2023大作业常见问题汇总

2023年12月26日 10:46 JunFang

这是去年的大作业点评，注意当时有些规定与今年有差别。

这几天我集中改大作业，已改完。虽然大作业只占总成绩10分（2024年占20分），但它所体现的各种能力和训练，比如文档写作，软件工程文档的结构组织，绘图，前后一致性保持等能力，会对大家以后的学习产生深远影响，大家应重视。本文档总结了本次大作业大家常犯的错误，文档较长，希望大家耐心阅读，引以为戒！

第一部分：所有环节（需求、概要设计、详细设计、测试）都有的共同性问题

1. 模板问题。模板质量不是很高（2024年的模板已做了一些修改），导致大家有些地方被模板所误导。虽然我已在《大作业要求（先看我）-2023版》中提醒大家不要被该模板牵着走，但不少同学还是硬套。

   - 《软件项目报告模板》最早是由邱元杰老师提供，我做了一些精简。该文档是一个通用的报告模板，结构比较完整（还包含硬件部分的描述），但有些庞大和细碎。另外该模板采用了较多的结构化分析和设计思想，对面向对象开发思想的体现不是很足。同学们可根据自己对“成绩管理系统”的认识，自行裁剪和调整该模板的内容（有什么就写什么，没用到的模板中的章节要删掉；还可以调整条目的顺序，修改条目的标题，或增加条目。对模板说明性的文字要删掉）。↓

   我小改了去年的模板，例如概要设计中我删掉了“关键处理过程或者算法的设计”（其实本节应放到详细设计）、“系统出错处理设计”和“维护处理过程”三节；软件测试中我删除了“路径测试的检查表”一节。并且有些小节调整了位置。但有的同学没用新的模板，还是用去年的老模板，让人怀疑是否抄袭了去年同学的或其他班上的大作业，要扣点分。
2. ChatGPT滥用问题。这是今年的新情况。我说了，不是不能用，而是不能乱用，要用的好、用的巧，知道什么地方能用，什么地方不能用。有同学认为chatgpt用了不容易看出来，其实有一些使用经历后还是挺容易看出来。我知道每个组其实都或多或少的使用了，用得好的组能够和自己写的内容浑然天成，有加分效果；用得不好的则显得言之无物，混乱随意，反而减分了。大概有五、六组或多或少存在滥用的情况。

   去年没有ChatGPT时，大致呈现出字数与质量成正比的情况，最高分4万多字。但今年不是了！最高质量的文档反倒在25000字左右，而更高字数的有些就是用ChatGPT通篇凑数了。
3. 详细设计描述重要方法成员的算法时，我对于用伪代码或形式化描述语言来表现的，不如用

图表（例如流程图、顺序图）来表现的给分高。因为伪代码较容易用ChatGPT生成。大家可以用ChatGPT生成伪代码或解释来辅助理解是可以的，但最后在文档中宜用流程图、顺序图、判定表等图表的形式来表现。至于AI生成图表，可能用最新的ChatGPT或其他LLM（大语言模型）也能搞定，但我没去详细了解，至少比生成文字要麻烦点吧。

4. 需求->概要设计->详细设计三大环节的“递进”问题。有些组写的内容或画的图张冠李戴，把原本属于这部分的写到那部分，还有的重复表现。例如，类图和顺序图，其实三个环节都可以用，但每个环节的画法和内容划分粒度是有区别的，应体现出层层递进。

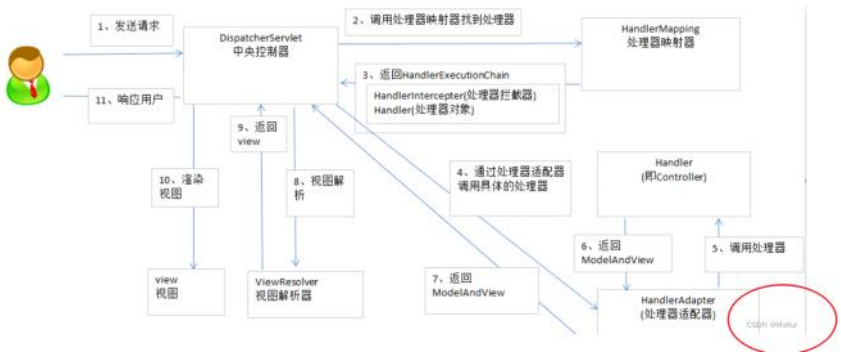
   而很多组没有体现出这种递进，每个环节的类图和顺序图的画法或粒度都差不多，只是选取的角色或功能模块不一样。这种属于简单的重复，没有体现“渐进性”。甚至一图多用，重复粘贴。还有的甚至出现“倒退”，概要设计的类图比详细设计还要详细！这体现出小组内部没有整体筹划沟通好，组稿后也没有仔细通篇审核。

   比如类图，可以从形态（分析型类图还是设计型类图）、类的数目，类划分的细粒度，类的属性和方法的完善程度，类之间关系的完善程度等方面体现不同阶段。

5. 有些组偷懒，每个环节画2个图示例一下就完了（那两幅图还是我课件里的），这样工作量就不够。其实《大作业要求（先看我）-2023版》已经给大家缩小范围了。这个大作业是让大家形成完整文档给程序员使用的，不是编教材，讲究的是完整性。编教材倒可以找两幅图示意一下就行。

6. 用例图和类图可以画一幅大的描述总体的图；活动图和顺序图不要画总体图，画也画不好。即使用例图和类图都应该总体图、部分图搭配着来表现才好，否则会导致图太大，字太小，可读性很差。比如，类图在需求、概要设计阶段多表现为总体但内部简单的类图，详细设计多用部分但内部复杂的类图。

7. 有同学的图片盗用网络上的，水印都在（下面画圈处）。有些图的风格也不是starUML画的，一看就像是网上找的。



8. 多个组提到本系统“采用C/S架构（客户端/服务器）”，还像模像样的论述一番。其实本系统就一简单的单机版结构，后端没有服务器。只需一个证据就能判断，C/S架构可以多用户并发访问和控制，你对这个系统能吗？

9. 图文结合。一份优秀的文档，应是图文结合、图文不分离，相互印证。不管是用例图、活动图、类图、顺序图或其他什么图，都应在每幅图下面配上文字说明，而不是孤零零地把图扔在那里就算完事。

第二部分：各个环节的一些具体问题

一、需求分析部分：

1. 《报告模板》中大量使用“接口”，大家对它的理解有很大偏差。“接口”是计算机行业最被广为混用的概念，在不同的语境或不同的环节下有不同的含义。比如
   a. 需求分析里的接口包括：用例图体现的接口，即参与者和某个用例之间的连接，表示系统为参与者提供的某项服务；“用户接口（UI，或用户界面）”，指用户怎么和系统交互，包括人机界面，给操作员的操作指令、外部数据输入输出等；如果系统涉及硬件，还有“硬件接口”，硬件怎么采集数据，怎么和计算机通信等。
   b. 概要设计的接口主要描述在某种体系结构下，子系统、模块、构件之间的接口，描述每个功能模块之间怎么传数据、怎么交互。例如，若软件采用层次体系结构，那么接口包括各层之间，底层与操作系统之间的请求调用关系和数据通信等。如果采用MVC体系结构，包括前端和后端怎么传数据，数据库的规格和约定等。
   c. 详细设计里的接口主要在类和对象的层次，它既可以是一种面向对象语言中的结构（如Java中的Interface），也可以指“供对象间调用的接口”（即方法签名）。
2. 需求分析第三节外部接口需求中的“用户界面”，不要直接截取软件最终界面图和运行时的界面图，而是要采用文字、流程或草图（sketch）的形式来描述（草图不是运行时的界面截图，草图的控件里没有具体内容）。其实模板里也解释了“必须注意，这里需要描述的是用户界面的逻辑特征，而不是用户界面”。这个问题绝大多数同学都搞错了。软件运行时的界面截图不是贴在需求分析部分，而应贴在系统设计和测试部分。
3. 需求部分只画一张总的用例图是不够的，还应按角色划分成至少3个用例子图，或者按大的功能模块划分用例子图。
4. 需求分析第5.2节用例描述（或用例规约），有同学为每个角色只画一个表，比如表1为学生用例描述，表2为教师用例描述……。完整的做法是用例图中的每个椭圆（即一个用例）都对应一个表（因此表的数量较多，至少有十几个主要的）。例如学生的“信息查询”用例一个表，“成绩查询”用例一个表（用例大都是动宾短语）……真正要完整描述需求，工作量还是比较大的。从图表的数量上看，用例规约的表应比用例图多。
5. 需求分析第6节“其他非功能性需求”，有同学理解有误，认为是“尚未实现，但很有用的其他功能”。例如可在统计部分排出学生排名，或查询课程时以课表的形式显示等。其实这些都属于“功能性需求”。非功能性需求包括质量属性和约束性，例如可靠性、性能、图形用户界面、安全性、压力等（见测试报告目录，从第3节开始）。例如下面的截图：

性能需求

- 相互合作的用户数量：5
- 系统支持的并发操作数量:500
- 响应时间：根据设备差异而有所不同，一般在 3 秒内
- 与实时系统的时间关系：相同
- 容量需求
  - 存储器：PC 磁盘
  - 磁盘空间：要求不高 100M 以内
  - 数据库中表的最大行数：3000

二、概要设计：

1．概要设计不应太按照类来划分，而应该按照用例、功能等来划分，因为用例、功能常常是跨类的，需要多个类支撑。详细设计才应主要按类来划分，因为详细设计要有利于coder的程序实现。

2．不少同学把概要设计和详细设计混为一谈，把原本属于详细设计的内容（例如类成员的设计，关键处理过程和算法等）写进了概要设计里，导致到了详细设计部分，要么无内容可写，要么就只能重复概要设计已写的内容。

其实概要设计的核心是“逻辑划分”，它的产出是软件体系结构或功能模块（每个模块由若干个类支撑，但类的设计大多应放在详细设计部分），它不关心具体实现。而详细设计的核心是“产生代码”，它关心具体实现，要仔细设计每个类的成员。要达到让一个只学过计算机语言的人，拿到详细设计就能立即编写代码的程度。

3．有同学在第3.2节“系统结构设计部分”采用MVC来划分，这个没问题。但他们把Model部分的“学生、教师、管理员”和View部分的各种窗口都叫做“用例”就不对了，这些应该叫做对象或系统组成部分。用例表示一种功能、服务、场景，一般用主谓或动宾短语来表示，很少用名词。另外有同学把MVC划分放到详细设计里面也不对。

4．有同学把那几个存有账号密码或课程成绩信息的文件归为数据库设计。其实这点数据构不成数据库的规模，最多只能叫数据词典或数据规约。考虑到大家还没学过数据库或正在学，因此没有要求大家作数据库的设计。

三、详细设计：

1．有些IDE（或装了插件）具有类图自动生成功能。但一般生成不全，有些细节还是要手工画的。用IDE生成的类图大都只能用于概要设计，详细设计阶段还是不够。

2．详细设计的第3节“部件详细设计”（或称类详细设计），如果要完整做的话，除了画类图，还应对每个类的每个成员逐一进行说明：对于属性要说明其类型、访问权限和用途，对于方法要说明其方法签名和用途等。对于稍复杂的方法，还需要描绘算法步骤，要靠文字、伪代码、图表等形式来描述，不应贴代码。代码是实现阶段的产出，这样做有凑字数的嫌疑。

3．流程图的过程框里的内容，不宜写成函数调用的形式，流程图体现的是动作序列，框里一般

是动宾结构或主谓结构的文字说明。

4. “部件/子系统详细设计”一节中，有许多同学是按照功能或界面来进行划分。这里大家受了模板的影响，也不能说不行，但是这是结构化的设计风格，不是面向对象的风格，面向对象应该以类为单元进行划分。
5. “部件/子系统详细设计”一节的表格中，不少同学把“部件入口参数”和“部件出口参数”写成函数名了，其实它的本意是某些基本数据类型的变量或类的对象，作为形参和返回值。
6. 有些同学把“部件”一词单纯地理解为只包含方法/函数。其实该模板中的“部件”可以指代任何具有“模块”属性的东西，包括但不限于类、方法/函数、多个类形成的构件/子系统、界面、界面中的控件、动态连接库、库文件等。

   这套模板是结构化设计时代的产物，比较老了。关于“部件”描述的表格其实设计得并不好，对于面向对象的设计表达起来比较别扭。同学们完全可以自己设计一种表，专门表达类的设计。

四、测试：

1. 有同学用代码代替测试用例设计，不好。测试用例设计是单独环节，宜用表格表现。有了测试用例设计后才用代码实现，代码的可读性自然不如表格好。有同学用了一些自动测试框架JUnit，并写了一些测试脚本，这是可以的。但测试代码不能代替对测试用例设计的表现。
2. 有不少同学仅完成用例设计，就以为测试部分做完了，其实还包括测试的执行和记录。即把测试用例输入系统，如实记录实测结果，并和预期结果比对。最好还加上一段测试后的分析和总结，这样才算一份完整的测试工作。展示测试结果只需在表格中记录数据即可。可以少量贴一些运行截图，但不要贴太多运行截图或IDE的开发截图。
3. 单元测试：《大作业要求》中写明了单元测试应以controller包中的一个“类”为单元，而不是只测一个方法（不过有些类只有一个方法）。当该类有多个方法时，每个方法先要单独测（单独画流图，分析基本路径）；单独测了后还可以更进一步，采用教材或完整版课件中介绍的面向对象的单元测试方法，例如随机测试、划分测试、基于动态模型的测试等手段（只需一种即可）进行联测。

   有些同学做基本路径测试不画控制流图，不计算环路复杂度，不做基本路径集合的分析，就直接给出了测试用例，没有体现出过程。
4. 集成测试：面向对象的集成测试我们课上只提了一下，没仔细讲，可参见教材的19.3节“面向对象测试策略”或完整版课件中介绍的方法，例如基于线程的测试、基于使用的测试等。集成测试应该分多次、增量式进行，即一次只集成一个类。有的同学只集成一次就完了，即在一次测试中把全部要集成的类都纳入其中。这种做法不叫集成测试，叫冒烟测试。不过鉴于集成测试需要编写桩模块或驱动模块，要同学们有一定的开发能力。
5. 验收测试：我的要求是“教师”角色，而不仅是Teacher类，如果测Teacher类就成了单元测试了。任务是结合用例图中“教师”这个角色，包含多个用例或功能（信息查询、修改信

息、开课、全部课程、成绩登记以及成绩统计），从外部进行测试（黑盒测试）。

6. 黑盒测试中，等价类划分和边界值分析之后，还有用例设计环节，然后才是测试记录。教师有多个功能或用例（信息查询、修改信息、开课、全部课程、成绩登记以及成绩统计），等价类划分最好依照功能来，每个功能一个小表，而不要都集中在一个大表里。
7. 有一组测试中出现了安卓操作系统，不知怎么回事？

五、写作与格式的常见问题：

1. 写作很重要，体现了组内的认真程度，可以给老师好的印象，减轻老师的“批改气”！
2. 组长最后的把关非常重要，这里体现了组长的“品控”能力。有许多组都是直接把组员各自写得粘贴到一起就完事，没有从整体上把关格式。因此出现了不同环节格式质量波动大的现象，可能看“需求分析”还好好的，到了“概要设计”换了一个人就乱得不行了！应该汇总后集中调整格式。而且调完后，还要再次发给所有组员确认后再提交。
3. Mac上的word编辑的文档，即使格式调好了，拷到Windows下的word时，仍然可能格式会乱。这种情况发生在组内不同的同学用不同类型的电脑上。由于我只有windows，组长汇总后最好在Windows电脑下再看看。
4. 《报告模板》中没有用到的章节、图表、注释性文字不能留在文档里，要删除，不能凑字数。
5. 图、表不居中。每一个独立的图和表都应该页面居中放置（注意要在段首顶格的情况下居中，参考我上传到QQ群里的《带你学格式规范》）。
6. 图和表都要编号，图标题在图的下面，表标题在图的上面。标题和图表不能分隔在两页上（表太长除外）。
7. 图的大小比例控制不当，不是太大，就是太小，要不就是长宽比失调，导致偏扁或偏瘦。有同学直接截屏starUML软件绘图界面，而不是直接截取图的部分，导致图两边有不少无效内容。有同学直接拿我课件里的图来用，但又没有清晰地拷贝过来。有同学为了能在页面放得下，强行缩得很小，导致字像蚂蚁看不清楚，其实这种情况可以把图切开分片，编上序号，就看清楚了。以上这些细节虽然不影响阅读，但体现了用心程度。
8. 图片糊，原因多种。有同学是网上找的，质量不佳。有同学是用截屏软件截取的。有同学不按要求用StarUML画。其实最好的方法使用绘图软件自带的导出功能导出。
9. 同一级别（比如正文、图标题等）的字体、字号不统一。段内行距不统一。
10. 文中不要用“我们”，太口语化。正式的技术文档应采用“笔者、本组、本课题、本系统”等。