**System Analysis and Design**

**Homework Assignment 2**

1. **Answer the following questions:**
   1. What is use case? What should be included in a use case? Why we should use use case in system development ?

**用例是什么？**

用例是文本故事（而不是图表），是一组相关的成功和失败的场景，是一系列动作的散文式表达，动作由一个或多个参与者（人类或非人类）和系统本身执行，描述了一些参与者使用一个系统来实现用户的目标的场景。用例用于发现和记录系统需求，如果一个图表清楚解释了用例的文本，也可以使用它。

**用例中应该包括什么？**

用例中至少应该包括参与者和主要成功场景，描述参与者执行动作从而实现目标的具体场景。

如果是详述格式的用例，还应该包括用例名，设计范围，级别，主要参与者，项目关联者和利益相关者，先决条件，成功保证，主要成功场景，扩展的成功和失败的交替场景，与非功能性需求相关的特殊需求，技术和数据变化列表，不同的I/O方法和数据格式，发生频率影响调查、测试和实现的时间，其它问题如未决问题。

**为什么要在系统开发的过程中使用用例？**

（1）、用例使领域专家或需求提供者能够自己编写（或参与书面）用例变得更加简单和具有可行性；

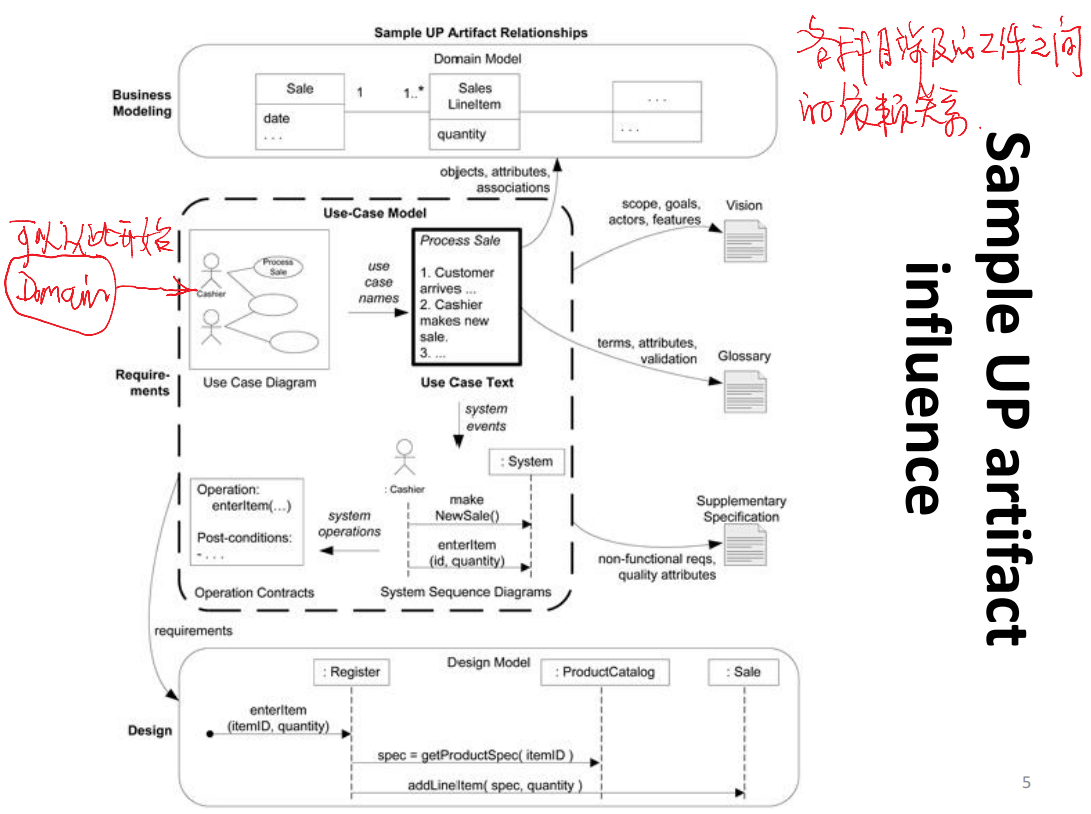
（2）、用例强调用户的目标和角度观点；

（3）、用例的一个优点是能够在复杂度和形式方面进行上下扩展。基于其精练的技巧和格式，有很强的伸缩性。

* 1. In system development cycle, artifacts may influence each other. Why we have to consider these influences in our development process?

因为用例中的高级目标和用例图会被输入到用例文本的创建中，而用例可以反过来影响许多其他的分析、设计、实现、项目管理和测试工件的过程。开发过程会受到这些工件的影响，所以需要考虑工件之间的互相影响对开发过程的影响。

* 1. UP give a guideline for UP artifacts influences. It is explained in a diagram to show artifacts influence relations. Describe these relations in your own words with more your detailed explanations.



工件之间的相互影响从上图可以看出，比如用例图会影响用例文本中用例的命名，用例文本可以提供系统事件帮助构建系统序列图，提供对象、属性、关联关系给领域模型，提供术语、属性、规范给术语表，用例模型可以提供系统范围、目标、参与者、特征等信息给项目愿景，提供非功能性需求、质量属性给补充说明，提供需求给设计，系统序列图可以提供系统操作给操作契约。

* 1. Use cases can be in three formats, what are they? What are the differences?

简约格式，非正式格式，详述格式。

简约格式是一段式的总结，通常是主要的成功场景，在早期需求阶段创建；

非正式格式的段落格式非正式，可以在多个段落中涵盖各种场景。

详述格式所有步骤和变化都写得很详细。有支持部分、成功保证、主要场景、备用场景等。

它们的包含内容详细程度和格式的正式性都有所不同，可以根据需要选择相应的用例格式。

* 1. What techniques can be used to find use cases ? Describe how to use these techniques.

用例被定义为满足主要参与者的目标。因此，找到用例的基本过程是：

（1）、选择系统边界

（2）、确定主参与者和每个参与者的目的

（3）、根据目标定义用例

使用用例图、参与者-目标列表、识别外部事件、测试、UML图等技术可以帮助找到用例，使用方法：

（1）、使用用例图时，当发现系统运行之后可能导致的结果后，就把结果画到用例图中，并且用实现这些结果所达到的目标为用例命名。

（2）、在使用参与者-目标列表时，首先需要对系统进行分析，列出系统的参与者和这些参与者使用系统所要达到的目标，按照这些参与者和他们使用系统所要达到的目标编写一个参与者-目标列表，审查并优化这个列表，然后绘制用例图。

（3）、通常，进行事件分析，识别外部事件之后，那些被分成一组的事件属于相同的用例，根据这些事件的操作和产生的结果就可以找到相应的用例了，接着编写用例，

（4）、进行测试可以帮助找到更加有效的用例，比如老板测试、EBP测试等等，低层次的目标可以构成用例，但不是需求分析层用例。

1. **Revise the collection of all your artifacts you have done for your inception phase of your project in the following.**
2. Development Case
3. Project Vision
4. Business Case
5. Use Case Diagrams
6. Fully Dressed Use Cases
7. Domain Model (Conceptual Class Diagrams)
8. SSD
9. System Operation Contracts
10. Supplementary Specification
11. Glossary
12. Development Plan
13. Iteration Plan

Notice that some sections may be empty in your document at this moment. You may also add some supplementary sections in your document.

# **Inception Phase启动阶段**

**1 Development Case开发样例**

首先要确定mesh系统的统一过程开发步骤和这个项目会用到的一些统一过程的工件。

**1.1 UP steps统一过程开发步骤**

（1）、初始阶段(Inception)：首先确定系统是可行的，需要进行可行性研究(Feasibility study)、业务案例(Business case)、适用范围评估(Scope)、评价判断(Estimation)、风险评估(Risk assessment)、选择适合的体系结构(Candidate architecture)

（2）、精化阶段(elaboration)：然后建立在约束条件下构建系统的能力，需要进行分析功能需求(Functional Requirements)、设计用例(Use cases)、设计域模型(Domain model)、解决风险(Address Risks)、计划/重新计划(Plan/replan)、设计模型(Design model)、建立体系的基准线(Architectural baseline)

（3）、构造阶段(construction)：接着构建测试版的系统，需要进行添加新代码(New code)、重构(Refactoring)、添加/扩展用例(Add/extend use cases)、计划/重新计划(Plan/replan)

（4）、移交阶段(Transition)：最后向客户推出全功能系统，需要进行优化(Optimization)、错误修复(Bug fixes)、制造成品(Production)、发布(release)

**1.2 UP Artifacts统一过程工件**

（1）、设想(Vision)、范围(scope)和业务样例描述(business case descriptions)

（2）、用例(Use cases) 、用例图(Use case diagrams)

（3）、补充规范(Supplementary specifications)

（4）、域模型（概念类图）Domain Model (Conceptual class diagrams)

（5）、术语表(Glossary)

（6）、系统开发软件(SSD)

（7）、系统操作合同（System Operation Contracts）

（8）、使用UML图表进行域建模和系统建模

（9）、源代码和源代码文档

（10）、网络图形(Web graphics)

（11）、数据库模式(Database schema)

**2 Project Vision项目设想**

**2.1 Overview of the Project项目概况**

网格可以由许多不同的领域工具（预处理工具）以不同的格式（文本、二进制、数据库、自定义格式等）创建。这些格式将作为我们设计系统的输入。我们的系统将网格加载到内存中，将网格从内存保存到一个或多个文件中，可以将一种网格格式转换为另一种网格格式，可以以某种视觉形式查看网格，可以有一种用于高效计算的内存存储格式，可以将要存储的网格划分为分布式内存格式以用于高效并行计算，并且可以绘制、粗化、编辑网格。存储格式可以直接转换成文件。

我们还为领域程序员提供了一个域应用程序模板。mesh应用程序通常由mesh声明、mesh元素的核函数定义、对mesh所有结构的操作以及一些io操作组成。

**2.2 Goals of the Project项目目标**

（1）、领域程序员可以从文本文件以某种格式加载网格到内存中。

（2）、以有效的数据结构在存储器中存储网格结构。

（3）、将网格从内存中以特定格式保存到文本文件

（4）、领域程序员可以直接在代码中定义网格（元素）。

（5）、领域程序员可以在Mesh上定义内核操作

（6）、领域程序员可以定义网格上的算法（网格应用程序）

（7）、类库应该在mesh上实现内核操作的循环

（8）、类库应该输出到一个文本（或二进制）文件中，以某些用户定义或预定义的格式保存每个步骤中的计算（模拟）结果。

（9）、系统应在C/C++语言或FORTRAN语言中实现

（10）、领域程序可以运行在CPU、多核/多核和集群体系结构上

（11）、类库应该允许用户程序跟踪性能

**2.3 Constraints of the Project项目约束**

（1）、开发时间：一学期

（2）、开发人数：7人

（3）、开发技术：Windows/Linux系统下的C/C++编程技术

**3 Business Case 业务案例**

**3.1 项目功能可行性分析**

我们的网格技术系统可以通过输入网格物体（由许多不同的域工具（预处理工具）以不同的格式（文本，二进制，数据库，自定义格式等）创建），将网格物体加载到内存中，将网格物体从内存中保存到一个文件或多个文件中，可以将一种网格物体格式转换为另一种网格物体，以某种可视形式查看网格物体，系统可以绘制，粗化，编辑网格。内存存储格式可以直接转换为文件。系统还为领域程序员提供域应用程序模板。

**3.2 项目前景价值**

该系统有很好的商业价值和前景，在商业方面比如Johnson&Johnson与Merck等制药公司、BMW与波音等制造企业却已利用这一技术的处理能力和存储空间进行仿真试验，因此可以实现很好的对商品的测试工作，因此该产品出售于产品公司将有不错的收益

此外网格技术的还有很多的应用面，如医生可以用网格技术制作出病人器官的三维模型，作为诊断疾病的辅助手段; 网格可以处理来自商店现金记录或金融市场的数据流。

综上我们可以看到这个系统它的商业价值，为了具体的实现收益方案，我们可以采取前期提供试用的方式，如让相关企业试用一个月，并针对不同的企业提供有针对性的使用手册，如对于建筑企业，我们应该教会他们如何使用其实现建筑的建模。因此他们更加能够使用该系统得到收益，在试用期得到一定的收益后，企业将会续用该产品，自此形成好的合作，这样我们就可以有个比较稳定的产品收入。

总之，网格计算并非是可望不可及的乌托邦，其商业应用的广阔前景就在眼前。

**3.2 key non-functional requirements 关键非功能性需求**

（1）、最后的制品需要有一个较为良好的外观界面，最好是一个合格的GUI图形交互界面，既能够较好的对控件的使用范围进行规范，也可以能够满足客户的感官需求。

（2）、产品的易用性也是一大关键需求，最后的制品在人机交互上最好尽量的简单，使得客户凭借简单的感觉就能直接上手，不需要额外学习复杂的操作。

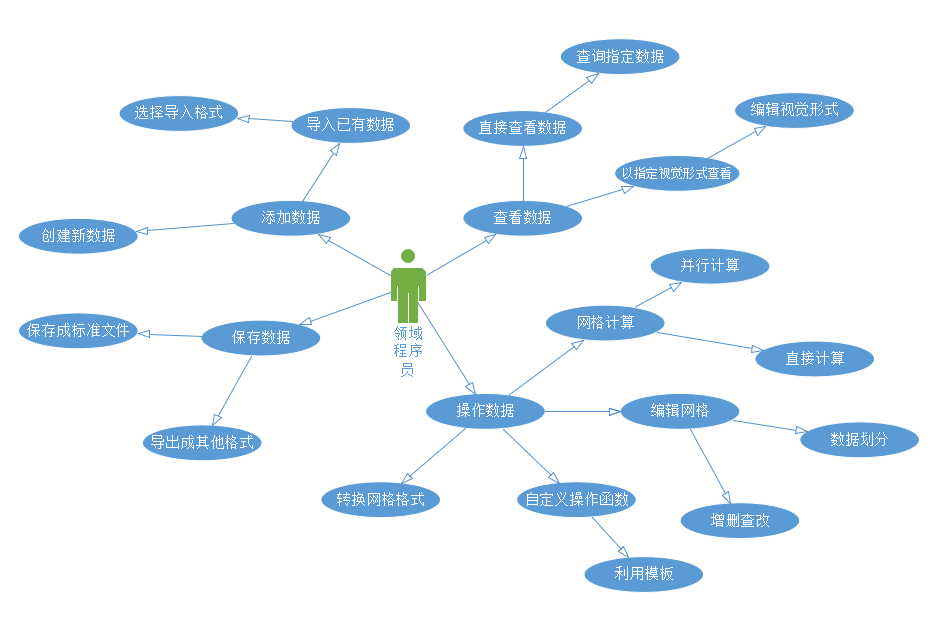
（3）、产品的运行最好尽可能地稳定，尽量使用合格的算法使完成任务的速度不至于太慢，资源的使用效率尽可能提高，出现的故障尽可能的少。

（4）、产品的安全性也十分重要，最好可以规定不同用户的使用权限，限定访问范围。

（5）、产品在不同平台最好能有较好的兼容性。

**4 Use-Case Model 用例模型**

**4.1 Use case diagrams 用例图**



**4.2 Use cases 用例**

4.201 UC1：读取数据文件到还未有文件打开软件中

范围：整个系统

级别：用户目标级

参与者：领域用户、类库

主要成功场景：

1. 领域用户在软件的工具栏点击“打开文件”，选择要打开的文件后，点击“打开”

2. 类库调用函数，查询判断文件格式是否正确，是否可读，成功则将文件导入软件

3. 类库调用函数，将导入的文件以图表或图形的形式显示

4. 领域用户可以在视图区域看到导入的文件以图表或图形的形式显示

备选场景：文件打开失败

2a. 在第2步，类库调用函数发现文件格式错误，弹出提示打开文件格式错误，并允许领域用户重新选择格式正确的网格数据文件。

2b. 类库调用函数发现文件不可读，弹出提示打开文件不可读，并允许领域用户重新选择可读的网格数据文件。

4.202 UC2：保存文件

范围：整个系统

级别：用户目标级

参与者：领域用户、类库

主要成功场景：

1. 领域用户在软件的工具栏点击“保存文件”

2. 领域用户选择要保存的文件路径后，选择“保存格式”

3. 领域用户点击“保存”

4. 类库调用函数，将文件以选择的格式保存在指定的文件路径中

备选场景：文件保存失败

4a.在第4步，若保存的文件路径容量不够，弹出提示存储容量已满，并允许领域用户选择新的存储路径

4.203 UC3：导入数据文件到已经有文件打开软件中

范围：整个系统

级别：用户目标级

参与者：领域用户、类库

主要成功场景：

1. 领域用户在软件的工具栏点击“打开文件”，选择要打开的文件后，点击“打开”

2. 类库调用函数，查询判断文件格式是否正确，是否可读，成功则将文件导入软件

3. 类库调用函数，弹出提示，让用户选择是覆盖当前文件的视图窗口还是创建一个新的视图窗口显示

4. 领域用户若选择覆盖，则在原来的视图窗口进行操作，选择创建一个新的视图窗口，那么就在新的视图窗口进行操作

5. 将导入的文件在选择的视图窗口以图表或图形的形式显示

6. 领域用户可在选择出的视图区域看到导入的文件以图表或图形的形式显示

备选场景：文件打开失败

2a. 在第2步，类库调用函数发现文件格式错误，弹出提示打开文件格式错误，并允许领域用户重新选择格式正确的网格数据文件。

2b. 类库调用函数发现文件不可读，弹出提示打开文件不可读，并允许领域用户重新选择可读的网格数据文件。

4.204 UC4：创建新的结点

主要成功场景：领域用户在软件的工具栏点击“创建新的结点”，类库调用函数将鼠标光标呈现为指示链接的指针，领域用户移动光标在一个位置点击鼠标，这个位置上就会出现一个新的结点。

备选场景：

用户可以在坐标区域输入想要创建的结点的坐标，然后点击“创建结点”，就可以在指定的位置创建一个新的结点

4.205 UC5：创建新的线段

主要成功场景：领域用户在软件的工具栏点击“创建新的线段”，类库调用函数将鼠标光标呈现为指示链接的指针，领域用户移动光标在一个位置点击鼠标，然后拖动光标到另一位置直到放开鼠标，这两个位置之间就会出现一个新的线段。

备选场景：

用户可以在坐标区域输入想要创建的线段的两个端点的坐标，然后点击“创建线段”，就可以在指定的位置创建一个新的线段

4.206 UC6：创建新的矩形

主要成功场景：领域用户在软件的工具栏点击“创建新的矩形”，类库调用函数将鼠标光标呈现为指示链接的指针，领域用户移动光标在一个位置点击鼠标，然后拖动光标到另一位置直到放开鼠标，类库会自动根据这两个位置之间的水平距离和垂直距离创建一个新的矩形。

备选场景：

用户可以在坐标区域输入想要创建的矩形的左上角端点和右下角端点的坐标，然后点击“创建矩形”，就可以在指定的位置创建一个新的矩形

4.207 UC7：创建新的圆形

主要成功场景:领域用户在软件的工具栏点击"创建新的圆形",类库调用函数将鼠标光标呈现为指示链接的指针,领域用户在一个位置点击鼠标,将圆心确定于此位置,然后拖动鼠标,将出现一个以圆心到鼠标的距离为半径的新的圆形。

备选场景:

用户可以在坐标区域输入想要创建的圆的圆心和半径,然后点击"创建圆形",就可以在指定位置创建一个圆形。

4.208 UC8：创建新的三维立体矩形

主要成功场景:领域用户在软件的工具栏点击"创建新的三维立体矩形",类库调用函数将鼠标光标呈现为指示链接的指针,领域用户在一个位置点击鼠标,然后拖动光标到另一位置直到放开鼠标,类库会自动根据这两个位置之间的水平距离和垂直距离创建出一个立方体底面,然后用户向上或者向下拖动鼠标可以确定出这个立方体的高度并创建图形。

备选场景:

用户可以在坐标区域输入一个基点的坐标,并输入立方体的长宽高即可在指定位置创建一个矩形。

4.209 UC9：创建新的球

主要成功场景:领域用户在软件的工具栏点击"创建新的球",类库调用函数将鼠标光标呈现为指示链接的指针,领域用户在一个位置点击鼠标,将球心确定于此位置,然后拖动鼠标,将出现一个以球心到鼠标的距离为半径的新的球体。

备选场景:

用户可以在坐标区域输入想要创建的球的球心和半径,然后点击"创建球",就可以在指定位置创建一个球体。

4.210 UC10：拉伸二维图形的点

主要成功场景：领域用户在软件的工具点击“拉伸点”，类库调用函数将鼠标光标呈现为指示链接的指针，领域用户在二维图形位置点击鼠标，此时二维图形进入可修改模式，并显示所有的坐标点。领域用户用鼠标选择任意坐标点后，进行拖动操作，即可对该点进行拉伸，拉伸效果为与该点相连的线段都会随着坐标点的移动而变化

备选场景：

在创建二维图形时，二维图形将处于可修改模式，用户此时可以对二维图形的任意坐标点进行拉伸操作

用户可以在坐标区域输入二维图形的坐标点，然后点击“拉伸”，就可以对该坐标点进行拉伸

4.211 UC11：拉伸二维图形的线段

主要成功场景：领域用户在软件的工具点击“拉伸点”，类库调用函数将鼠标光标呈现为指示链接的指针，领域用户在二维图形位置点击鼠标，此时二维图形进入可修改模式。领域用户用鼠标选择任意二维图形的任意线段后，进行拖动操作，即可对该线段进行拉伸

备选场景：

在创建二维图形时，二维图形将处于可修改模式，用户此时可以对二维图形的任意线段进行拉伸操作

用户可以在坐标区域输入二维图形任意线段端点的坐标点，然后点击“拉伸”，就可以对选定线段进行拉伸

4.212 UC12：拉伸三维图形的点

主要成功场景：领域用户在软件的工具点击“拉伸点”，类库调用函数将鼠标光标呈现为指示链接的指针，领域用户在三维图形位置点击鼠标，此时三维图形进入可修改模式，并显示所有的坐标点。领域用户用鼠标选择任意坐标点后，进行拖动操作，即可对该点进行拉伸，拉伸效果为与该点相连的线段都会随着坐标点的移动而变化

备选场景：

在创建三维图形时，二维图形将处于可修改模式，用户此时可以对三维图形的任意坐标点进行拉伸操作

用户可以在坐标区域输入三维图形的坐标点，然后点击“拉伸”，就可以对该坐标点进行拉伸

4.213 UC13：拉伸三维图形的线段

主要成功场景：领域用户在软件的工具点击“拉伸点”，类库调用函数将鼠标光标呈现为指示链接的指针，领域用户在三维图形位置点击鼠标，此时二维图形进入可修改模式。领域用户用鼠标选择任意三维图形的任意线段后，进行拖动操作，即可对该线段进行拉伸

备选场景：

在创建三维图形时，二维图形将处于可修改模式，用户此时可以对三维图形的任意线段进行拉伸操作

用户可以在坐标区域输入三维图形任意线段端点的坐标点，然后点击“拉伸”，就可以对选定线段进行拉伸

4.214 UC14：点击两个点在之间创建线段

主要成功场景：领域用户在软件的工具栏点击“创建新的线段”，库类调用函数将鼠标光标呈现为指示连接的指针，领域用户移动光标在一个点处点击鼠标，然后拖动光标到另一个点后放开鼠标，便可以在这两个点之间生成线段。

备选场景：

领域用户在软件的工具栏点击“创建新的线段”，库类调用函数将鼠标光标呈现为指示连接的指针，领域用户移动光标在一个点出点处鼠标，然后在另一个点处点击鼠标，便可以在这两个点之间生成线段。

4.215 UC15：放大视图

主要成功场景：领域用户在软件的工具栏点击“调整视图”，库类调用函数将鼠标光标呈现为指示连接的指针，领域用户移动光标在一个位置点击鼠标，然后拖动光标使生成的虚线框包含图形后放开鼠标，点击”调整视图“栏的加号（放大）图标将虚线框出的视图放大到需要大小停止点击即可。

备选场景：

在绘图界面持续按住鼠标左键，向上滑动鼠标中键直至视图放大到需要大小即可。

用户可以在坐标区域输入想要放大的视图矩形范围的左上角端点和右下角端点的坐标，然后点击”调整视图“中的加号（放大）图标将选择范围的视图放大到需要大小停止点击即可。

4.216 UC16：缩小视图

主要成功场景：领域用户在软件的工具栏点击“调整视图”，库类调用函数将鼠标光标呈现为指示连接的指针，领域用户移动光标在一个位置点击鼠标，然后拖动光标使生成的虚线框包含图形后放开鼠标，点击”调整视图“栏的减号（缩小）图标将虚线框出的视图缩小到需要的大小停止点击即可。

备选场景：

在绘图界面持续按住鼠标左键，向下滑动鼠标中键直至视图缩小到需要大小即可。

用户可以在坐标区域输入想要缩小的视图矩形范围的左上角端点和右下角端点的坐标，然后点击”调整视图“中的减号（缩小）图标将选择范围的视图缩小到需要大小停止点击即可。

4.217 UC17：删除点

主要成功场景：领域用户在软件的工具栏点击“删除点”，库类调用函数将鼠标光标呈现为指示连接的指针，领域用户移动光标在图像一个点处点击鼠标，这个位置的点会被删除，呈现空白。

备选场景：

用户可以在坐标区域输入想要删除的点的坐标，然后点击“删除点”，就可以将指定位置的点删除

4.218 UC18：删除线段

主要成功场景：领域用户选择一条线段，这条线段会高亮显示。按下delete键删除或右键（再选择删除），这条线段就会被删掉。

备选场景：

用户可以点击菜单栏下的删除，再点击删除下的删除线段，进入删除线段模式，这个阶段不会选择其他的图形。之后Ctrl-A来选择所有的线段（不进入删除模式的话Ctrl-A会选择界面上的所有的图形），这些线段高亮显示，再进行如上的删除操作。

同理，在进入删除线段模式后，用户用户可以拖动鼠标选择一部分线段，这些线段高亮显示，再进行如上的删除操作。

4.219 UC19：删除二维图形

主要成功场景：领域用户选择一个二维图形，这个二维图形会高亮显示。按下delete键删除或右键（再选择删除），这个二维图形就会被删掉。

备选场景：

用户可以点击菜单栏下的删除，再点击删除下的删除二维图形，进入删除二维图形模式，这个阶段不会选择其他的图形。用户可以Ctrl-A来选择所有的二维图形，这些二维图形高亮显示，再进行如上的删除操作。

同理，在进入删除二维图形模式后，用户可以拖动鼠标选择一部分二维图形，这些二维图形高亮显示，再进行如上的删除操作。

4.220 UC20：删除三维图形

主要成功场景：领域用户选择一个三维图形，这个三维图形会高亮显示。按下delete键删除或右键（再选择删除），这个三维图形就会被删掉。

备选场景：

用户可以点击菜单栏下的删除，再点击删除下的删除三维图形，进入删除三维图形模式，这个阶段不会选择其他的图形。用户可以Ctrl-A来选择所有的三维图形，这些三维图形高亮显示，再进行如上的删除操作。

同理，在进入删除三维图形模式后，用户可以拖动鼠标选择一部分三维图形，这些三维图形高亮显示，再进行如上的删除操作。

4.221 UC21：计算选定直线的长度

主要成功场景：领域用户在软件的功能栏点击“计算”，选定计算“计算选定直线的长度”，类库调用函数将鼠标光标呈现为指示链接的指针，领域用户移动光标点击一条直线，就会在结果区域输出选定直线的长度。

备选场景：

用户可以直接在元素选择区的元素1输入框输入想要计算的直线的标号，然后点击“计算长度”，就会在结果区域输出选定直线的长度。

4.222 UC22：计算选定两条直线之间的角度

主要成功场景：领域用户在软件的功能栏点击“计算”，选定计算“计算选定两条直线之间的角度”，类库调用函数将鼠标光标呈现为指示链接的指针，领域用户移动光标依次点击两条直线，就会在结果区域输出选定两条直线之间的角度。

备选场景：

用户可以直接在元素选择区的元素1输入框输入想要计算角度的第一条直线的标号，然后点击“添加元素”，在元素2输入框输入想要计算角度的第二条直线的标号，点击“计算角度”，就会在结果区域输出选定两条直线之间的角度

4.223 UC23：计算选定平面的面积

主要成功场景：领域用户在软件的功能栏点击“计算”，选定计算“计算选定平面的面积”，类库调用函数将鼠标光标呈现为指示链接的指针，领域用户移动光标点击一个平面，就会在结果区域输出选定平面的面积。

备选场景：

用户可以直接在元素选择区的元素1输入框输入想要计算的平面的标号，然后点击“计算面积”，就会在结果区域输出选定平面的面积。

4.224 UC24：计算选定立体的体积

主要成功场景：领域用户选择一个立体图形，这个立体图形会高亮显示。按下右键选择“计算体积”，这个立体的体积就会经过计算显示出来。

备选场景：

用户可以先点击菜单栏下的“计算体积”，再点击需要计算的立体图形，这个立体图形的体积也会被显示出来。

**5 Glossary术语表**

|  |  |
| --- | --- |
| **术语** | **定义** |
| 商品 | 用于销售的产品或服务 |
| 支付授权 | 外部支付授权服务进行的验证活动，该服务将完成并保证对卖方的支付 |
| 制品 | 所有工作产品的统称，如代码、web图形、数据库模式、文本文档、图、模型等 |
| 系统分析 | 旨在研究特定[系统结构](https://www.wiki-wiki.top/baike-%E7%B3%BB%E7%BB%9F%E6%9E%B6%E6%9E%84" \o "系统架构)中各部分（各[子系统](https://www.wiki-wiki.top/baike-%E5%AD%90%E7%B3%BB%E7%BB%9F" \o "子系统)）的相互作用，系统的对外接口与界面，以及该系统整体的行为、[功能](https://www.wiki-wiki.top/baike-%E5%8A%9F%E8%83%BD_(%E8%BB%9F%E9%AB%94%E5%B7%A5%E7%A8%8B)" \o "功能 (软件工程))和局限，从而为系统未来的变迁与有关[决策](https://www.wiki-wiki.top/baike-%E5%86%B3%E7%AD%96" \o "决策)提供参考和依据。 |
| UML | UML是一种开放的方法，用于说明、可视化、构建和编写一个正在开发的、面向对象的、软件密集系统的制品的开放方法。 |
| 接口 | 强加于外部系统接口之上的约束 |
| 系统用例 | 描述对一个软件系统的使用的用例 |
| 业务用例 | 描述顾客和有关人员如何使用业务的用例 |
| 基本业务过程 | 一个人在某个时刻某个地点所执行的任务，用以响应业务事件。该任务能够增加可量化的业务价值，并且以持久状态留下数据。例如，批准信用卡的信用额度或者确定订购的价格。 |
| 领域模型 | 领域概念的可视化，类似于领域实体的静态信息模型 |
| 设计模型 | 描述逻辑设计的一组图，包括软件类图、对象交互图等 |
| 数据模型 | 包括数据库方案，以及在对象和非对象表示之间映射的策略 |

**6 Development plan开发计划**

**6.1 Artifacts for the development开发过程工件**

精化阶段(elaboration phase)是迭代进行的的，每次迭代都需要一周的时间。在这个阶段，我们需要以下工件：

（1）、工具Tools：Vscode（用于编写具体计划的markdown文件）、gitee（用于协作）。

（2）、开发者People：一个组长和其他七个组员一起制定一个计划。

（3）、掌握知识Education：使用git工具进行开发，可能需要matlab知识或其他参考资料来学习如何处理网格存储或计算问题。

（4）、资源Resource：需要用户提供更详细的需求。

**7 Iteration Plan迭代计划**

**7.1 迭代模型概述**

迭代式模型是统一软件开发过程所推荐的软件生命周期模型。迭代包括产生产品发布(稳定,可执行的产品版本)的所有开发活动,和使用该产品发布的所以其它外部因素。开发迭代是一次完整地经过所有工作的过程,且过程至少包括需求分析,详细设计,编码和测试工作。

相比于传统的瀑布模型，我们计划对Mesh系统较早引入编程和测试,并重复这一循环。在这种生命周期方法中,开发被组织成一系列短周期的小项目，根据产品的发布需求,我们将这个短周期设置为14天，也成为一个迭代。

迭代生命周期基于对经过多次迭代的系统进行持续扩展和精化,并以循环反馈和调整为核心驱动力,使之最终成为适当的系统。每次迭代都产生可执行的但不完整的系统,这个系统并不是可以交付的产品,直到多次迭代之后(10~15次迭代)之后才能合格地用于产品部署。

**7.2 第一次迭代**

开发团队计划在第一次迭代之前召开一个大约3个小时的需求工作会议,所有的业务和开发人员都需要参加本次会议。开发团队根据风险分析选出（1.具有重要的架构意义;2.具有高业务价值;3.具有高风险）的开发业务活动，初步确定可能为：

1. 、确定Mesh的存储形式。
2. 、细化Mesh的核心函数(Kernel Function)，包括核心的初始化和退出,创建Mesh集合/图，Mesh系统的日志，Mesh内部数据的校验，Mesh的裁剪，输出到第一条规定的存储文件中，Mesh数据结构的获取，并行接口的设计。

（3）、对系统的整体需求进行分析。