第二次系统与设计分析作业

组号	姓名	学号
0	郑有为	19335286

1. Answer the following questions:

a) What is use case? What should be included in a use case? Why we should use use case in system development ?

- What is use case： Use case，即用例，是文本形式的情节描述，用以说明某参与者使用系统以实现某些目标，广泛应用于需求的发现和记录工作中。其本质是通过编写使用系统实现用户目标的情节来发现和记录功能性需求。一般有摘要、非正式和详述三种表示形式。
- What should be included in a use case： 用例是一组相关的成功和失败场景的集合，其中场景指参与者和系统之间的一系列特定的活动和交互。
- Why we should use use case in system development：
    1. 用例可以使得定义和评审变得更为简单，它使领域专家或需求提供者能够自己编写（或参与编写）用例。
    2. 由于在使用用例的时候，通常是以用户为主体，更强调了用户的目标和观点，有利于用户的参与，降低了失败的风险。
    3. 能够根据需要对复杂程度和形式化程度进行增减调节。

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

1. 这些影响是相互关联的。用例模型中用例文本编写的对象以及其相关的属性会、与领域模型会有所关联，而用例文本的规定的范围、目标、参与者、特性等则会影响到设想，它的术语，属性及其验证则是词汇表的重要内容，用例的非功能性需求，质量属性则在补充性规格说明中有所阐释，当然，这也会影响到最后的编写代码。这只是说明了用例对其他部分的影响。设想，业务规则，词汇表等也会有相互影响的关系，这决定了我们必须考虑这些影响因素，不能忽视任何一个部分的影响。

2. 并且，我们还会迭代地进行一些过程，前一个周期不同部分（阶段）的相关内容，也会对现在这个周期的不同部分（阶段）的内容产生影响（至少我们必须考虑我们前一个周期的成果，在此基础上不断补充和修改）。

c) 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.

我们以教材P46的UP工件关系(UP Artifacts Influences)示例(图6-1)为例，从此图我们可以总结出各科目(Discipline)之间工件的依赖关系:

- UP工件的影响关系贯穿UP科目中的业务建模、需求和设计。
    1. 业务建模：
        1. 领域模型可以利用用例模型中总结出的概念和行为，比如用例文本中给出的对象、属性和对象关联性。 解释：业务建模的制品为领域建模产品，其目的是使得领域中的重要概念可视

化，用例模型中的对象属性，行为和关联性有利于领域模型的完善和合理性，勾勒出领域模型的重点部分。

2. 需求：
    1. 领域内容从用例图开始，融入用例模型中； 解释：用例图能够帮助开发者和受众更好沟通和理解项目，开发者将领域的内容导入用例图中，作为整个用例模型构建的基础。
    2. 高级目标和用例图是用例文本创建的输入，输入的内容可以包括用例名称、用例的行为；解释：用例文本是需求阶段得出的核心工件，是对用例图的文字性、系统性的总结。
    3. 我们可以通过提炼用例文本中的系统事件，得到系统顺序图； 解释：系统顺序图（SSD）是为阐述与所讨论系统相关的输入和输出事件而快速、简单地创建的制品。表示对于用例的一个特定场景，外部参与者产生的事件，其顺序和系统之内的事件。
    4. 基于系统顺序图得到系统的操作，作为操作契约的输入和基础； 解释：操作契约是用例描述的补充，在细化阶段引入，针对于复杂或微妙的系统描述。
    5. 根据需求分析的范围、目标、参与者和特性，修正和完善出项目的设想； 解释：建立项目的共同设想是初始阶段的工作，总结需求分析能澄清设想。
    6. 总结和归纳用例文本中出现的术语、对象属性和验证，得到一份项目的词汇表； 解释：词汇表包括关键领域术语和数据字典，是对用例中概念的收集与解释。
    7. 分析用例模型之外的非功能性需求和质量属性等，做出需求的划分和项目的补充性规格说明。 解释：项目的需求分为功能性和非功能性，补充性规格说明考量非功能性需求和质量属性，记录了所有未在用例中描述的需求，例如FURPS+分类模型中对应的因素。
3. 设计：
4. 根据用例模型得到设计模型，需求阶段的用例模型是整个设计阶段工作的基础。 解释：分析模型是概念模型，是系统的一个抽象，并回避了实现问题；设计模型是物理模型，是实现的蓝图。 除此之外，用例还会影响许多其他的分析、设计、实现、项目管理和测试工件。

d) Use cases can be in three formats, what are they? What are the differences?

- 用例能够以不同形式化程度或格式进行编写：
    - 摘要——简洁的一段式概要，通常用于主成功场景。在早期需求分析过程中，为快速了解主题和范围时使用。
    - 非正式——非正式的段落格式。用几个段落覆盖不同场景。在早期需求分析过程中，为快速了解主题和范围时使用。
    - 详述——详细编写所有步骤及各种变化，同时具有补充部分，如前置条件和成功保证。确定并以摘要形式编写了大量用例后，在第一次需求讨论会中，详细地编写其中少量的具有重要架构意义和高价值的用例
- 不同点：三种形式从简洁到详细。摘要只需一段简介，非正式描述需要进一步用几段来对多方面概括，而详述则是深入细节详细编写所有可能的情况、每一个步骤和变化。

e) What techniques can be used to find use cases ? Describe how to use these techniques.

- 老板测试、EBP测试、规模测试、合理的违例等。

1. 老板测试： 如果所做的用例不会令老板高兴，往往意味着该用例与达到可量化价值的结果没有太大关系。这也许是更低级别目标的用例，但不是需求分析所适用的级别。 但是，这不意味着要忽略在老板测试中失败的用例，这些用例可能是需要实现的重点和难点。

2. EBP测试： EBP即**基本业务过程**（Elementary Bussiness Process），是源于业务过程工程领域的术语：一个人于某个时刻在一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，并以持久状态留下数据。 我们应该关注反映EBP的用例，它们对于业务价值而言是可量化的。
3. 规模测试： 用例很少由单独的活动或步骤组成，相反，用例通常包含多个步骤，在详述形式的用例通常需要3~10页文本。用例建模中的一个常见错误是仅将一系列相关步骤中的一个步骤定义为用例，例如将*输入商品ID*定义为用例。
4. 测试的合理违例： 尽管应用中主要用例的定义和分析可以满足上述测试，但是常常会出现例外。 有时，需要为子任务或常规EBP级别用例中的步骤编写单独的子功能级别用例。例如，诸如"信用卡支付"等子任务或扩展可能在多个基本用例中出现。如果有这种现象，即使不能真正满足EBP和规模测试，但也需要考虑将其分离为单独的用例，并且将其连接到各个基本用例上，以避免文字上的重复。

二、项目进展

The collection of all your artifacts you have done for your inception phase of your project in the following.

概览

1. 设想
2. 业务规则
3. 词汇表
4. 用例模型
5. 补充性规格说明（新增）

本次主要更新了补充性规格说明

设想

简介

我们希望开发一个面向对象的类库，我们称之为网格应用程序开发系统，该类库能提供网格应用程序编辑、存储、执行、计算、可视化等基本功能。

用户概要

该产品面向的用户为使用c++创建类的开发人员、使用c++类库来建立完整应用程序的开发人员、需要应用网格图进行2D、3D建模的工程师和研究人员。

用户目标

- 医学工作者：研究某种疾病传播网络模型
- 机械工程师：研究热量在某物理设备中的传播
- 建筑师：建立一个建筑模型，测定其受力情况
- 网络管理：控制流量，确定数据流方向

产品概览

在网格应用程序系统中，我们提供文件输入接口和文件数据解析功能，用户可以将网格数据文件导入到程序中，并将这些数据以一定的数据结构存放在内存中。用户可以用一些系统内核功能定义一些函数以便处理数据和对网格进行计算，除此之外用户还可以利用一些算法来对数据结构进行处理。在系统设计中提供用户算法的循环、分支机制，使用户能够灵活定义和处理网格结构。

我们的系统以用户预定义的格式将模拟结果输出到一些处理工具。这些工具可以将模拟结果可视化并展示。

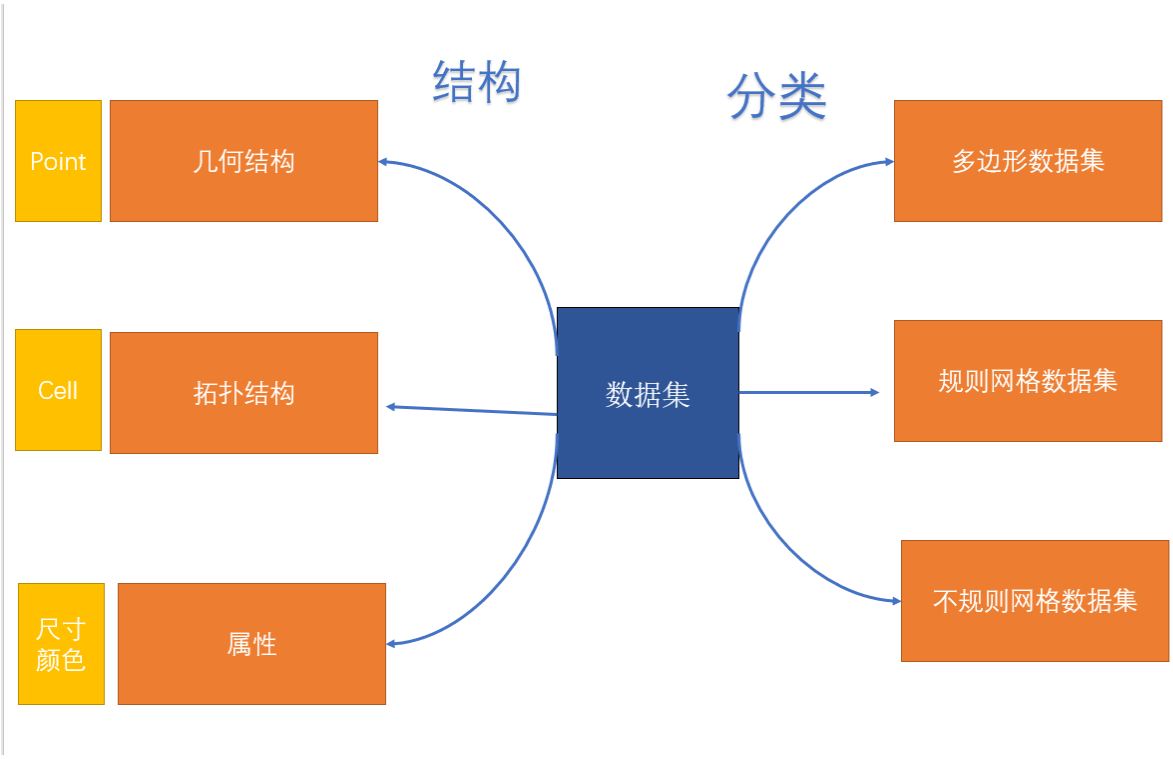
我们的系统提供基本的存储、IO等功能，支持某些平台和系统（windows/Linux、c/c++/python)下的网格应用程序的编程和执行。我们设计的系统允许用户用C/ c++之类的编程语言创建网格应用程序。并为用户程序提供一个执行环境。实现这个环境的工作主要是为网格程序实现一个运行时类库。

我们的系统可以应用于计算机图形学、图像处理和三维可视化等领域，独立于系统的图形界面接口，可嵌入到其它相关软件中。同时开发人员可以基于系统的独立类库开发自己的库函数，拓展应用范围。

系统特性概要

- 在windows/Linux下运行
- 允许使用c/c++/python进行编程
- 用户将文本文件、二进制文件作为输入，得到网格数据
- 以一种高效的数据结构在内存中存储网格结构
- 用户可以在代码中定义函数、算法，对网格做运算处理
- 系统提供内核循环、分支操作
- 系统由c/c++实现，可以在多核CPU上运行
- 系统允许用户跟踪进程性能

数据结构概况



业务规则（领域规则）

规则列表

ID	规则	可变性	来源
规则1	mesh类库是开源的放在gitee，但使用时需要声明原创及附上链接。	中，如果传播性高则不需要附上链接	专利权
规则2	存储mesh时需要由user确认存储格式，否则自动保存为text格式。	低	
规则3	user不允许对程序进行破坏性攻击行为。	低	程序安全
规则4	我们应该为域程序员提供一个域应用程序模板。网格应用程序通常由网格声明、网格元素上的核函数定义、网格的所有结构上的操作和一些io操作组成	中，有时候用户会自己定义另一种操作	网格的基本操作
规则5	所有的$结束标记都必须在一个新的行上开始	低	格式要求
规则6	类库应该在网格上实现内核操作的循环	低	类库要求
规则7	类库应该输出到文本（或二进制)文件，以某些用户定义或预定义格式保存计算(模拟）结果。	低	类库要求

词汇表

术语	定义和信息	格式/验证规则	别名
MADS	Mesh Application Development System		网格应用程序开发系统
静态网格数据	该网格点、边等信息在定义之后不会再被改变，类似于一个常量		
动态网格数据	该网格点、边等信息在定义之后可以在程序执行过程中被改变，类似于一个变量		

术语	定义和信息	格式/验证规则	别名
核函数	若支持向量机的求解只用到内积运算，而在低维输入空间又存在某个函数 K(x, x′)，它恰好等于在高维空间中这个内积，那么这样的函数 K(x, x′) 称为核函数	支持向量机通过某非线性变换 φ(x)将输入空间映射到高维特征空间,而低维输入空间又存在某个函数K(x, x′)使得：K(x, x′) = <φ(x)·φ(x′) >	径向基函数（Radial Basis Function 简称 RBF)
聚类	在此开发过程中主要指基于网格的聚类，即利用属性空间的多维网格数据结构，将空间划分为有限数目的单元以构成网格结构		
错误代码	提供出错原因、位置	error:cause/filename/position	

用例模型

1. 系统参与者

1.1开发人员：调用MADS类库进行编程的人员

1.2类库：指提供底层Mesh操作的类库

2. 用例概览

版本与配置

- 2.1. MADS的版本与配置
    - 2.1.1 本机配置检查
        - 解释：例如检测当前OS架构和版本、编译器版本、是否安装/安装了哪些可用的 网格可视化程序和其他MADS需要使用的第三方类库或软件、网络的状况。
    - 2.1.2 版本更新检查
        - 解释：启动时检查MADS版本，询问是否升级（考虑到后续可能会打补丁什么的）

使用帮助

- 2.2. MADS的使用帮助
    - 2.2.1 查询使用教程（帮助文档）
        - 解释：虽然听起来怪怪的，但是对于第一次使用MADS的用户，这个过程是应该包含进“用户的软件使用过程中”。
    - 2.2.2 查询选项帮助

        - 解释：例如Linux上的很多软件会提--help，选项指程序编译、链接、运行等过程中使用的-参数。

使用日志

- 2.3 用户使用日志的记录和查询
    - 解释：例如用户使用此功能查询上次操作的文件、异常关闭前文件的状态。

数据文件操作

- 2.4. MADS与网格数据文件
    - 2.4.1 检查网格数据文件内容的（格式）正确性
    - 2.4.2 将特定字节串转换为特定数据类型
    - 2.4.3 以给定格式读取网格数据文件并写入内存
    - 2.4.4 以给定格式将内存中网格数据写入文件

数据可视化

- 2.5. MADS与可视化（似乎是专业网格可视化软件的工作）
    - 2.5.1 显示静态网格数据
    - 2.5.2 显示动态网格数据
        - 解释：例如一块材料温度随时间变化的热力图。

基本数据结构和基本操作

- 2.6. MADS类库操作（或许根本没有基本数据类型，所有网格数据的建立都是用户用Set、Map、Dat等定义的）
    - 2.6.1 使用系统提供的基本网格数据类型（2D/3D）进行建模（正方体、三角形、球等）
    - 2.6.2 使用MADS提供的通用函数库操作数据
        - 2.6.2.1 平移
        - 2.6.2.2旋转
        - 2.6.2.3 对称翻转
        - 2.6.2.4 缩放
        - 2.6.2.5 合并点
        - 2.6.2.6删除点、边、面和体
        - 2.6.2.7 增加点、边、面和体
    - 2.6.3 使用者使用类库的模板自定义一种网格数据结构。
        - 解释：自定义一种不在基本类型中的单元结构，可以是继承于基本结构、可以是基本结构的组合，也可以直接通过定义Set和Map等来创建结构

- 2.6.4 使用者定义基于网格的操作（查找，遍历，比较，运算等）、核函数和特定算法（求权，排序，聚类等）
- 2.6.5 使用者获取Mesh基本的属性数值
  - 2.6.5.1计算面的数量
  -
  - 2.6.5.2 点的度数

3. 系统用例

UC1：以给定格式读取网格数据文件并写入内存(2.4.1)

- 主要参与者：开发人员、类库
- 前置条件：格式合法且网格数据文件存在
- 主成功场景：
  1. 开发人员调用类库的载入函数，传进参数：文件绝对地址、格式类型。
  2. 查询文件是否存在，存在。
  3. 尝试打开文件，成功。
  4. 本函数调用同类中的确认文件格式合法的函数，确认文件格式无误。
  5. 从数据段开始读取。
  6. 读入一个字节串，将其作为参数调用同类中的串转换函数，并将返回值赋给类中对应成员。
  7. 重复5.直至读入结束符。
  8. 关闭文件。
  9. 返回成功值给调用函数
- 扩展：
  - a.文件访问失败 处理过程：
    1. 调用stat函数查询文件状态，获得对应的错误代码。
    2. 根据错误代码返回对应的错误代码给开发人员。
  - b.文件打开失败 处理过程：
    1. 返回对应的错误代码给开发人员
  - c.格式错误 可能情况：
    1. 指定的格式和文件格式不符。
    2. 文件格式部分已经被破坏
    3. 指定的格式不合法 处理过程： 1.返回对应错误代码给开发人员
  - d.数据出错 可能情况：
    1. 数据段残缺
    2. 非法比特串 处理过程：
    3. 记录下路径、文件名、出错位置行标号以及错误类型
    4. 打印出错误信息
    5. 返回错误代码给调用者

UC2: 平移(2.6.2.1)

- 主要参与者：开发人员、类库
- 主成功场景：

    1. 编程者确定/选中需要平移的Mesh主体。
    2. 编程者输入平移的主体、主体维度信息、平移的方向以及距离等参数，并调用系统类库中的平移函数。
    3. 系统查询类库，并对被操作主体执行平移操作。
    4. 平移成功，修改后的主体被返回，控制台打印出操作成功的信息。

UC3: 旋转(2.6.2.2)

- 主要参与者：开发人员、类库
- 主成功场景：

    1. 编程者确定/选中需要旋转的Mesh主体。
    2. 编程者输入旋转的主体、主体维度信息、旋转的方向以及角度等参数，并调用系统类库中的旋转函数。
    3. 系统查询类库，并对被操作主体执行旋转操作。
    4. 旋转成功，修改后的主体被返回，控制台打印出操作成功的信息。

UC4: 对称翻转(2.6.2.3)

- 主要参与者：开发人员、类库
- 主成功场景：

    1. 编程者确定/选中需要翻转的Mesh主体。
    2. 编程者输入旋转的主体、主体维度信息、翻转的对称轴位置、方向等参数，并调用系统类库中的翻转函数。
    3. 系统查询类库，并对被操作主体执行翻转操作。
    4. 翻转成功，修改后的主体被返回，控制台打印出操作成功的信息。

UC5: 缩放(2.6.2.4)

- 主要参与者：开发人员、类库
- 主成功场景：

    1. 编程者确定/选中需要缩放的Mesh主体。
    2. 编程者输入缩放的主体、主体维度信息、缩小或放大的选择、缩放中心点的位置、缩放倍率等参数，并调用系统类库中的缩放函数。
    3. 系统查询类库，并对被操作主体执行缩放操作。
    4. 缩放成功，修改后的主体被返回，控制台打印出操作成功的信息。

UC6: 合并点(2.6.2.5)

- 主要参与者：开发人员、类库

- 主成功场景：

    1. 编程者确定/选中需要合并的点集。
    2. 编程者输入该点集、点集所在的维度信息、合并后点的位置等参数，并调用系统类库中的合并点函数。
    3. 系统查询类库，并对被选中的点集执行合并操作。
    4. 合并成功，返回合并点，控制台打印出操作成功的信息。

UC7: 删除点、边、面和体(2.6.2.6)

- 主要参与者：开发人员、类库
- 主成功场景：

    1. 编程者确定/选中需要删除的点、边、面或体（为方便描述，统称为被删除主体）
    2. 编程者输入被删除主体、该主体所在的维度信息等参数，并调用系统类库中删除主体的函数。
    3. 系统查询类库，并对被删除主体执行删除操作。
    4. 删除成功，控制台打印出操作成功的信息。

UC8: 增加点、边、面和体(2.6.2.7)

- 主要参与者：开发人员、类库
- 主成功场景：

    1. 编程者确定/选中需要增加的点、边、面或体（为方便描述，统称为新添主体）。
    2. 编程者输入新添主体、该主体所在的维度信息、主体将要放置的位置和方位信息等参数，并调用系统类库中新添主体的函数。
    3. 系统查询类库，并对新添主体执行添加操作。
    4. 添加成功，控制台打印出操作成功的信息。

UC9: 计算面的数量(2.6.5.1)

- 主要参与者：开发人员、类库
- 主成功场景：

    1. 编程者确定/选中需要计算的主体（维度在2维及以上，包含一个或多个面）。
    2. 编程者输入主体、主体所在的维度信息等参数，并调用系统类库中计算面数量的函数。
    3. 系统查询类库，并根据主体内部数据结构信息执行计算其面的数量的操作。
    4. 返回计算结果，该结果输出到编程者指定的文件中或者直接输出到控制台。

UC10: 点的度数(2.6.5.2)

- 主要参与者：开发人员、类库
- 主成功场景：

    1. 编程者确定/选中需要计算的主体（包含一个或多个顶点）。
    2. 编程者输入主体、主体所在的维度信息等参数，并调用系统类库中计算点的度数的函数。

3. 系统查询类库，并执行计算主体点的度数的操作。
4. 返回计算结果，该结果输出到编程者指定的文件中或者直接输出到控制台。

UC11：2.4.4 以给定格式将内存中网格数据写入文件

- 主要参与者：开发人员、类库
- 前置条件：操作系统的文件系统正常
- 主成功场景： 1.编程者处理好文件的绝对路径格式、文件名和文件类型，以参数形式调用类库的存储函数。 2.类库的存储函数以只读形式打开或者创建文件，以覆盖的形式更新文件内容。 3.读出成功，返回正常存储的代码。
- 扩展：
    - a.对指定路径的文件夹没有操作权限

      1.判断错误并打印错误信息到终端

      2.函数返回错误代码
    - b.存储格式是不支持的类型

      1.打印错误信息到终端。

      2.函数返回错误代码

补充性规格说明

FURPS+需求

功能性

1. 操作，特别是计算，应当支持多线程。
2. 应当支持循环执行编程者定义的操作函数。
3. 遇到错误时应当记录并生成错误文档

可用性

1. 函数的命名应当有**自明性**且**格式统一**。
2. 支持**用户反馈**，收集用户评分和改善意见。

可靠性(安全性)

1. 支持**数据结构的自检查**，遇到错误应提醒上层程序。
2. 支持一定限度的**数据恢复**，在遇到异常退出时，系统能临时保存数据与操作以便于下依次恢复。
3. **代码开放**：源代码向使用者开放，但使用者没有权限修改MADS库，或需要许可才能修改MADS库。

4. **可靠实现**：确保MADS不会因内部漏洞而意外删除，篡改用户数据。

性能

1. 提供**高性能算法支持**和数据结构（例如：堆，AVL树结构），保证系统使用流畅
2. 能够估计计算的规模，提供计算性能预估，如在用户在对某大型数据进行某种复杂操作时，**预估计算时间**，并向用户提供（计算过程中）中途取消的功能。
3. 能够估计计算的规模（同上一点），遇到算不了的问题（出于内存限制、计算机计算能力限制等因素），能够**自动终止**计算，返回错误，而不是崩溃。

可支持性

1. 确保系统能够**在Linux上运行**，不与操作系统、C/C++库、编译器等发生冲突。
2. **可扩展函数库**：提供足够的、考虑周全的操作（函数）清单和参数列表（包括数据类型，数据文件，数据大小，操作类别，输出方式等），即便在现阶段不支持或没有能力实现。

开发约束

1. 考虑到高性能的要求以及此类库开发者的实际能力，采用**C/C++实现。**

文档化和帮助

1. 提供**使用时帮助**，动态提供参数列表指引和函数使用说明。
2. 提供**安装帮助手册**
3. 提供容易阅读的函数和参数说明文档（**用户使用教程**），并随每一次MADS的更新而更新。

国际化问题

1. **双语支持**：以英语为基础，提供中文（编码格式要求：UTF-8)的使用时帮助、安装手册、使用教程等。

许可与其他法律问题

1. 简化开发中使用的构件为**开源组件。**
2. **开源协议**：使用GPL开源许可协议或者MPL-2.0，具体开源协议待商定。

操作问题

1. 处理错误：遇到错误时终止当前操作，向用户反馈简要的错误信息及错误原因，记录错误并生成错误文档。
2. 备份情况：当用户的光标离开程序界面十分钟（默认时间），且检测到键盘十分钟内（默认时间）无输入，则自动更新备份当前已命名的文档。对于未命名的文档，自动命名为“unnamed-x.txt”，其中 x 为当前工作目录下所有未命名文件中，该文件的序号；默认输出格式为文本文件，见“业务规则.md”中规则列表说明。