# 细化阶段-迭代1

**1 Iteration Plan迭代计划**

**1.1 修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 启动阶段1.02 | 2021年3月27日 | 创建迭代计划 | 李浩天 |
| 细化阶段-迭代1 | 2021年5月22日 | 确定第一次迭代的要处理的需求 | 杨宗恺 |

**1.2 Mesh应用在第一个迭代要处理的需求如下：**

+ 实现Mesh网格用例的基本和关键场景：对网格图形进行计算和着色等基本操作

+ 实现用于支持迭代初始化需要的启动用例

+ 域程序员可以从文本文件以某种格式加载网格到内存中。

+ 以有效的数据结构在内存中存储网格结构。

+ 将网格从内存中以特定格式保存到文本文件中

+ 避免不兼容的风险，保证应用在Windows和Linux下都能正常运行。

**1.3 等级划分**

|  |  |  |
| --- | --- | --- |
| **等级** | **需求** | **注解** |
| 高 | 文件的读入和保存 | 进行后续操作的基础 |
| 高 | 以有效的数据结构存储网格结构 | 其它操作需在数据结构上进行 |
| 高 | 实现对网格的计算和染色等基本操作 | 应用的最主要功能 |
| 中 | 维护用户 | 对安全子域有影响 |
| 低 | 图形界面 | 可以先写出程序再建图形界面 |

**1.4 完成任务**

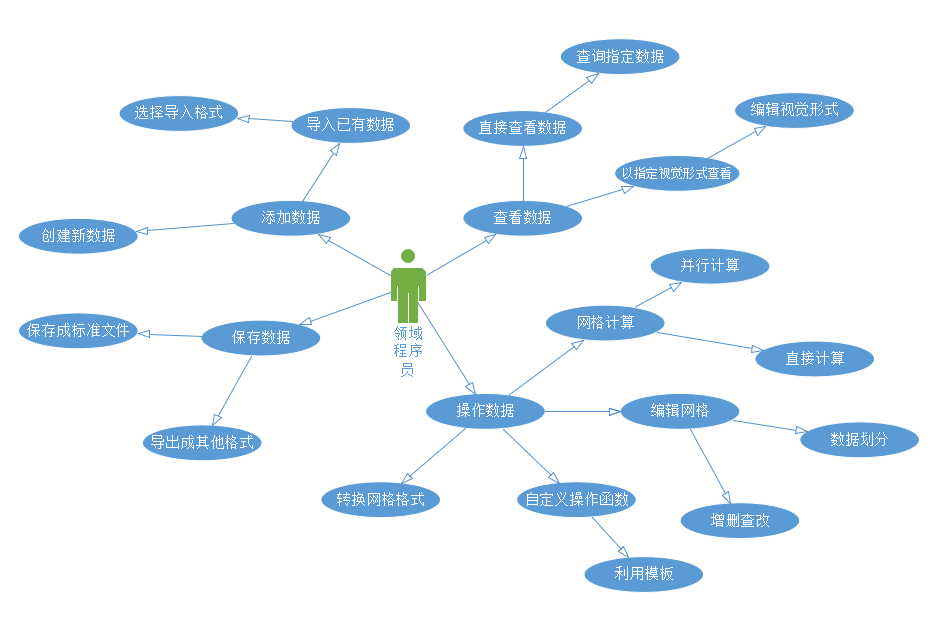
主要进行了读写功能的编程及其测试，确定了稳定需求的主体的数据结构，规避了在不同平台的兼容性的风险。可执行代码见仓库文件。

**2 Use-Case Model 用例模型**

**2.1 修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 启动阶段1.00 | 2021年3月27日 | 创建用例4、5、6、14、24 | 杨宗恺 |
| 启动阶段1.01 | 2021年3月27日 | 创建详细用例1、2、3 | 胡若晴 |
| 启动阶段1.02 | 2021年3月27日 | 创建用例7、8、9 | 李浩天 |
| 启动阶段1.03 | 2021年3月27日 | 创建用例10、11、12、13 | 莫振威 |
| 启动阶段1.04 | 2021年3月27日 | 创建用例15、16、17 | 胡洁珏 |
| 启动阶段1.05 | 2021年3月27日 | 创建用例18、19、20 | 崔子潇 |
| 启动阶段1.06 | 2021年3月27日 | 创建用例21、22、23 | 丁维力 |
| 启动阶段1.07 | 2021年3月27日 | 创建用例图 | 陈家龙 |
| 细化阶段-迭代1 | 2021年5月22日 | 创建详细用例25、26、27 | 杨宗恺 |

**2.2 Use case diagrams 用例图**



**2.3 Use cases 用例**

2.301 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。

2.302 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

2.303 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。

2.304 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

2.305 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

2.306 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

2.307 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。

2.308 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。

2.309 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。

2.310 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

2.311 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

2.312 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

2.313 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

2.314 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。

2.315 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。

2.316 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。

2.317 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

2.318 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。

2.319 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。

2.320 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。

2.321 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。

2.322 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

2.323 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。

2.324 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。

2.325 UC25：以有效的数据结构在内存中存储网格结构

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. 领域用户在软件目录放好存储有网格数据的dat文件
2. 领域用户选择好要载入的数据文件，将文件名放在程序中
3. 类库调用函数，将dat文件中的数据存储在程序的网格结构中

备选场景：文件不存在

在第3步，类库调用函数发现文件不存在，打印错误信息，直接退出程序，令用户选择好数据文件再进行。

2.326 UC26：将网格从内存中以ASCII形式保存到文本文件中

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. 领域用户选择好要保存到文本文件中的形式为ASCII
2. 类库调用函数，将载入的网格数据以ASCII的形式写入到目录下的VTK文件里

备选场景：文件写入失败

在第2步，类库调用函数但写入失败，直接打印错误信息，退出程序。

2.327 UC27：将网格从内存中以二进制形式保存到文本文件中

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. 领域用户选择好要保存到文本文件中的形式为二进制
2. 类库调用函数，将载入的网格数据以二进制的形式写入到目录下的VTK文件里

备选场景： 文件写入失败

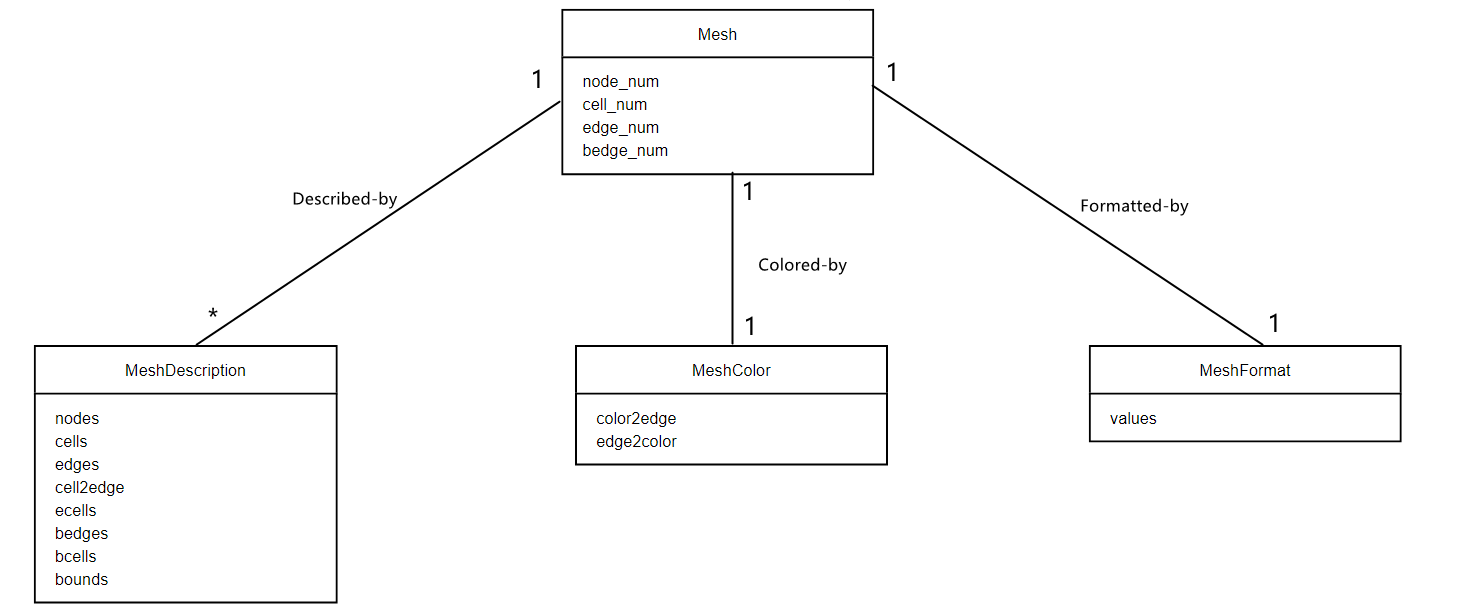
在第2步，类库调用函数但写入失败，直接打印错误信息，退出程序。

**3 Domain Model 领域模型**

**3.1 修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 细化阶段-迭代1 | 2021年5月22日 | 创建概念类图 | 杨宗恺 |

**3.2 Conceptual Class Diagrams 概念类图**

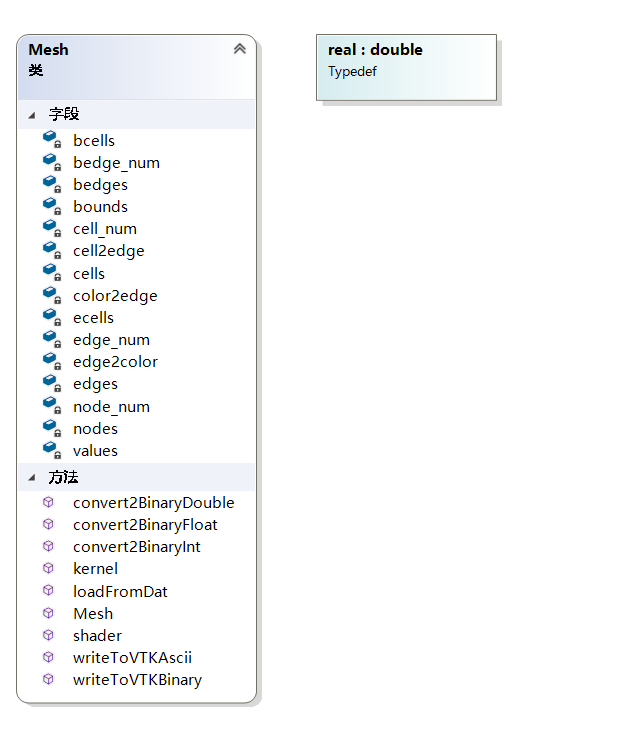


**4 设计模型**

**4.1 修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 细化阶段-迭代1 | 2021年5月22日 | 创建软件类图 | 杨宗恺 |

**4.2 软件类图**



**5 Supplementary Specification 补充规范**

**5.1 修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 启动阶段1.00 | 2021年3月27日 | 创建补充规范 | 杨宗恺 |
| 细化阶段-迭代1 | 2021年5月22日 | 添加功能性补充规范 | 杨宗恺 |

**5.2 功能性**

（1）、日志与错误处理

在持久性存储中记录所有错误。

（2）、可插拔规则

在几个用例的不同场景点执行任意一组规则，以支持对系统功能的定制。

（3）、安全性

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。

**5.3 可用性**

（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）、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。

**5.4 可靠性**

（1）、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。

**5.5 可支持性**

（1）、产品在不同平台如Windows、Linux等最好能有较好的兼容性。

**6 Development Case开发样例**

**6.1 修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 启动阶段1.00 | 2021年3月27日 | 创建开发样例 | 杨宗恺 |

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。

**6.2 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)

**6.3 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)

**7 Project Vision项目设想**

**7.1 修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 启动阶段1.06 | 2021年3月27日 | 创建项目设想 | 丁维力 |

**7.2 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。

**7.3 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

**7.4 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

**8 Business Case 业务案例**

**8.1 修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 启动阶段1.01 | 2021年3月27日 | 创建业务案例 | 胡若晴 |
| 启动阶段1.04 | 2021年3月27日 | 修改业务案例 | 胡洁珏 |

**8.2 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。

**8.3 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。

**9 Glossary术语表**

**9.1 修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 启动阶段1.03 | 2021年3月27日 | 创建术语表 | 莫振威 |

**9.2 图表**

|  |  |
| --- | --- |
| **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)中各部分（各[子系统](https://www.wiki-wiki.top/baike-%E5%AD%90%E7%B3%BB%E7%BB%9F)）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))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)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 |

**10 Development plan开发计划**

**10.1 修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 启动阶段1.05 | 2021年3月27日 | 创建开发计划 | 崔子潇 |

**10.2 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。

**11 风险列表和风险管理计划**

**11.1 修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 启动阶段1.00 | 2021年3月27日 | 创建风险列表和风险管理计划 | 杨宗恺 |

**11.2 业务风险**

软件有可能不能满足客户的需求，或者客户需求会变更，原来设计的功能不再是主要需求。

**管理计划：**使用快速迭代开发方法，及时与客户进行交互，明确每一次迭代的需求，快速的设计软件功能满足客户变化的需求。

**11.3 技术风险**

软件有可能存在在不同平台下的可移植性问题。

**管理计划：**尽量使用在不同环境和平台下都通用的库和函数，减少不兼容性。

**11.4 资源风险**

软件有可能会占用过多内存，导致计算时间过长，或者占据过多计算资源。

**管理计划：**尽量优化算法，降低计算的时间复杂度和耗费的资源。

**11.5 进度风险**

软件有可能不能在规定进度下完成所有的需求。

**管理计划：**对开发计划进行适当的规划，同时对开发人员进行合适的管理。