规模（字数）	需求项数（RUCM 用例描述数量）	其他模型（类图、顺序图、状态图等，不含非规范的示意图）	其他模型中包含的元素累积数（节点和边的数量）	版本更新次数	累积工时（含需求修改工时）
7055	14	框架图	18	12	53
		用例图	20		
		模块结构图	16		

从版本修改次数上来看，针对于需求文档进行了反复的修改，文档的质量有一定的保证；所有的需求都使用了 RUCM 模型来描述，并使用了其他的一些模型来描述系统的运行流程与功能，文档的可读性较强。

另外，累计工时为 53 工时，可以看出在需求分析阶段做的工作比较细致，这也是使得后面各个阶段返工较少的原因。

2.2 需求评审

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）	接收到的问题数（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	累积工时（不含需求修改工时）
21	70	23	27	43	4394	15

从给其他两个组评审的数量共有 70 条，说明我组在评审其他组的需求文档的过程中还是较认真负责的。

互评审和老师共给出了 50 个意见，我组修改了 43 条，修改率为 86%。

2.3 改进与扩展

设计实现规模（字数）	其他模型（类图、顺序图、状态图等，不含非规范的示意图）	其他模型中包含的元素累积数（节点和边的数量）	代码行数	问题数	修改的问题数	版本更新（最大）次数	累积工时数
2849	类图	38	1610	3	3	5	78.5
	顺序图	38					
	流程图	13					

从类图、顺序图、流程图等模型的使用数量上来看，我组使用了比较多的图来描述代码的设计，文档质量较高。

代码行数和工时适中，说明项目具有一定的复杂度。

2.4 测试需求

测试需求和测试报告等文档规模（字数）	测试需求用例数	测试数据（实例）数	测试类型	覆盖率	测试需求文档更新次数	累积工时（含需求修改工时）
8254	25	25	功能需求	100%	4	38.5
			非功能需求	100%		

测试用例比需求分析阶段的 RUCM 数要多一倍左右，说明测试较为充分。所有的需求都使用了 RTCM 模型来描述，文档较为规范。

另外，累计工时为 38.5 工时，可以看出在测试需求分析阶段做的工作也比较细致，这也是使得后面各个阶段返工较少的原因。

2.5 测试评审

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）	接收到的问题数（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	累积工时（不含测试需求修改工时）
15	30	15	12	17	3989	9

从给其他两个组评审的数量共有 30 条，说明我组在评审其他组的测试需求文档的过程中还是较认真负责的。

互评审和老师共给出了 27 个意见，我组修改了 17 条，修改率为 63%。

2.6 项目计划

项目计划书、小组会议记录和周记等累积字数	项目计划表（MPP）中分解的任务项数（含组合任务）	项目计划耗费的实际工时数	项目计划（MPP）更新次数	项目的累积实际工时数
13905	235	30.5（项目计划 mpp）+9（项目计划书）	15	637.4

从项目计划的更新来看，我组是每周更新一次，更新频率较为合理。

2.7 配置管理

版本更新此数（提交次数）	配置管理报告等累积字数	累积工时
192	5933	20.4

可以看到，提交次数为 192，平均每周每人提交 3.2 次，数量较为合理。

2.8 统计分析

统计分析项数	统计分析报告字数	累积工时
23	18872	35.5

统计分析项数共有 23 项，统计和分析项目较为具体；从统计文档的总字数也可看出对于实验各阶段的统计比较详细，数据较齐全。

3 实验 6~8 分析数据说明

对实验 6~8 的总结分析过程中，涉及到了许多不同类型的数据，下面将对这些数据进行完整性和准确性的说明。

3.1 实验 6

实验 6 的总结分析文档为《进度计划与控制分析报告》，其中第 2 章涉及到《项目计划》文档的修改日期、版本号、变更情况、变更原因与累计消耗时间，第 4 章涉及到各项任务的计划工时与实际工时数据，第 5 章涉及到成员的总工时数据。

其中文档的修改日期、版本号、变更情况、变更原因和计划工时数据都是完整并且准确的，虽然在项目的进行过程中《项目计划(v4.0)》以前的文档被覆盖掉了没有保留，但是因为数据记录的及时，中间间隔的版本号数量以及它们的修改日期都是完整并且准确的。

累计消耗时间、实际工时和成员总工时数据，其实都属于实际工时的范围，该数据来源于小组各成员每周提交的工作日志，数据具有一定的主观性，并不十分准确，由于本组从项目初期就开始进行了每周工作日志的记录，因此数据的完整性可以保证，从始至终的实际工时都有对应的工作日志作为数据来源。虽然该项数据很难进行查证，但是出于对小组成员的信任并考虑到数据的性质，预计该数据的偏差不超过 10%。

3.2 实验 7

实验 7 的总结分析文档为《配置管理总结与分析 v13.0》，其中涉及到的数据有本组在 Github 上的 Commit 数量、有效 Commit 数量和每个 Commit 的时间与内容。

其中 Commit 数量和每个 Commit 的时间与内容均为完整并准确的，因为这些在 Github 上都有记录。

有效 Commit 属于本组提出的一个概念，主要指的是较为重要的更新，例如对于其他组员提交内容进行审查过程中发现的诸如格式、错别字之类的修改不属于有效 Commit。由该定义也可以看出，对于一个 Commit 是否属于有效 Commit

的判断是具有主观性的，因此该项数据并不一定很准确，但是由于该项数据的统计前后只是两个同学在做，基本可以维持判断标准的前后一致性，所以预计该数据的偏差不超过 5%。

3.3 实验 8

实验 8 的总结分析文档包括《实现与需求和测试的变更分析》、《需求文档修改和测试文档修改与问题报告的关系分析 V2.0》和《工作量统计分析 v8.0》，下面分别对这三个文档中涉及到的数据进行说明。

《实现与需求和测试的变更分析》涉及到的数据主要是修改日期、修改内容和变更原因，数据来源为平时的记录与项目计划 mpp 文件，数据完整且准确。

《需求文档修改和测试文档修改与问题报告的关系分析 V2.0》涉及到的数据主要是需求文档的版本记录、变更的内容、变更中内因和外因的数量，数据来源为平时的记录、FG 组对我组需求说明书的问题清单和老师的批注文档，数据完整且准确。

《工作量统计分析 v8.0》中涉及到的数据较多，第三章有各文档字数、小组各成员在该文档中完成的字数、图表个数以及耗时；各文档各版本修改日期、版本号，修改内容与修改人；各阶段评审的意见个数与修改个数；代码行数与类个数。第四章除了已经在第三章中包含的数据外，主要有各文档的累积变更量和代码难度数据。

其中各文档字数来源于 word 左下角的统计字数，完整且准确；图表个数是从对应文档中数出来的，也是完整且准确的；各文档各版本修改日期、版本号，修改内容与修改人是及时记录的，而且大部分都有前后文档的对比，是完整且准确的；各阶段评审的意见个数和修改个数来源于问题清单和老师批注，是完整且准确的。

小组各成员在文档中完成的字数和各文档的累积变更量是不太准确的，因为本组在文档迭代的过程中没有记录该次修改具体删除了多少字、修改了多少字、新增了多少字，因此这两个数据都只是一个大概的估算，预计该数据的偏差大约为 15%~25%。代码行数和类个数来源于源代码，但是由于本组代码是对 Lire 源码进行的修改与新增，其中新增的代码数据较易统计，而修改的代码数据较难统

计，因此该数据具有一定的不准确性，不过因为主要进行的还是代码的增添，只是对修改的代码数据进行了估计，所以预计该项数据的偏差不超过 5%。代码难度数据因为具有一定的主观性，因此也是不太准确的。

3.4 总结

通过以上三个小节的说明，可以看到虽然在实验 6~8 中涉及到的数据种类和数量较多，但是其中大部分数据具有或客观或主观的来源，数据的完整性有所保障。客观数据的准确性较高，而一些主观的数据可能不太准确，比如代码难度和有效 Commit 数量；还有另外一些数据难以查证，所以也可能存在准确性问题，比如实际工时；剩下还有一些在变更过程中未记录，导致只能进行估计推算的数据，比如各文档的累积变更量。

对于以上这些可能存在准确性问题的数据，本文档通过简要的分析给出了其中大部分的预期偏差值，可以看到虽然这些数据可能存在准确性问题，但是其偏差并不会特别大。

4 主要制品的质量水平说明

本章主要对于实验 1 和实验 7，即软件需求和配置管理方面的主要制品进行质量水平说明。

4.1 软件需求

4.1.1 软件需求概述

软件需求包括软件的业务需求，软件的功能需求，软件的非功能性需求。

在软件的业务需求中，Lire 作为一个工具包为构建 CBIR 系统提供功能支持。为满足该需求：

(1) Lire 工具包提供了尽可能多的当前主流的图像特征提取算法，并提供简洁的调用接口，供 CBIR 系统开发者进行选择；

(2) Lire 工具包提供了一种高效的数据库存储方式，能够快速高效地进行存储、读取和检索。

(3) Lire 工具包提供了适应多种图像特征的图像入库算法，并且提供尽可能多的当前主流的索引生成算法；

(4) Lire 工具包提供了尽可能多的主流图像特征距离计算方法，提供高效、准确的多种排序算法，并将图像检索功能封装为简洁的调用接口，供开发人员快速实现不同的图像检索功能模块。

在软件的功能需求中，由于 Lire 主要面向的是 CBIR 系统的开发，因此 Lire 需要满足：

(1) 开发人员可以利用 Lire 编写图像入库的程序

(2) 开发人员可以利用 Lire 编写全局特征索引构造的程序

(3) 开发人员可以利用 Lire 编写局部特征索引构造的程序

(4) 开发人员可以利用 Lire 编写混合特征索引构造的程序

(5) 开发人员可以利用 Lire 编写图像检索的程序

(6) 开发人员可以利用 Lire 编写图像特征提取的程序

(7) 开发人员可以利用 Lire 编写特征距离计算的程序

为了全面地说明功能需求，还应对 CBIR 系统的功能需求进行说明，CBIR 系统的功能需求主要有：

(1) 利用 CBIR 系统进行图像入库

(2) 利用 CBIR 系统进行图像检索

在软件的非功能性需求中，Lire 应该具有比较强大的兼容性。为满足该需求：

(1) 基于 Lire 开发的应用程序能运行于不同的操作系统（如 Windows，Mac OS X 和 Linux）上

(2) Lire 提取图像各类特征之后生成的索引数据能以跟操作系统和运行环境无关的方式储存，使不同平台上能够共享生成的图像数据库

(3) 作为一种完全开源的框架，Lire 的代码对开发者完全透明

(4) Lire 能够高效实现大规模数据索引入库

(5) Lire 能够迅速响应查询条件并返回结果

4.1.2 质量水平分析与说明

需求规格说明书从最初的初稿到最终版总共经历了 11 次或大或小的改版，所有章节内容都至少修改过一次。对于重点的业务需求进行过 4 次修改，功能需求进行过 6 次修改，非功能需求进行过 5 次修改。可以看到总修改次数和对于重点内容的修改次数均较多。另外对于语言描述问题进行修改的次数有 5 次。可见对于各项内容的描述语言也是得到了较为充分的修改的。

需求规格说明书的各历史版本总共收到了来自老师和 F\G 组同学的共 51 个问题，我们对其中所有问题都进行了讨论与分析，修改了其中的 44 个问题，对于剩下的 7 个问题也进行了较为详细的解释。所有的问题都得到了处理。

通过以上分析，我们认为需求规格说明书在语言描述的准确性和业务需求、功能需求、非功能需求分析的完整性和准确性上都有了较大改进，并且老师和其他同学提出的各种问题都进行了处理，没有遗漏。总的来说，需求规格说明书质量较高。

4.2 配置管理

4.2.1 配置管理方法概述

配置管理使用的主要工具是 Github，管理方法主要包括以下几个方面：

在文档分类组织结构方面，我们依据本次软件工程综合实验八个子实验的任务要求建立八个子文件夹，用于存放各子实验阶段对应的产出物。

在文档命名方法方面，我们将软件工程综合实验的各文档命名由组号、配置项名称、迭代号/版本号三部分组成。

在更新说明方面，我们对文档的每次版本更新，都在文档内附上较为详细的版本更新记录表，表中的字段依次为：版本号、变更时间、修改人、审核人、备注。

在更新频率方面，随着实验的推进，我们逐渐形成了小的修改通过微信群传递，汇总之后再上传到 Github 的方式，以此来避免频繁更新造成版本混乱的问题。

在历史版本的处理方面，我们将各文档的历史版本保留在了单独的文件夹中，便于进行追踪和分析。

4.2.2 质量水平分析与说明

通过上节对于配置管理方法的总结概述，我们认为，本组 Github 项目具有比较清晰的目录，且各文档名称形式较为统一，便于区分。

由于本组采取小的修改通过微信群传递，汇总之后再上传到 Github 的方式，本组 Github 的有效 Commit 比例均较高，且随着试验阶段的推进，该比例稳中有升，具体可见《配置管理总结与分析》。我们认为，这种方式可以有效避免频繁更新造成的版本混乱问题。

本组对每个重要文档都在文档内部记录有较为详细的版本更新记录表，且本组将所有文档的历史版本保留在了单独的文件夹中（最早期的几周内容被覆盖了没有保留）。我们认为，本组对历史信息的记录较为完善，便于进行追踪与分析。

总的来说，我们认为本组在配置管理这方面做的较为出色，文档目录结构清

晰，内容完整。

5 有效方法详细说明

5.1 软件需求分析

5.1.1 如何确定软件的需求

（1）从基于该项目的应用开发需求反推

本组分析的项目 Lire 是一个 CBIR 系统的开发工具包，其功能是为 CBIR 系统的开发者提供快速搭建 CBIR 系统的相应接口。因此，从 CBIR 系统的需求出发，可以反推出 Lire 项目的功能需求。

（2）从项目文档中提取需求

开源项目提供的文档内容会体现项目所提供的功能、使用项目的方法、项目的体系结构，从这些资料中，可以一定程度提取出项目的需求。

（3）从项目使用样例中提取功能

项目的使用样例代码示范了使用项目进行各种功能开发的编码方式，从样例代码中可以相对准确地提取出一部分项目的功能。

（4）从代码模块结构中提取需求

在完成以上环节后，组员基本对项目有了相当程度地了解，可以进一步阅读项目的源代码，源代码的模块结构体现了项目的体系结构，从中可以提取出项目的模块结构，进一步提取项目各模块的功能，最终提取出需求。

（5）从源代码中核实需求

最后，通过阅读详细的源代码，完成对需求的核实，进一步，基于源代码完成对需求用例 RUCM 模型的描述。

5.1.2 如何描述业务需求

本组在描述业务需求之前，针对业务需求查阅了相关资料，发现各类资料对业务需求的定义不尽相同，并没有统一的标准。通过在课上与老师沟通后，结合组员的思考，确定了本组进行业务需求描述的内容。

Lire 项目视为 CBIR 的系统开发服务的，因此业务需求的描述从 CBIR 系统

的框架、模块结构出发，描述 CBIR 系统的需求。进一步，基于 CBIR 系统的需求，分析 CBIR 开发人员对于开发辅助工具包的需求，从中提取最主要、最上层的需求，作为 Lire 的业务需求进行描述。

5.2 软件需求评审

5.2.1 如何准确理解老师提出的问题

在进行需求评审的答辩时，老师会提出很多关键性的问题。但往往组员在面对老师提出的问题时，容易出现理解不清，解决不到位的情况。

一方面，需求分析者在进行需求分析时会建立自己的一套逻辑，也容易陷入自己的逻辑中，不容易接受评审者的意见和要求，因此在需求分析的过程中要多自省，积极地从自身发现问题。

一方面，由于老师拥有丰富和深厚的软件工程知识和经验，因此往往提出的问题会更深刻、更关键。但由于组员们在软件工程的经验很浅、知识不足，容易出现对老师的问题理解不清。因此组员需要及时与老师沟通，主动提出自己的问题，得到老师进一步的指导，也要积极补充软件工程的相关知识，帮助自己更深入的理解老师提出的问题。

5.2.2 如何对同伴组的需求文档进行评审

在需求评审阶段，本组组员要对同伴组的需求规格说明书进行评审，提出问题和意见。在这个阶段，本组组员会面临对同伴组的项目了解较少，评审时无从下手的问题。

作为需求评审者，完全可以站在无任何领域先验知识的立场去阅读别组的需求规格说明书，在阅读中出现的任何影响阅读和理解的用例、描述都可以进一步总结为你认为的问题的所在。

5.2.3 如何处理同伴组对本组的评审意见

本组对同伴组评审意见的处理方式主要为接受和解释。

对于确切的问题，本组的处理方式是接受并进行修改；对于同伴组理解有误产生的意见，本组会对同伴组进行解释和说明。

5.3 改进与展示

5.3.1 如何确定实现部分的工作内容

确定实现部分的工作内容这一问题，应当在需求分析阶段就开始考虑，在需求分析的过程中，在对项目的理解一点点深入的过程中，发现可改进或者可扩展的工作点。

本组在需求分析阶段，在分析图像特征提取的功能实现部分时，发现该部分的代码实现具有很强的可扩展性，具有很明确的扩展框架支持开发人员引入新的图像特征提取方式。因此，本组组员经过讨论后决定在 Lire 中引入深度学习模型的特征。

基于确定的工作目标，再进一步确定可行的实现方案，并将实现方案中的工作内容进行有效合理的分工，分配到各个组员。

5.3.2 如何进行实现的展示

为了方便实现内容的展示，本组在确定实现工作内容时，将实现的目标制品确定为一个简单的 CBIR 系统，系统基于深度学习模型提取图像特征。为了达到 CBIR 系统最好的展示效果，图像界面是必不可少的，因此我们在实现内容中加入了图像界面的编写，以达到最好的展示效果。

在实际的课上演示中，我们通过远程连接的方式连接组员的个人电脑，通过操作个人电脑的虚拟机中的软件，展示效果。

5.4 测试需求分析

5.4.1 如何设计测试用例

测试应当至少完成两方面的工作。一方面，针对需求规模说明书中的需求用

例进行测试；一方面，针对实现阶段的实现内容进行测试。

针对需求规格说明书中的需求用例，针对每个需求用例针对性地设计相应的测试用例。

针对实现阶段的实现内容，对实现部分相应的各个功能设计了测试用例。

5.4.2 如何进行测试用例的描述

本组通过浏览往届各组的测试需求规格说明书，发现各组的测试用例描述模型的大体框架一致，但具体内容各有不同。

本组综合考虑往届各组的测试用例描述模型，汲取模型中有效的部分，舍弃掉不适用于本组的部分，设计了本组的测试用例描述模型。

5.5 测试评审

5.5.1 如何对同伴组进行测试评审

本组对同伴组进行测试评审的思路和进行本组测试需求分析的思路是一致的，从需求文档出发，针对需求文档中的需求用例，检查测试文档对需求用例的覆盖程度。

对某个测试用例，检查描述的规范性和一致性，检查测试数据设计的合理性和充分性。

5.5.2 如何完成同伴组的实际测试

本组需要对 D 组的项目进行实际的测试，由于 D 组项目对设备和环境的要求，本组由组长与 D 组组长协商后，使用 D 组组长的设备进行测试。在完成 D 组原有测试的基础上，针对本组在评审时发现的可能问题设计了本组的测试用例进行测试。

5.6 进度计划与控制

5.6.1 如何有效控制项目按照计划执行

本组在周初进行会议讨论时，会针对本周工作进行详细的讨论而不是简单的分工，对各部分工作进行详细的讨论并确定具体工作方案，然后分配到个人。这样可以在进行计划之后，在实际的完成中不会出现较大地变化，可以保证计划有效地执行。

5.7 配置管理

5.7.1 如何合理配置目录结构

在实验的前期，本组主要参考往届组中较好的目录结构设计本组的目录结构。在实验过程中，本组会针对使用目录的经验和问题调整目录结果，最终形成了当前的目录结构。

5.7.2 如何处理历史文档

本组在实验初期对部分历史版本文档进行了覆盖的操作，造成了部分文档的丢失，而在最后的分析阶段，这等于丢失了部分可分析数据。

因此，对于历史版本的文档应当集中归纳保留，通过版本号进行区分，不要覆盖。

5.8 工作量估计与统计分析

5.8.1 如何对数据进行有效的统计

本组参考了往届组通过工作日志记录工时的方法。本组组员每周进行工作日志的编写，工作日志中记录本周的各项工作内容、工时等信息，负责进度计划与控制的组员根据工作日志更新 MPP 文件。

本组在进行版本变更分析时遇到一个问题，在各个文档的版本变更时，版本

变更表中对变更内容的记录不够详细，有些记录太概括，在分析变更项时无法区分。因此，建议在进行版本变更时，详细记录各个变更项，便于后期的统计分析。

6 总结与建议

该课程的整体设计较为新颖，通过将需求分析、设计实现与测试这三个软件开发过程中最重要的部分串联起来，文档与编程并重，加上评审任务与每周的讨论，使得学生了解软件工程的整个流程，并且亲身实践，很有意义。

很感谢老师在这一整个学期的课程实践过程中对我们的悉心指导，对我们提出了很多很有帮助的建议。

建议老师对于每周的任务与要求尽量明确一些，可以节省掉一些不必要的修改与返工，不过如果出于“犯错之后再改会更有记性”的考虑出发，也可以保持现在这种出了错再改的情况，不过可能会不利于学生的积极性。