面向对象团队报告

1 目标系统简介

DLMMS 是一个深度学习模型管理系统。主要支持以下几部分功能：

- 支持用户线上托管模型，并支持用户向其他用户分享已在线上托管的模型；
- 支持用户将线上托管的模型部署上线，并对外提供服务；
- 支持监控和维护用户线上模型服务的运行状态；
- 支持实时采集和记录用户线上模型服务日志，并支持生成数据集以供查询。

2 目标系统需求

姓名	学号	工作内容	工作占比
陈泽人	SY2121101		
刘炜	SY2121110		
任婷伊	SY2121113		
王子勤	SY2121117		
李楠	SY2121108		

表 2.1 目标系统需求工作量统计表

2.1 用户故事

DLMMS 系统拥有五种角色：模型拥有者、模型管理者、模型访问者、系统普通用户、未

注册普通游客。其中前三种角色只是相对而言，权限限制模型范围，只针对特定模型拥有模型拥有者、管理者和访问者相应的权限，对其他模型只是系统普通用户的角色。模型访问者的访问权限可以细分为模型访问、外服服务访问、外部 API 访问和数据集访问权限。模型管理者的权限可以细分为模型部署权限、模型修改权限、API 管理权限、访问链接管理权限、数据集管理权限和访问者管理权限。

DLMMS 的功能模块划分为模型部署管理、深度学习网络模型分享管理、深度模型 Docker 环境及健康监控、实时数据采集与管理四部分，详细的用户故事如下所示。

2.1.1 模型部署管理相关用户故事

在模型管理模块，用户可以以压缩包的方式将深度学习网络模型上传到指定目录下，用户成为该模型的模型拥有者。DLMMS 系统提供可视化的方式，为模型拥有者和模型管理者提供查找模型的操作，为模型拥有者提供删除模型操作。模型拥有者和模型管理者也可以移动深度学习网络模型的路径和位置，管理深度学习模型的版本。

模型拥有者和模型管理者可以下载指定版本的深度学习网络模型，还能以命令行或脚本的方式部署已经上传的深度学习网络模型，开放对外服务，并添加相关说明，也可以下线已经部署的深度学习网络模型。

模型拥有者和模型管理者可以查看自己部署的深度学习网络模型，同时其他拥有访问权限的用户可以查看提供对外服务的深度学习网络模型和服务说明。

2.1.2 深度学习网络模型分享管理用户故事

首先，模型拥有者对该模型拥有最高权限，为该模型添加、删除模型管理者。模型的拥有者和模型管理者可以为系统普通用户添加该深度学习网络模型的访问权限，也可以修改模型访问者对该深度学习网络模型的访问权限。

模型拥有者和模型管理者用户可以创建深度学习网络模型的访问链接，并设置该访问链接状态、密码、访问次数和有效期。只要链接处于有效期的开放状态、拥有访问密码，所有用户包括未注册普通游客都可以通过访问链接下载深度学习网络模型。如果链接设置了访问次数，则只有登录的系统用户才可以通过访问链接下载深度学习网络模型。

模型拥有者和模型管理者可以创建已部署的深度学习模型服务的访问 API，可以设置

token 来限制 API 的访问权限和 API 访问状态。模型拥有者和模型管理者用户可以修改已公开的深度学习模型服务的访问 API 状态、访问 token，或是下线已部署的访问 API。只要携带 token，所有用户包括未注册普通游客都可以通过访问 API 调用已部署并公开的深度学习网络模型。

2.1.3 深度模型 Docker 环境及健康监控用户故事

DLMMS 提供 dockerfile 的上传渠道，在指定目录在部署相应的 Docker 环境。同时 DLMMS 也提供表单的方式，模型拥有者和模型管理者可以选择填写参数，自动化生成 dockerfile 文件。

模型拥有者和模型管理者可以查看已部署的深度学习网络模型服务的运行状况，也可以修改已部署的深度学习网络模型服务的运行状况。

2.1.4 实时数据采集与管理用户故事

模型拥有者和模型管理者可以定义已部署的深度学习网络模型服务的采样规则。DLMMS 按照采样规则，为指定深度学习网络模型手机数据。模型拥有者和模型管理者可以查找、下载模型服务收集的数据集，也可以删除某段时间范围内产生的数据集。拥有数据集访问权限的模型访问者也可以查找、下载模型服务收集的数据集。

2.2 EARS

本章节展示了章节 2.1 中用户故事对应的系统需求（EARS），如表 2.2 所示。

用户故事	序号	系统需求	类型
用户可以上传深度学习网络模型	1.1	用户可以上传深度学习网络模型	UB
	1.2	如果用户因为网络连接问题导致上传深度学习网络模型失败，那么网站将显示“上传失败”	UW
用户可以查找上传的	2.1	用户可以查找已上传的深度学习网络模型信息	UB

深度学习网络模型	2.2	当用户查找已上传的深度学习网络模型信息时，用户需要提供查找的关键字	EV
	2.3	如果用户输入的内容不符合要求，那么网站将显示“请输入正确格式”	UW
用户可以删除已上传的深度学习网络模型	3.1	用户可以删除已上传的深度学习网络模型	UB
	3.2	如果删除深度学习网络模型失败，那么网站将显示“删除模型失败”	UW
用户可以管理深度学习网络模型版本	4.1	用户可以修改深度学习网络模型的版本信息	UB
	4.2	当用户修改深度学习网络模型的版本信息时，用户需要提供新的版本信息	EV
	4.3	当用户修改深度学习网络模型的版本信息时，网站将展示之前的版本信息以及修改前后的差异	ST
	4.4	如果修改深度学习网络模型版本信息失败，那么网站将显示“修改模型失败”	UW
用户可以下载已上传的深度学习网络模型	5.1	当用户下载深度学习网络模型时，用户需要选择需要下载的模型	EV
	5.2	如果用户因为网络连接问题导致下载深度学习网络模型失败，那么网站将显示“下载模型失败”	UW
用户可以部署已上传的深度学习网络模型，以便对外提供服务	6.1	用户可以部署已上传的深度学习网络模型	UB
	6.2	如果部署深度学习网络模型失败，那么网站将显示“模型部署失败”	UW
用户可以下线已部署	7.1	用户可以下线已部署的深度学习网络模型	UB

的深度学习网络模型	7.2	如果下线已部署深度学习网络模型失败，那么网站将显示“下线模型失败”	UB
用户可以查找已部署的深度学习网络模型	8.1	用户可以查找已部署的深度学习网络模型	UB
	8.2	当用户查找已部署的深度学习网络模型时，用户需要提供查找的已部署的模型信息关键字	EV
用户可以添加深度学习网络模型的访问权限	9.1	深度学习网络模型的拥有者或管理员可以添加新用户对该模型的访问权限	UB
用户可以修改深度学习网络模型的访问权限	10.1	深度学习网络模型的拥有者或管理员可以修改拥有访问权限的用户对该模型的访问权限	UB
	10.2	如果深度学习网络模型的管理者修改该模型拥有者的访问权限时，那么网站将显示“您无权修改权限”	UW
用户可以创建深度学习网络模型的访问链接	11.1	深度学习网络模型的拥有者或管理员可以创建一个提供模型下载能力的链接	UB
用户可以修改已公开的深度学习网络模型的访问链接状态	12.1	深度学习网络模型的拥有者或管理员可以修改已公开的访问链接状态，包括是否可以访问、过期时间	UB
用户可以通过访问链接下载深度学习网络模型	13.1	外界用户可以通过访问链接下载深度学习网络模型	UB
	13.2	如果下载深度学习网络模型失败，那么网站将显示“下载失败”	UW

	13.3	如果访问链接已过期或不再提供访问权限，那么网站将显示“该链接已经失效”	UW
用户可以创建已部署的深度学习模型服务的访问 API	14.1	深度学习网络模型的拥有者或管理员可以创建一个提供调用能力的 API 请求	UB
用户可以修改已公开的深度学习模型服务的访问 API 状态	15.1	深度学习网络模型的拥有者或管理员可以修改已公开的深度网络学习模型的访问 API 状态，包括是否可以访问、过期时间	UB
用户可以通过访问 API 调用已部署的深度学习网络模型	16.1	外界用户可以通过访问 API 调用深度学习网络模型	UB
	16.2	如果外界用户提供的参数不合法，那么访问 API 将返回“400 Bad Request”状态码	UW
	16.3	如果访问 API 已过期，那么访问 API 将返回“404 Not Found”状态码	UW
	16.4	如果访问 API 不再提供访问权限，那么访问 API 将返回“403 Unauthorized”状态码	UW
用户可以查看已部署的深度学习网络模型服务的运行状况	17.1	用户可以查看一个已部署的深度学习网络模型服务的运行状况	UB
用户可以修改已部署的深度学习网络模型服务的运行状况	18.1	用户可以重新分配已部署的深度学习网络模型服务运行时的资源	UB
	18.2	当用户重新分配已部署的深度学习网络模型服务运行需要的资源时，用户需要提供该服务运行时消耗	EV

		的资源数	
	18.3	用户分配的资源超过用户的资源上限时，那么网站将显示“用户资源超过上限”	UW
	18.4	用户分配的资源超过系统的资源上限时，那么网站将显示“系统资源超过上限”	UW
	18.5	当用户重新分配深度学习网络模型服务的资源并且分配的资源量未超过用户资源和系统资源的上限时，那么系统会停止该服务，重新分配资源后重启服务	ST
用户可以定义已部署的深度学习网络模型服务的采样规则	19.1	户可以定义已部署的深度学习网络模型服务的采样规则	UB
	19.2	果上传深度学习网络模型服务采样规则失败，那么网站将显示“上传失败”	UW
	19.3	定义深度学习网络模型服务的采样规则不符合规范标准，那么网站将现实“不符合规则规范”	UW
	19.4	当用户上传深度学习网络模型服务采样规则符合规范时，那么系统将停止该服务，加载用户新定义的采样规则并重启服务	ST
用户可以查找模型服务收集的数据集	20.1	用户可以搜索模型服务已收集的数据集	UW
	20.2	如果用户没有对目标模型所收集的数据集有检索权限，那么网站会显示“没有搜索权限”	UW

用户可以下载模型服务收集的数据集	21.1	用户可以下载模型服务已收集的数据集	UB
	21.2	如果因为网络连接问题导致下载失败，那么网站会显示“下载失败”	UW
	21.3	如果用户没有对目标模型所收集的数据集有下载权限，那么网站会显示“没有下载权限”	UW
用户可以删除某段时间范围内产生的数据集	22.1	用户可以删除某段时间范围内产生的数据集	UB
	22.2	如果用户删除某段时间范围内产生的数据集失败，网站会显示“删除失败”	UW
	22.3	如果用户没有对目标模型所收集的数据集有删除权限，那么网站会显示“没有删除权限”	UW

表 2.2 目标系统需求（EARS）表

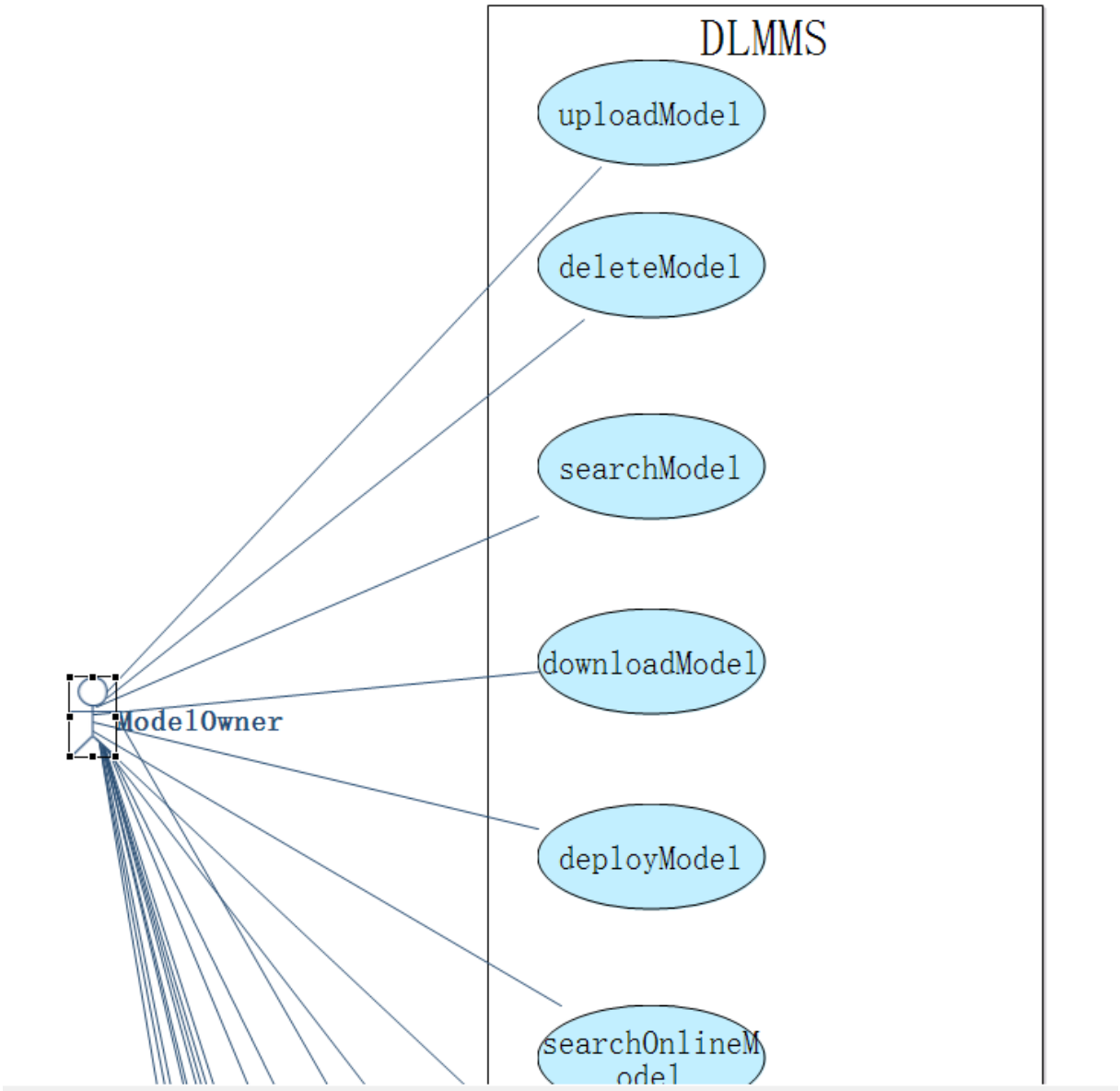
3 目标系统 UML 最终需求模型

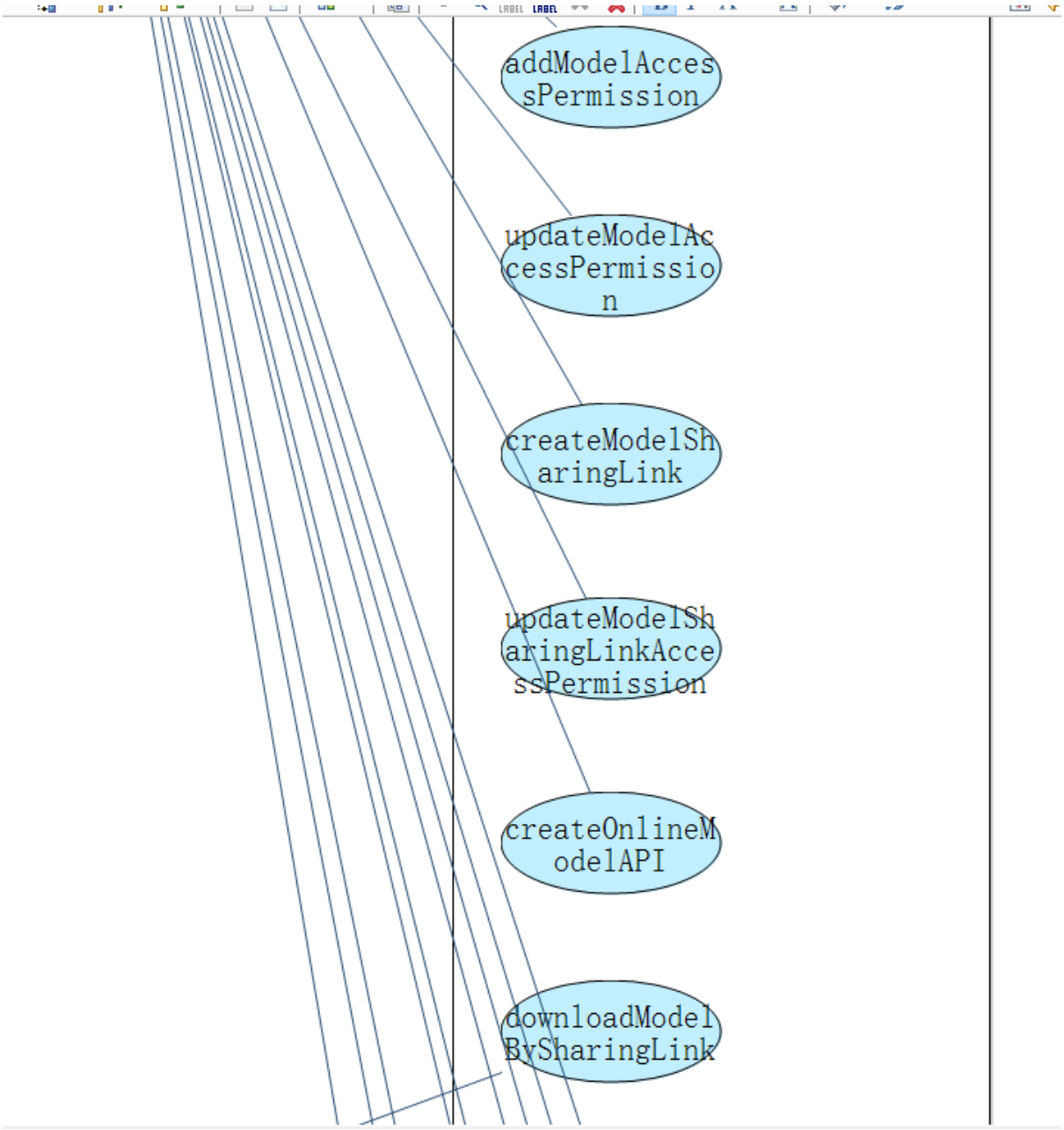
姓名	学号	工作内容	工作占比
陈泽人	SY2121101		
刘炜	SY2121110		
任婷伊	SY2121113		
王子勤	SY2121117		
李楠	SY2121108		

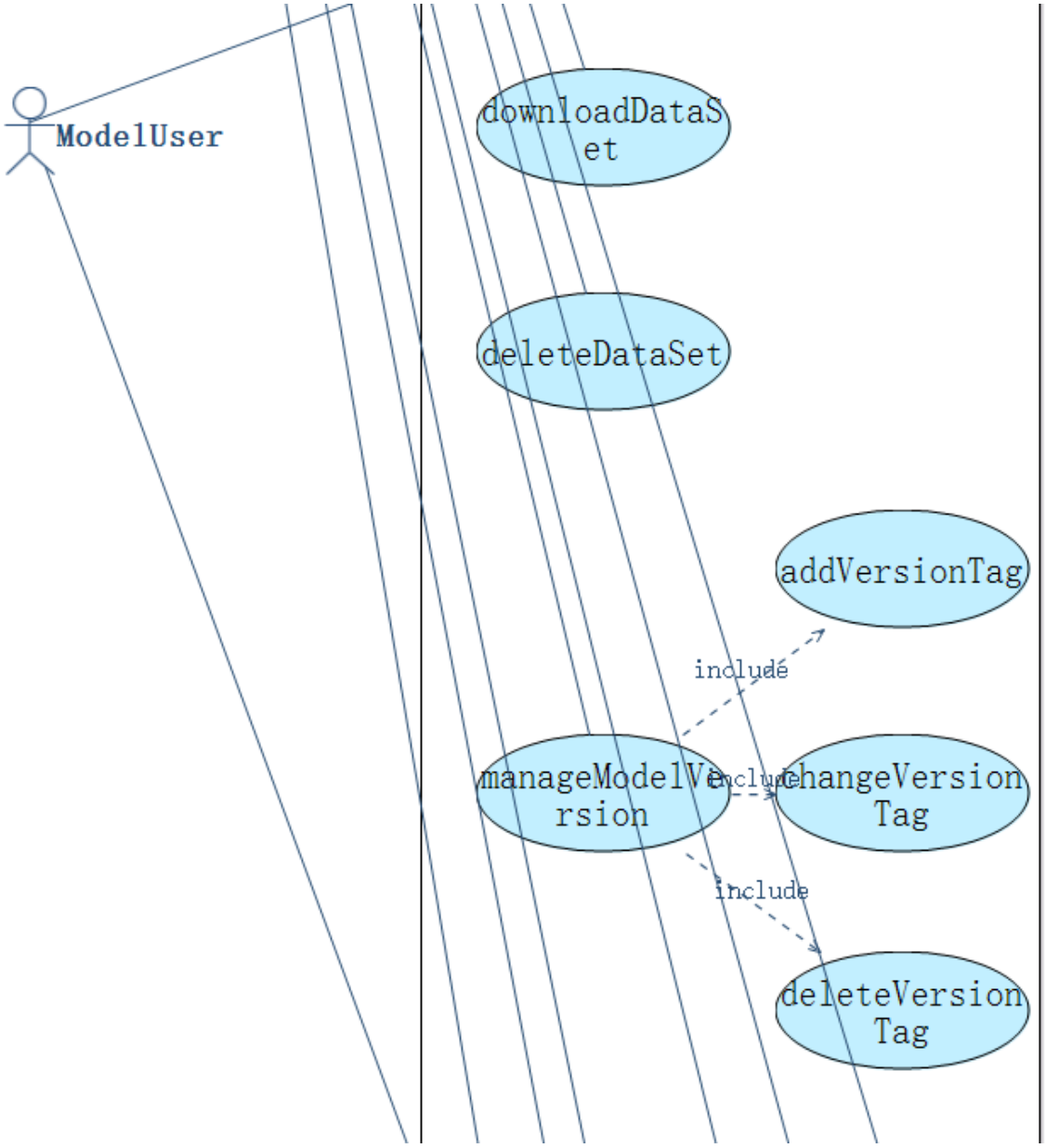
表 3.1 需求模型工作量统计表

3.1 用例图

我们根据用户故事设计了如下的用例图：




addModelAccessPermission
updateModelAccessPermission
createModelSharingLink
updateModelSharingLinkAccessPermission
createOnlineModelAPI
downloadModelBySharingLink


ModelUser
downloadDataSet
deleteDataSet
addVersionTag
include
manageModelVersion
include
changeVersionTag
include
deleteVersionTag

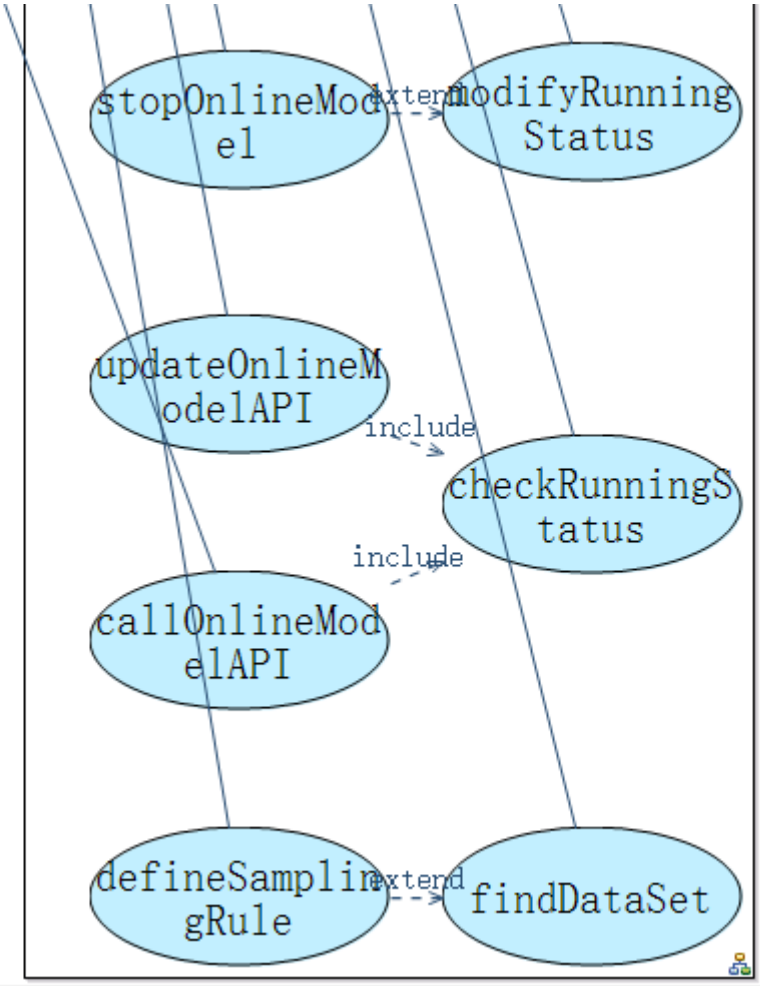


图 3.1 DLMMS 用例图

第一部分，模型拥有者作为参与者，可以操作模型，根据这一部分的需求，我们设计了如下的用例：新增模型，删除模型，搜索模型，下载模型，同时也会进行模型的版本管理，包括新增/修改/删除版本信息等。

第二部分，模型拥有者作为参与者，可以对上线的模型进行操作，以供普通用户使用，根据这一部分的需求，我们设计了如下的用例：部署模型，更新（新建）模型的访问权限，新建模型的访问链接以及更新改访问链接的权限，更新（新建）模型的在线访问端口，查看/修改模型运行状态，下线已部署模型等。

第三部分，模型拥有者作为参与者，可以对该系统拥有的数据集进行一系列的操作，根据这一部分的需求，我们设计了如下的用例：下载数据集，删除数据集，查找数据集，同时也定义从数据集中挑选数据的采样规则。

第四部分，模型使用者作为参与者，主要可以通过网站提供的模型下载链接进行模型的下

载/访问，以及使用网站提供的模型 API。

3.2 系统顺序图

这里我们根据设计的系统用例图，选择了一些用例对应的顺序图设计如下。

3.2.1 调用在线模型 API

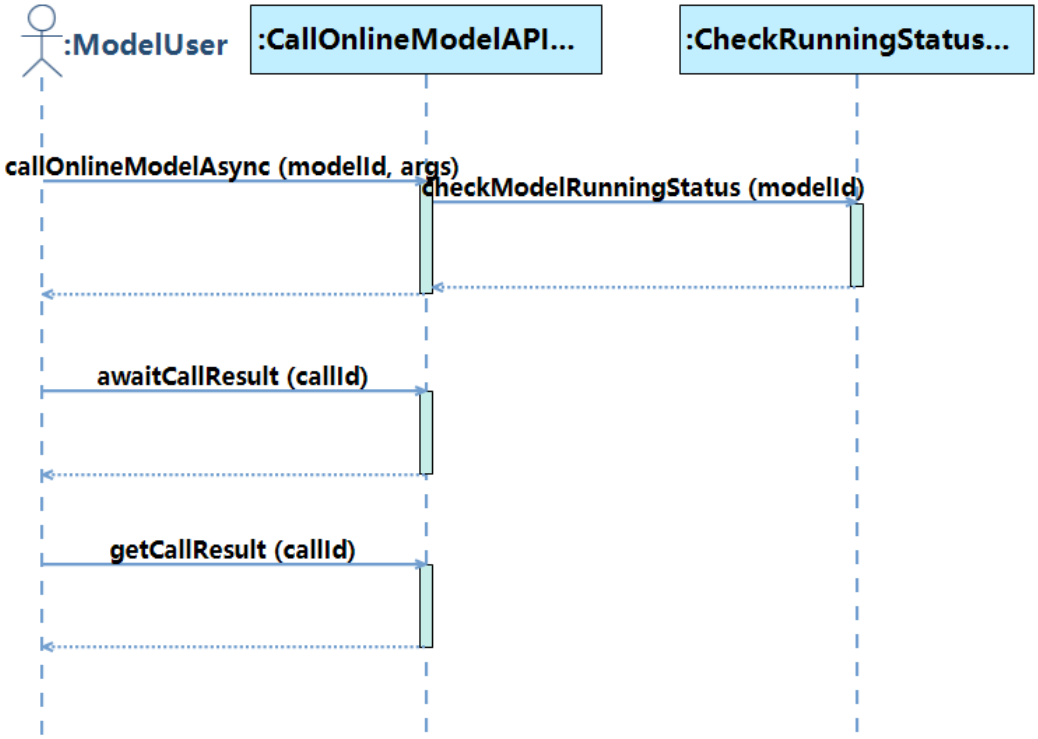


图 3.2 调用在线模型 API 顺序图

1) 这里用户（模型使用者）首先向系统提出 api 的调用申请

2) 接到用户申请后，系统会首先判断模型的运行状况，若模型正常运行，则系统不会返回异常。

3) 用户这边移步提交 api 的调用申请，系统会根据需求返回相应的调用结果。

3.2.2 查找数据集

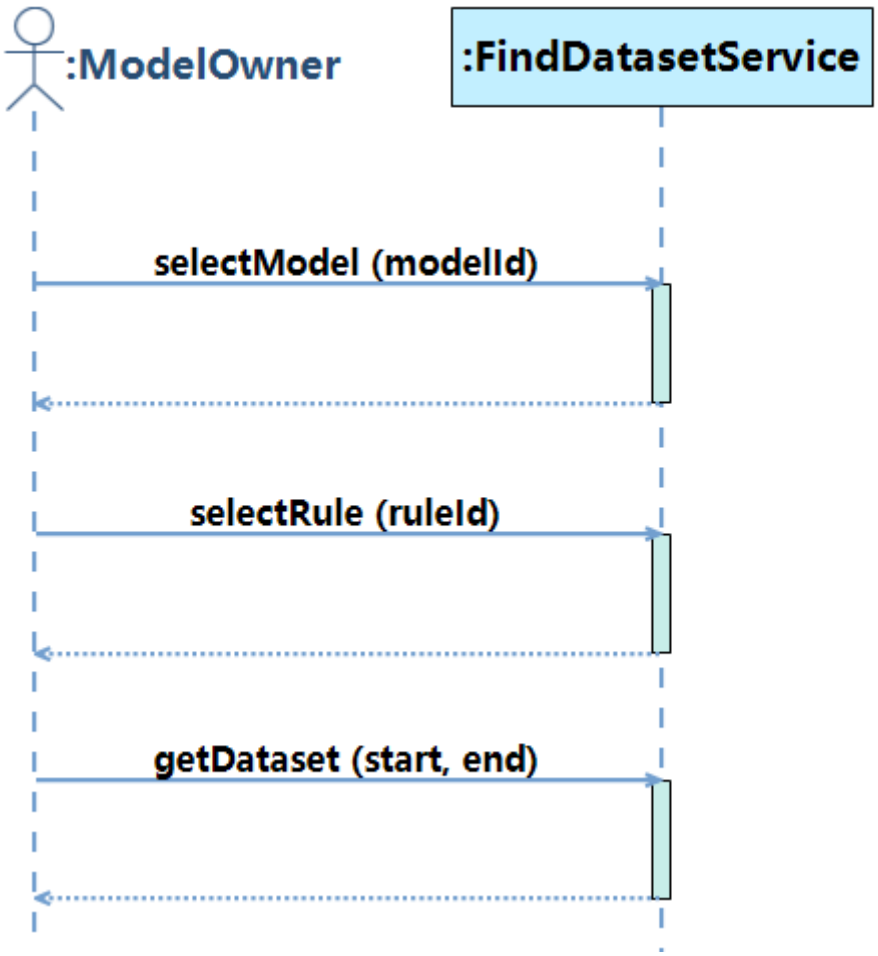


图 3.3 查找数据集顺序图

1) 这里模型的拥有者首先向系统发出模型查找的请求。

2) 模型拥有者会根据模型去筛选对应的数据集，减少搜索范围。

3) 接着会通过输入采集规则，进一步筛选数据集。

4) 最后，系统会返回需要的数据集，以供模型管理者下载使用数据。

3.2.3 修改模型运行状态

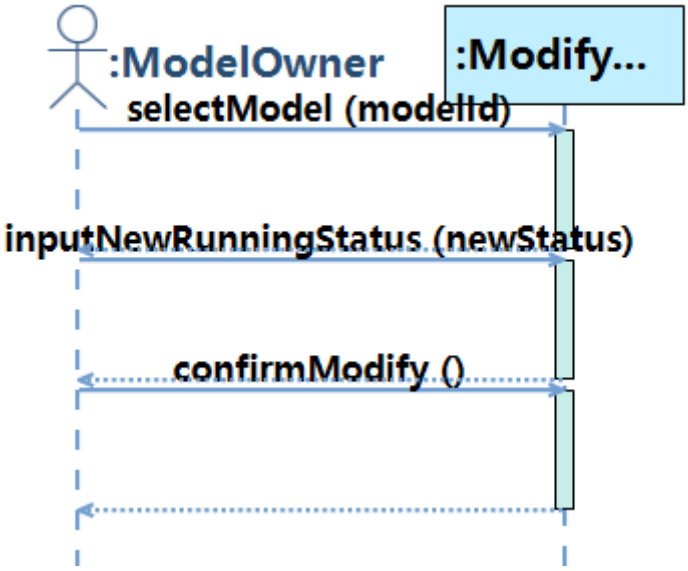


图 3.4 修改模型运行状态顺序图

1) 模型管理者首先输入模型 id/名字，定位模型，系统会更具需求返回对应的模型。

2) 接着管理者会根据需求选择新的模型运行状态来对原有权限进行修改。

3) 修改后，系统会返回一个修改的结果，以通知管理者是否修改成功，以及修改后的模型运行状态。

3.2.4 更新模型的获取权限

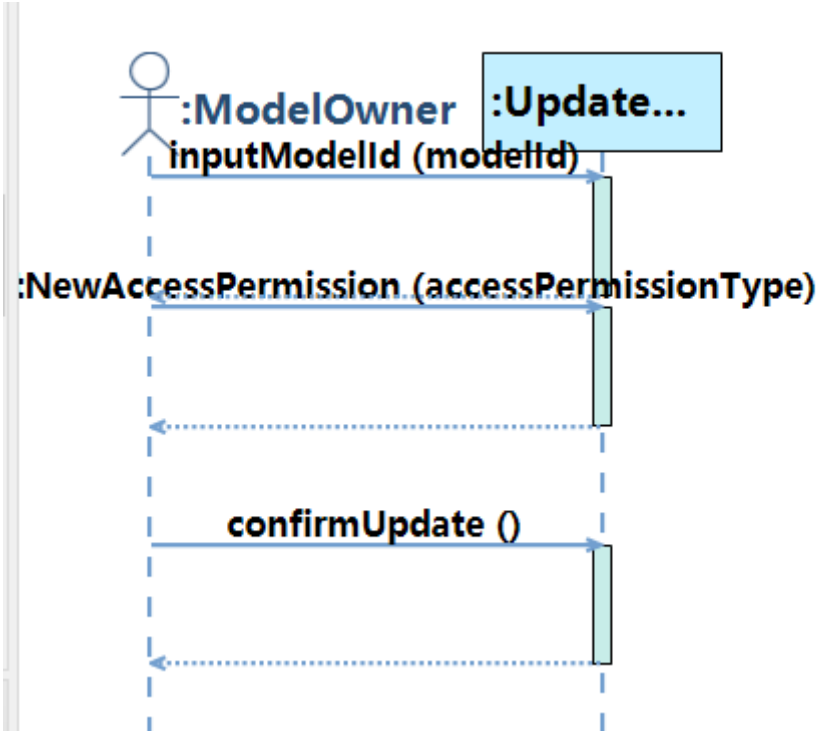


图 3.5 更新模型的获取权限顺序图

1) 首先模型管理者会输入模型的 id 来定位模型，系统会返回相应的模型。

2) 接下来模型管理者会输入新的模型获取权限，进行权限更新。

3) 系统会返回模型权限的更新结果，提醒模型管理者模型权限是否更新成功。

3.2.5 更新模型

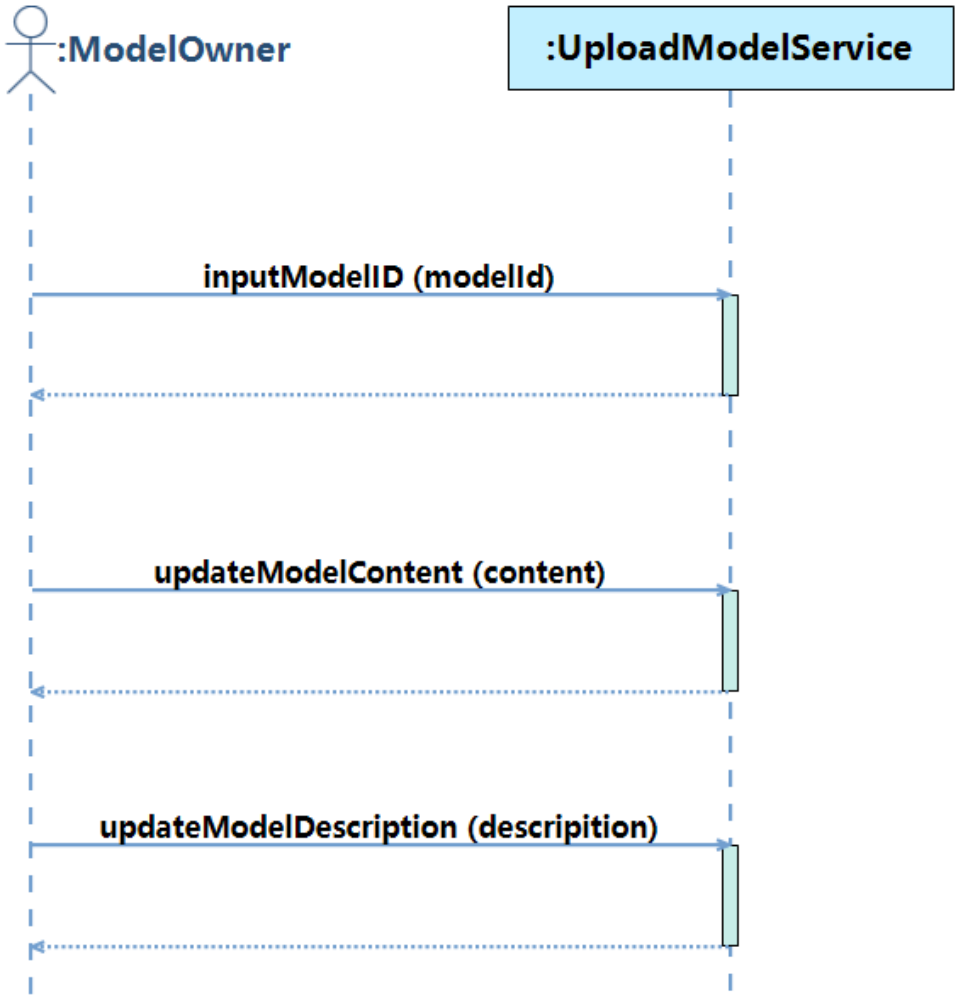


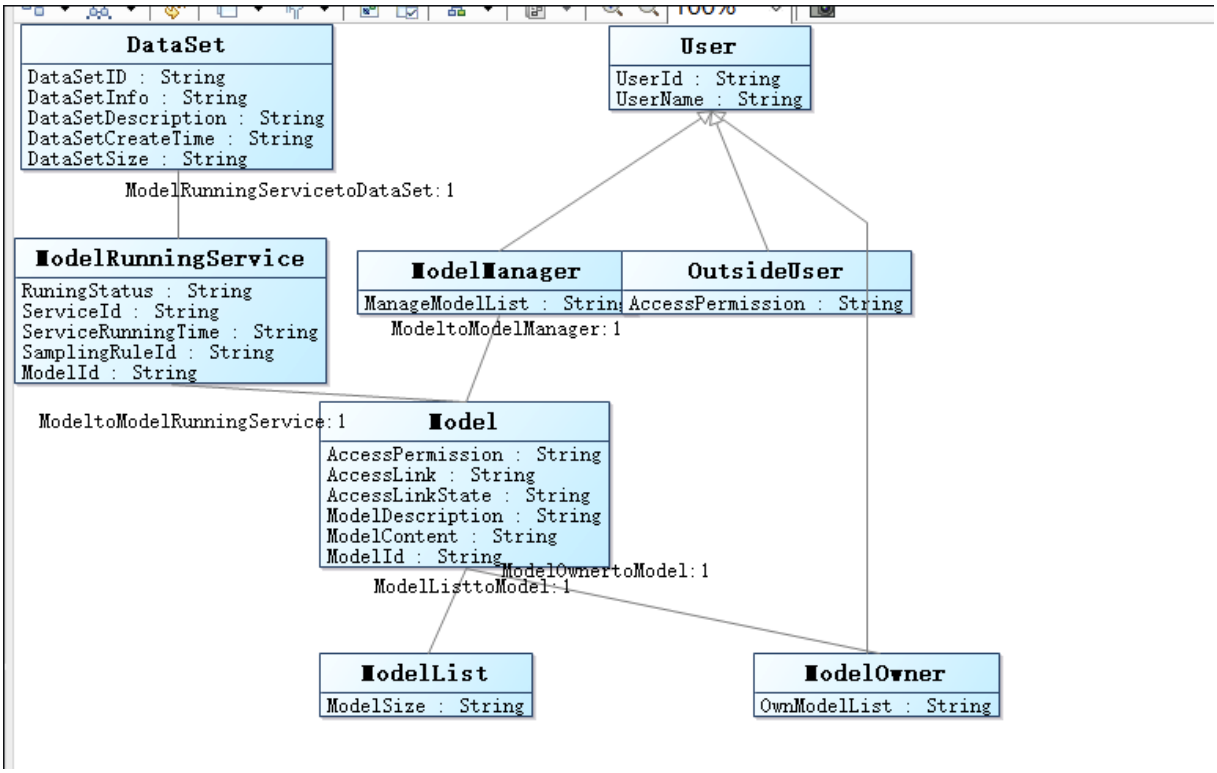
图 3.6 更新模型

1) 模型拥有者想要对已有的模型进行一定的更新，首先会输入模型的 id 信息，来定位模型；系统会根据对应信息返回对应的模型。

2) 模型拥有者会根据需求，输入需要更新的信息，提交到系统，系统会更新模型的信息，若成功的更新，则会返回需求，需要模型拥有者提供更新的描述。

3) 模型拥有者提交相应的描述语句，完成更新流程。

3.3 概念类图



概念类图主体有七个类所构成， 主要是两大部分，用户相关的类与模型/服务相关的类。用户相关的类主要分为用户的主要父类， 已经用户的具体角色实现的子类，子类主要包括模型管理者类、外部用户类、模型拥有者类。子类之间的差异主要体现在对模型的访问权限、模型的分享权限上：其中模型管理着可以对若干模型进行管理，因此拥有一个成员变量（所管理模型列表）。模型相关的类有模型的主体类与模型列表容器类，模型运行后所对应的服务的类， 模型所对应的数据集的类。模型主体的类包括一些关于模型内容相关的描述以及一些模型访问权限相关的描述、模型外链的访问链接等。模型运行后会对应相应的运行服务，运行服务有其相应的状态以及服务所对应的规则和模型的 ID，其次模型运行的服务会涉及到数据集相关的问题，数据集类主要包含了数据集的描述和创建时间等。如何调用线上模型主要涉及到的类是模型主体类， 模型运行服务类，模型持有者类，模型拥有者向系统提出请求调用相关的数据集，系统同意后会返回一个模型调用成功后的服务，模型调用者可以监控模型调用服务的状态。

3.4 OCL 合约

1. CallOnlineModelAPIService::callOnlineModelAsync

2. CheckRunningStatusService::checkModelRunningStatus
3. CallOnlineModelAPIService::getCallResult
4. FindDatasetService::selectModel
5. FindDatasetService::selectRule
6. FindDatasetService::getDataset
7. UploadModelService::inputModelID
8. UploadModelService::updateModelContent
9. UploadModelService::updateModelDescription
10. UpdateModelAccessPermissionService::inputNewAccessPermission
11. UpdateModelAccessPermissionService::inputModelId
12. UpdateModelAccessPermissionService::confirmUpdate
13. ModifyRunningStatusService::selectModel
14. ModifyRunningStatusService::inputNewRunningStatus
15. ModifyRunningStatusService::confirmModify
16. CreateModelSharingLinkService::inputAccessLink
17. CreateModelSharingLinkService::confirmCreate
18. UpdateModelSharingLinkAccessPermissionService::inputTargetModelID
19. UpdateModelSharingLinkAccessPermissionService::updateLinkState
20. UpdateModelSharingLinkAccessPermissionService::confirmUpdateLinkPermission

下面重点介绍一下“异步调用线上模型 API”这一用例的合约。

合约代码如下：

```
Contract CallOnlineModelAPIService::callOnlineModelAsync(modelId: String, args: String): String {

    definition:
        model:Model = Model.allInstance()->any(m:Model | m.ModelId = modelId)

    precondition:
        true
    postcondition:
        if
            (model.oclIsUndefined() = false)
        then
           self.ModelService = CheckRunningStatusService.checkModelRunningStatus and
            self.Args = args and
            result = self.generateCallId
        else
            result = ""
        endif
}
```

因为该合约描述的 API 调用操作是是异步的，即，调用者不希望在调用 API 的过程中被阻塞，

API 被调用的具体执行工作应该被异步执行。

调用者在调用模型的服务时，必须给出调用的模型服务的 ID 和调用所使用的参数。该函数会首先检查对应 ID 的模型服务是否在线，并且状态是否健康。如果满足条件，则将调用请求传递给下游服务，并将“调用成功”的信息返回给调用者。在返回的同时，会设置一个“callID”，以便调用者后续凭借该“callID”来获取最终的模型服务的调用结果。

一、xx 服务 CallOnlineModelAPIService

二、xx 服务 CheckRunningStatusService

三、xx 服务 FindDatasetService

四、上传模型服务 UploadModelService

有三个合约 inputModelID、updateModelContent、updateModelDescription，分别用来确定 Model、更新 Content、更新 Description。下面为服务三个合约的详细描述：

第一步调用 inputModelID 函数，传递 modelId 参数，返回布尔值。首先根据 modelId 找到相应的 model。如果 model 未定义，则 ModelIDValidated 为 false，并且返回 false；ModelIDValidated 为 true，保留 Model，并且返回 true。

第二步调用 updateModelContent 函数，传递 content 参数，返回布尔值。首先判断 ModelIDValidated 是否为真，model 是否定义。只有满足条件才能更新 Model 的 Content 属性，并返回 true。

第三步调用 updateModelDescription 函数，传递 descripition 参数，返回布尔值。首先判断 ModelIDValidated 是否为真，model 是否定义。只有满足条件才能更新 Model 的 Description 属性，并返回 true。

五、更新模型访问权限服务 UpdateModelAccessPermissionService

有三个合约 iinputModelId、inputNewAccessPermission、confirmUpdate，分别用来确定

Model、获取新的访问权限、确认更新模型的访问权限。下面为服务三个合约的详细描述：

第一步调用 iinputModelId 函数，传递 modelId 参数，返回布尔值。首先根据 modelId 找到相应的 model。如果 model 未定义，则 ModelIDValidated 为 false，并且返回 false；ModelIDValidated 为 true，保留 Model，并且返回 true。

第二步调用 inputNewAccessPermission 函数，传递 accessPermissionType 参数，返回布尔值。首先判断 ModelIDValidated 是否为真，model 是否定义。只有满足条件才将 accessPermissionType 赋值给 ModelAccessPermission，并返回 true。

第三步调用 confirmUpdate 函数，返回布尔值。首先判断 ModelIDValidated 是否为真，model 是否定义。只有满足条件才将 Model 的 AccessPermission 属性更新为 ModelAccessPermission 值，并返回 true。

六、修改模型运行状态服务 ModifyRunningStatusService

有三个合约 selectModel、inputNewRunningStatus、confirmModify，分别用来确定 Model、获取新的模型运行状态、确认修改模型的运行状态。下面为服务三个合约的详细描述：

第一步调用 selectModel 函数，传递 modelId 参数，返回布尔值。首先根据 modelId 找到相应的 model。如果 model 未定义，则 ModelIDValidated 为 false，并且返回 false；ModelIDValidated 为 true，保留 Model，并且返回 true。

第二步调用 inputNewAccessPermission 函数，传递 newStatus 参数，返回布尔值。首先判断 ModelIDValidated 是否为真，model 是否定义。只有满足条件才将 newStatus 赋值给 ModelRuningStatus，并返回 true。

第三步调用 confirmModify 函数，返回布尔值。首先判断 ModelIDValidated 是否为真，model 是否定义。只有满足条件才将 Model 的 RuningStatus 属性更新为 ModelRuningStatus 值，并返回 true。

七、创建模型分享链接服务 CreateModelSharingLinkService

有两个合约 inputAccessLink、confirmModify，分别用来确定 Model 和确认创建模型访问

链接。下面为服务两个合约的详细描述：

第一步调用 inputAccessLink 函数，传递 modelId 参数，返回布尔值。首先根据 modelId 找到相应的 model。如果 model 未定义，则 ModelIDValidated 为 false，并且返回 false；否则 ModelIDValidated 为 true，保留 Model，程序为 ModelLink 自动生成值，ModelLinkState 默认为“closed”并且返回 true。

第二步调用 confirmModify 函数，首先判断 ModelIDValidated 是否为真，model 是否定义。只有满足条件才将 Model 的 AccessLink 属性更新为 ModelLink 值，将 Model 的 AccessLinkState 属性更新为 ModelLinkState 值，并返回 true。

八、更新模型分享链接权限服务 UpdateModelSharingLinkAccessPermissionService

有三个合约 inputTargetModelID、updateLinkState、confirmUpdateLinkPermission，分别用来确定 Model、获取新的模型分享链接状态、确认修改模型的访问权限。下面为服务三个合约的详细描述：

第一步调用 inputTargetModelID 函数，传递 modelId 参数，返回布尔值。首先根据 modelId 找到相应的 model。如果 model 未定义，则 ModelIDValidated 为 false，并且返回 false；ModelIDValidated 为 true，保留 Model，并且返回 true。

第二步调用 updateLinkState 函数，传递 state 参数，返回布尔值。首先判断 ModelIDValidated 是否为真，model 是否定义。state 是个长字符串，值中包含模型分享链接权限的信息，只有满足条件才将 state 赋值给 ModelLinkState，并返回 true。

第三步调用 confirmUpdateLinkPermission 函数，返回布尔值。首先判断 ModelIDValidated 是否为真，model 是否定义。只有满足条件才将 Model 的 ModelLinkState 属性更新为 AccessLinkState 值，并返回 true。

4 目标系统 UML 最终设计模型

姓名	学号	工作内容	工作占比

陈泽人	SY2121101		
刘炜	SY2121110		
任婷伊	SY2121113		
王子勤	SY2121117		
李楠	SY2121108		

4.1 架构图

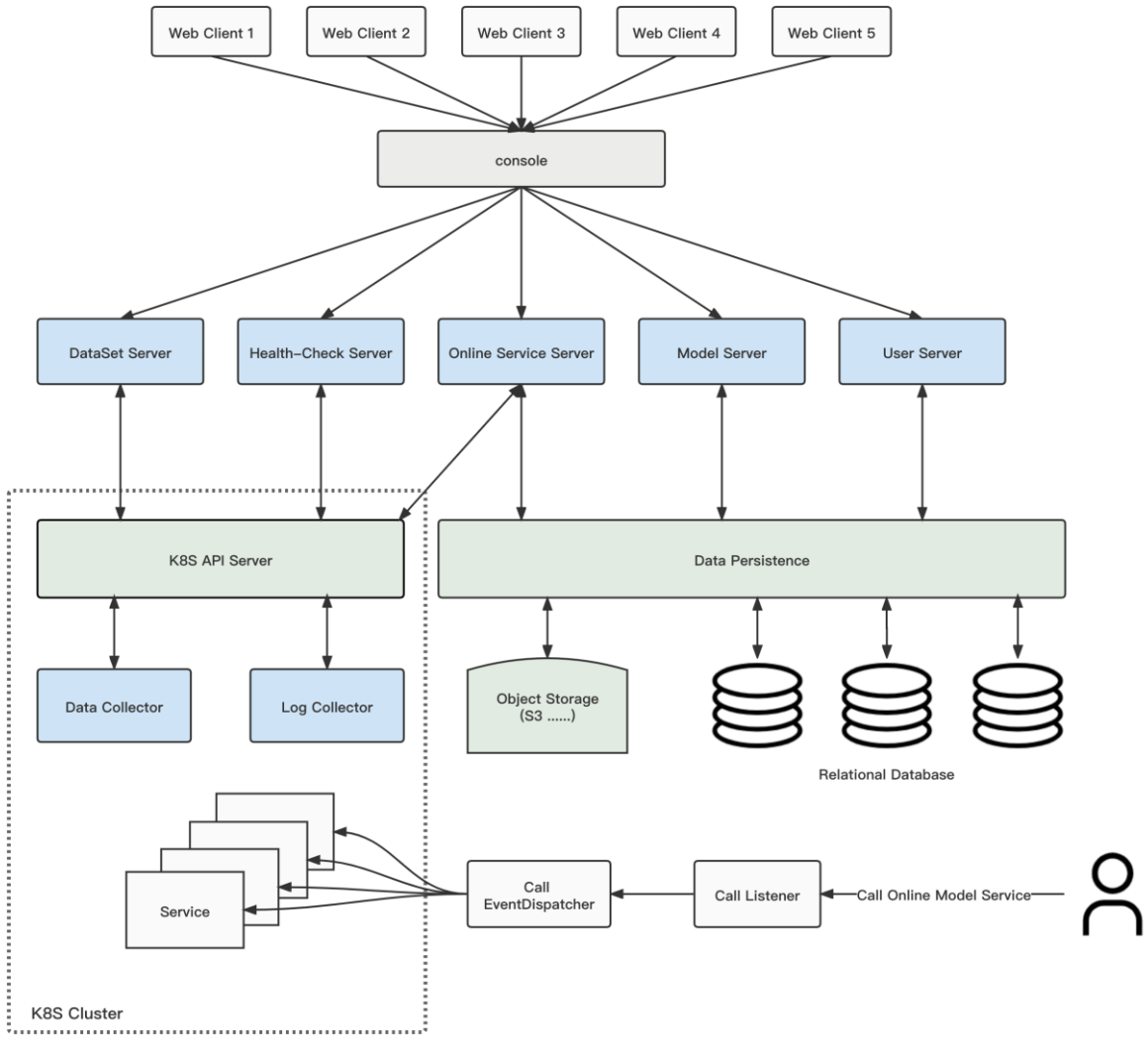


图 4.1 DLMMS 架构设计图

在 DLMMS 的设计中，我们使用了两种架构设计模式： 分层架构和事件驱动架构。

分层架构：整个系统分为用户 UI 层、网关层、服务层和支持层（其中，支持层包括一个数据持久化服务和一个 Kubernetes 集群）。用户在 UI 层发起请求后，请求会经由网关根据请求的类型通过负载均衡算法发送给对应的负责处理具体业务逻辑的 Server。这些微服务 Serve 会调用的数据持久层进行数据的增删改查，具体地，对于普通的数据信息，使用关系型数据库进行管理；对于“模型文件”这种大型的二进制文件，则直接使用对象存储服务。同时，模型的线上服务将会被部署在 Kubernetes 集群中，每个模型的“线上服务”都是一个 Kubernetes 中的“Service”实例。负责对“模型线上服务”进行管理的微服务 Server 将通过调用 Kubernetes 的 API 实现相应功能。

事件驱动架构：当一个外部用户调用“模型线上服务”时，由事件驱动的 Call listener 会响应用户的请求，并由 EventDisptcher 根据用户请求中携带的不同信息，派发给已上线的服务进行处理，将结果返回给用户。

4.2 类图

我们设计了 DLMMS 的类图，如图 4.2 所示。

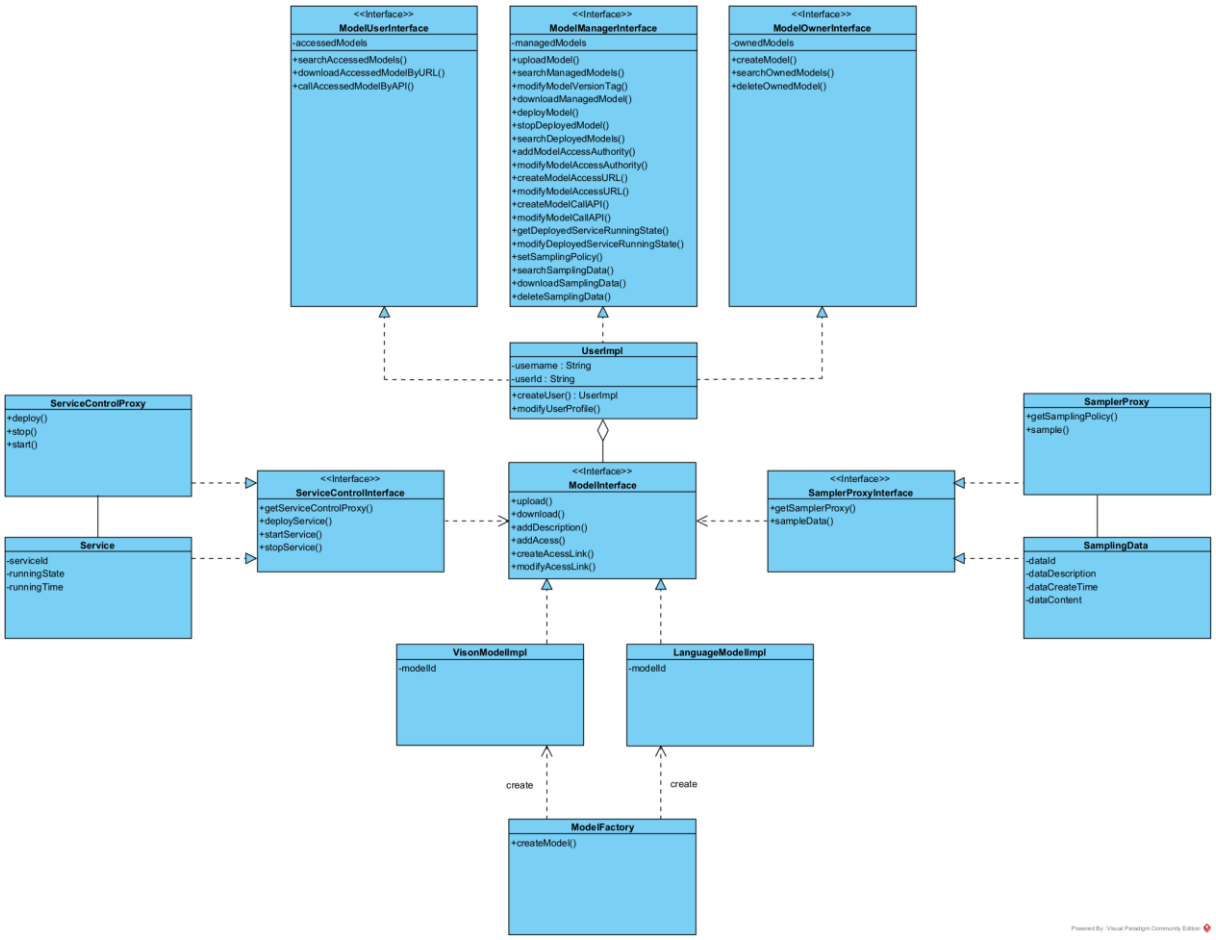


图 4.2 DLMMS 类图

该系统主要由用户类 UserImpl 和深度学习模型类 VisionModelImpl、LanguageModelImpl 组成。

平台用户通常拥有三个身份：模型创建者、模型管理者和模型使用者，为了将不同用户的权限和操作隔离，我们针对上述三个身份设计了三个接口：

- ModelUserInterface：模型的使用者，当用户被模型的创建者或管理者授予访问权限后，可以访问对应模型并调用该模型提供的接口；
- ModelManagerInterface：模型的管理者，当用户被模型创建者授予管理权限后，将成为模型的管理者，可以管理模型部署后的运行状态、模型的访问权限等，同时也拥有使用者的所有权限；
- ModelOwnerInterface：模型的创建者，拥有最高的管理和操作权限，同时也拥有管理者和使用者的所有权限。

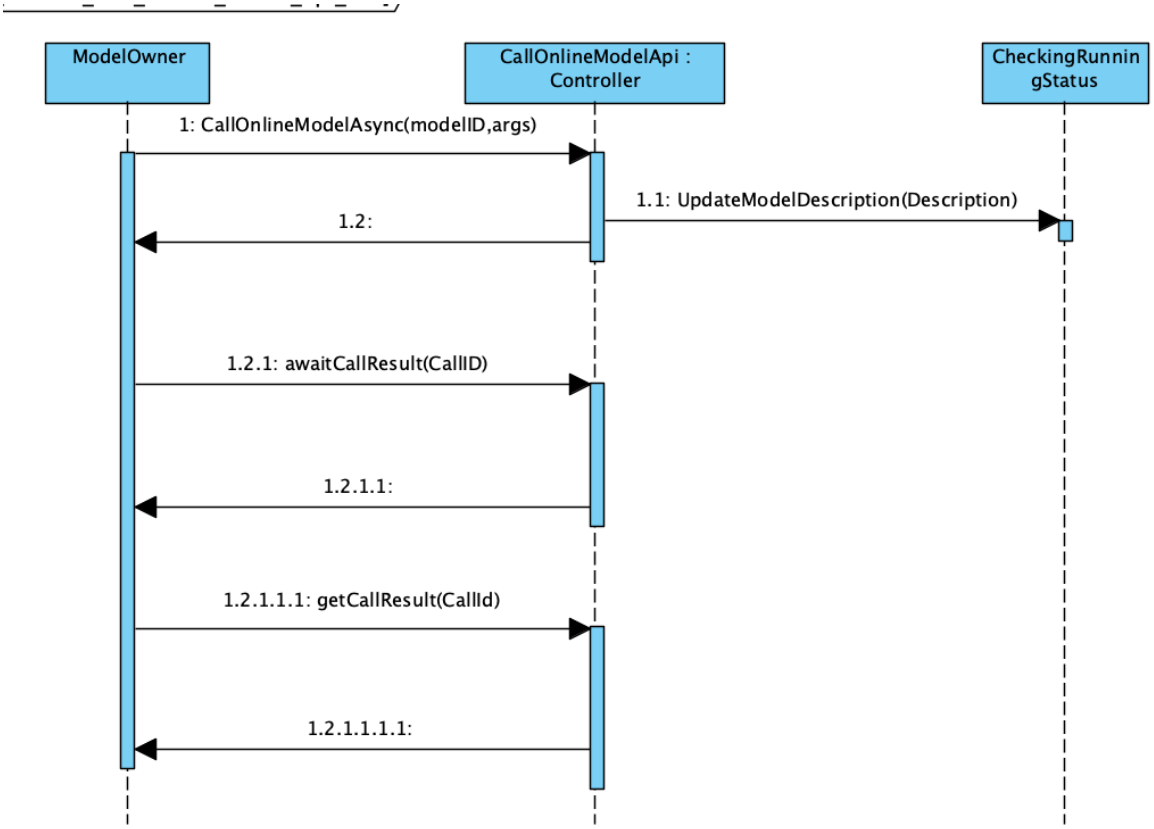
对于深度学习模型，我们考虑了两种比较流行的深度学习任务：视觉任务和语言任务，分别对

应 VisionModelImpl 和 LanguageModelImpl。通过工厂模式，我们设计了 ModelFactory 类，可以根据用户的不同需求去创建不同任务的深度学习模型类。

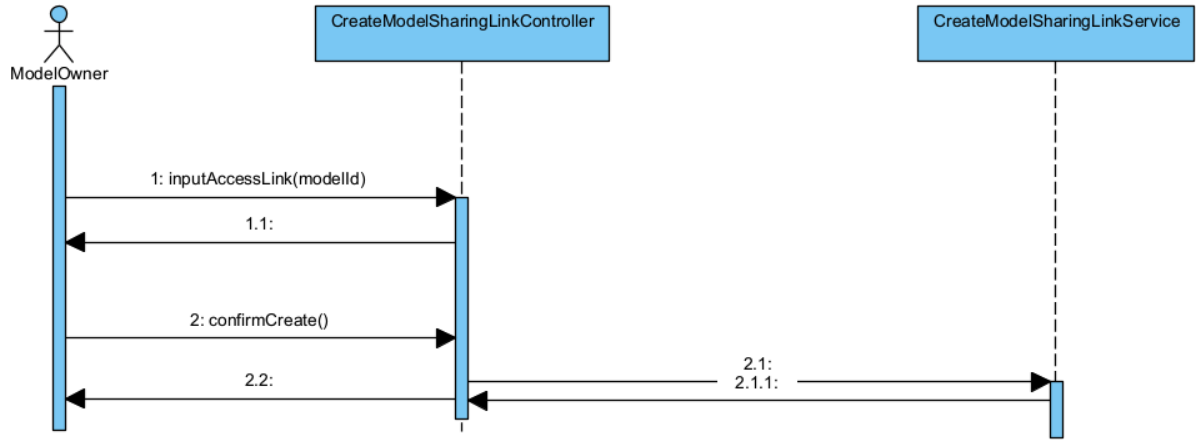
当模型部署后会生成一个对应的服务类 Service，模型使用者可以通过调用该服务对外提供的接口来调用模型。对于管理端，我们设计了代理类 ServiceControlProxy 对 Service 类进行基本的管理，包括模型部署、服务启动和服务中止等。代理模型的引入会后续添加更多的控制操作留出了拓展空间。

此外，DLMMS 还会对运行的服务采集数据信息，我们设计了 SamplingData 类表示采集的数据对象。为了应对不同的采集策略，我们设计了代理类 SamplerProxy 来处理不同的采集策略。

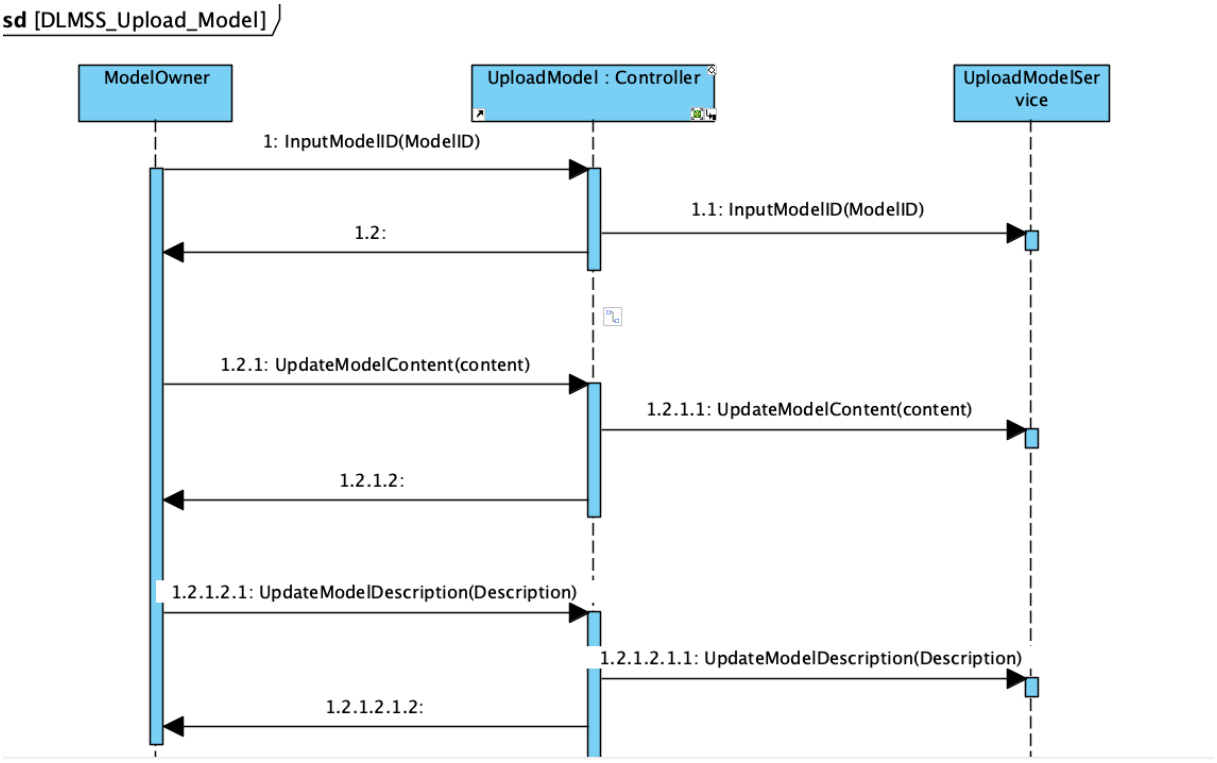
4.3 顺序图



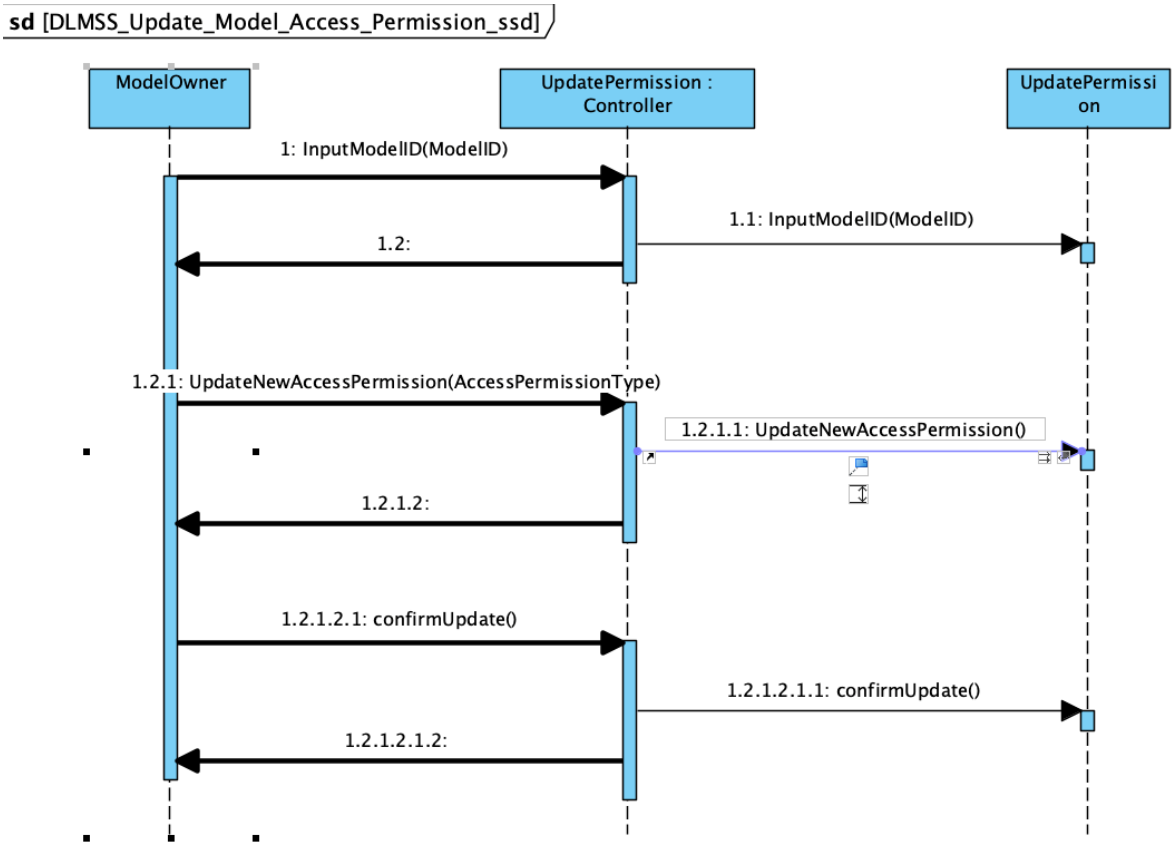
（模型拥有者）首先向系统提出模型 api 的调用申请，如果模型可以被正常调用，那么系统会判断模型的运行状况，若模型正常运行，则会返回模型的服务调用结果以及相应的模型运行服务状态。



（模型拥有者）首先向系统提出分享模型调用接口的 API，如果模型的访问权限足够，那么改用户可以讲模型调用 API 接口分享，系统会生成相应的访问 URL，返回给模型拥有者。



（模型拥有者）首先向系统提出模型上传的请求，同时输入上传模型的 ID 以及模型的具体内容，当系统的模型 ID 和模型内容校验成功后， 模型呗上传至数据库中进行保存，并将上传成功的消息反馈至用户



（模型拥有者）向系统提出更新模型的访问权限的请求，系统首先会核验该请求提出者是否具有对目标模型的修改权限，如果权限一切校验成功后，那么系统会修改指定模型的访问权限，然后反馈修改结果。