图像处理系统
个人报告

林悦如

2018-08-23

项目时间	:	2018-07-12 - 2018-07-23	
项目参与人	:	3160104062	刘涵宇
		3160102459	林宇翔
		3140103548	林悦如（组长）
指导人	:	浙江大学	袁昕
项目主页	:	SummerCPP/segprototype/	

目录

1 个人工作

1.1 项目简介

在本次段学期课程中，我们小组决定开发一个基于opencv的图像处理应用。项目的初期目标是实现一个能够导入图片，导出处理结果，记录处理历史，支持传统的图像处理功能的系统。
课程要求系统采用MVC，或MVVM架构；

1.2 我的工作

本次开发中, 我的职责如下：

- 作为组长，负责安排其他组员的任务；根据组员的水平选择开发工具，制定开发计划。
- 作为项目经理，负责督促项目的开发进度；确定每轮迭代系统功能增量的大小。
- 作为架构师，负责架构代码的编写，集成测试等事情。

我们团队的开发工作分为以下五类：

- View层的开发，也就是图形界面的开发。
- Core层的开发，也就是算法层的开发。
- Model层的开发，负责数据模型以及业务逻辑的开发。
- ViewModel 的开发，负责视图逻辑的实现以及模型数据的封装。
- App层的开发，负责系统的集成。

App层的工作固定由我负责。Core层，由三人同时进行开发。由于图像处理功能的算法之间相互独立，所以在三人负责不同的算法的前提下，就算提交到同一个github库，也不会产生冲突。MVVM的三个模块分别由不同的人负责。这就是我作为组长对组员的任务分配的主要依据。

本项目基于opencv, Qt5, 以C++作为开发语言，以Cmake作为模块集成工具，以Qmake+QtCreator作为系统集成工具，进行开发。
由于，对opencv和Qt和MVVM框架都不很熟悉，我们只进行了3次完整的迭代。这几次迭代中我负责的开发工作依次，分别是：

a. App + ViewModel + Core

b. App + View + Core

c. App + Model +Core

具体细节见下一个章节。

1.3 具体细节

1.3.1 初期工作

测试了GithubTeam，VSTS（VisualStudioTeamService)等集成构建和团队协作工具， 选择Github Public Repository+Git作为最终方案。
选题。
搭建开发环境。
制定协作策略。

1.3.2 第一轮迭代

安排组员的开发工作。
写ViewModel层的代码。
重构组员提交的代码，使其大致符合MVVM架构。
写App层的代码，实现ViewModel和View层的数据绑定。
写Qmake脚本，利用QtCreater，对系统进行集成测试，最后成功构建了系统原型。
在Github上发布了系统原型。

1.3.3 第二轮迭代

安排组员开发新的图像处理算法，并要求他们的代码需要符合统一的接口要求。
开发View层，为不同的图像处理算法设计图形界面。
修改App层的代码，添加新的数据绑定（由于功能增多引入的数据绑定）。
修改Qmake脚本，在QtCreator中集成测试，构建。
在Github上发布。

1.3.4 第三轮迭代

转入Model层的开发，添加业务逻辑。
检查数据模型的生命周期。
检查数据模型的内存分配策略。
对Model层进行单元测试。
改App层代码。
对系统进行集成测试，构建，发布。

1.3.5 后期事项

进行验收和答辩。

2 总结与反思

2.1 总结

通过本次短学期， 我在敏捷开发和团队协作方面积累了宝贵的经验, 比如

- 分配任务时要尽量具体
- 不要事事亲为
- 敏捷开发的核心是可以执行的软件
- 好的架构是团队合作的决定因素之一

2.2 反思

本次开发我们组在MVVM架构的开发上存在重大的瑕疵；我们在实现ViewModel和View层的数据绑定时，没有在真正意义上将ViewModel的属性暴露出来，实现 真正的数据绑定，而是用Qt的Signal/Signal的机制模仿了这套机制的数据同步功能, 实现了一个伪数据绑定。这样, 不仅代码混乱难以维护，而且使得View层对暴露的属性的选择变得死板，而且ViewModel和View之间的耦合度变得很大。 这样实现的系统的可拓展性很低，
完全浪费了MVVM框架提供的好处。

出现这样严重的问题，一方面由于我们对MVVM架构的认识十分浅薄，并不理解
架构背后的原理。一方面由于我们的确被Qt的某些方便的机制所迷惑，目光变得短浅，只追求单一功能的实现。

3 课程建议

对于这门课程，我有一下几点建议：

- 建立一个QQ群。
- 明确要求各组在前4天内就提交一个系统原型发布，并且要求实现MVC或者MVVM架构，尽早发现架构上的问题，让后期的快速迭代成为可能。
- 明确要求各组在后期，每天都要集成测试，并且发布可执行文件。