1 Introduction

1.1 Purpose

This subsection should

- a) Delineate the purpose of the SRS;
- b) Specify the intended audience for the SRS.

1.2 Scope

Name of software to be developed: DLModelMS1 System

This subsection should

- b) Explain what the software product(s) will, and, if necessary, will not do;
- c) Describe the application of the software being specifified, including relevant benefifits, objectives, and goals;
- d) Be consistent with similar statements in higher-level specififications (e.g., the system requirements specifification), if they exist.

1.3 Product Overview

1.3.1 Product perspective

This subsection of the SRS should put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.

This subsection should also describe how the software operates inside various constraints. For example,
these constraints could include

- a) System interfaces;
- b) User interfaces;
- c) Hardware interfaces;
- d) Software interfaces;
- e) Communications interfaces;
- f) Memory;
- j) Operations;
- k) Site adaptation requirements.

1.3.1.1 System interfaces

SI1 - DLModelMS1System

Service Name:	DLModelMS1System
Service ID:	SI1
Description:	
Operation:	• viewModel • callModel
Temporary Variable	**Variable Description**
CurrentModel	CurrentModel is a object of Model
CurrentRunningModel	CurrentRunningModel is a object of RunningModel

SI2 - ManageModelService

Service Name:	ManageModelService
Service ID:	SI2
Description:	
Operation:	• updateModelInformation • updateDataSet

SI3 - ManageRunningModelService

Service Name:	ManageRunningModelService
Service ID:	SI3
Description:	
Operation:	• modifyRunningModel • modifyKey

SI4 - UploadModelService

Service Name:	UploadModelService
Service ID:	SI4
Description:	
Operation:	• uploadModelInformation • uploadModelAttachment • uploadDataSet

SI5 - DeployModelService

Service Name:	DeployModelService
Service ID:	SI5
Description:	
Operation:	• viewModel • deployModel • createKey

SI6 - DownloadModelService

Service Name:	DownloadModelService
Service ID:	SI6
Description:	
Operation:	• searchModelByKeyword • viewModel • downModel

SI7 - AskForAPIService

Service Name:	AskForAPIService
Service ID:	SI7
Description:	
Operation:	• listRunningModel • askForAPI

SI8 - ThirdPartyServices

Service Name:	ThirdPartyServices
Service ID:	SI8
Description:	
Operation:	• generateURL • download • sendEmail

1.3.2 Product functions

Use Case Diagram


Use Case Diagram

ID	Use Case Name	Use Case Description	Subfunction
UC1	uploadModel		uploadModelInformation uploadModelAttachment uploadDataSet
UC2	deployModel		viewModel deployModel createKey
UC3	downloadModel		searchModelByKeyword viewModel downModel
UC4	callModel		
UC5	manageModel		updateModelInformation updateDataSet
UC6	manageRunningModel		modifyRunningModel modifyKey

1.3.3 User characteristics

ID	Actor	Description	Super Actor
A1	Owner		
A2	User		

1.3.4 Limitations

This subsection of the SRS should provide a general description of any other items that will limit the developer's options. These include

- a) Regulatory policies;
- b) Hardware limitations (e.g., signal timing requirements);
- c) Interfaces to other applications;
- d) Parallel operation;
- e) Audit functions;
- f) Control functions;
- g) Higher-order language requirements;
- h) Signal handshake protocols (e.g., XON-XOFF, ACK-NACK);
- i) Reliability requirements;
- j) Criticality of the application;

- k) Safety and security considerations.
- l) physical/mental considerations; and
- m) limitations that are sourced from other systems, including real-time requirements from the controlled system through interfaces.

1.4 Definitions

This subsection should provide the defifinitions of all terms required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

2 References

This subsection should

- a) Provide a complete list of all documents referenced elsewhere in the SRS;
- b) Identify each document by title, report number (if applicable), date, and publishing organization;
- c) Specify the sources from which the references can be obtained.

This information may be provided by reference to an appendix or to another document.

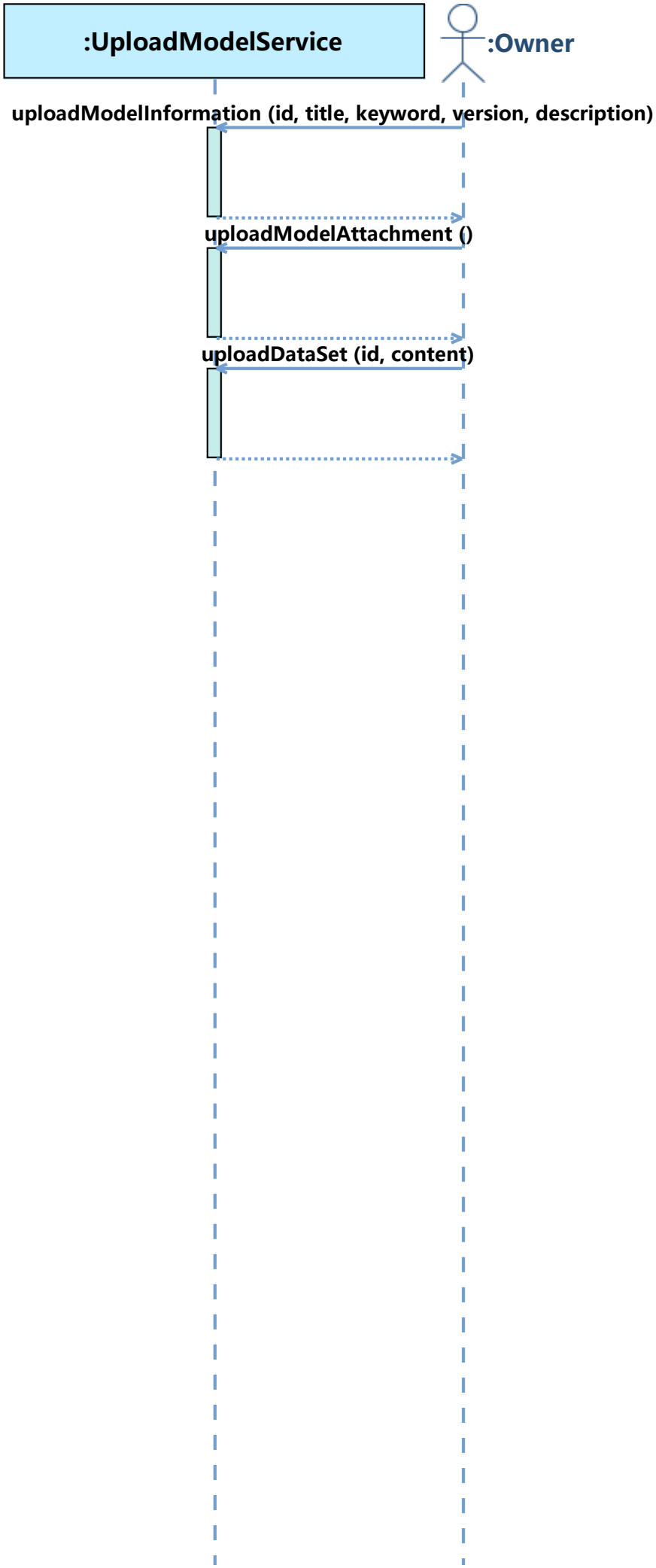
3 Requirements

3.1 Functions

3.1.1 Use Case

UC1 - uploadModel

System Sequence Diagram:

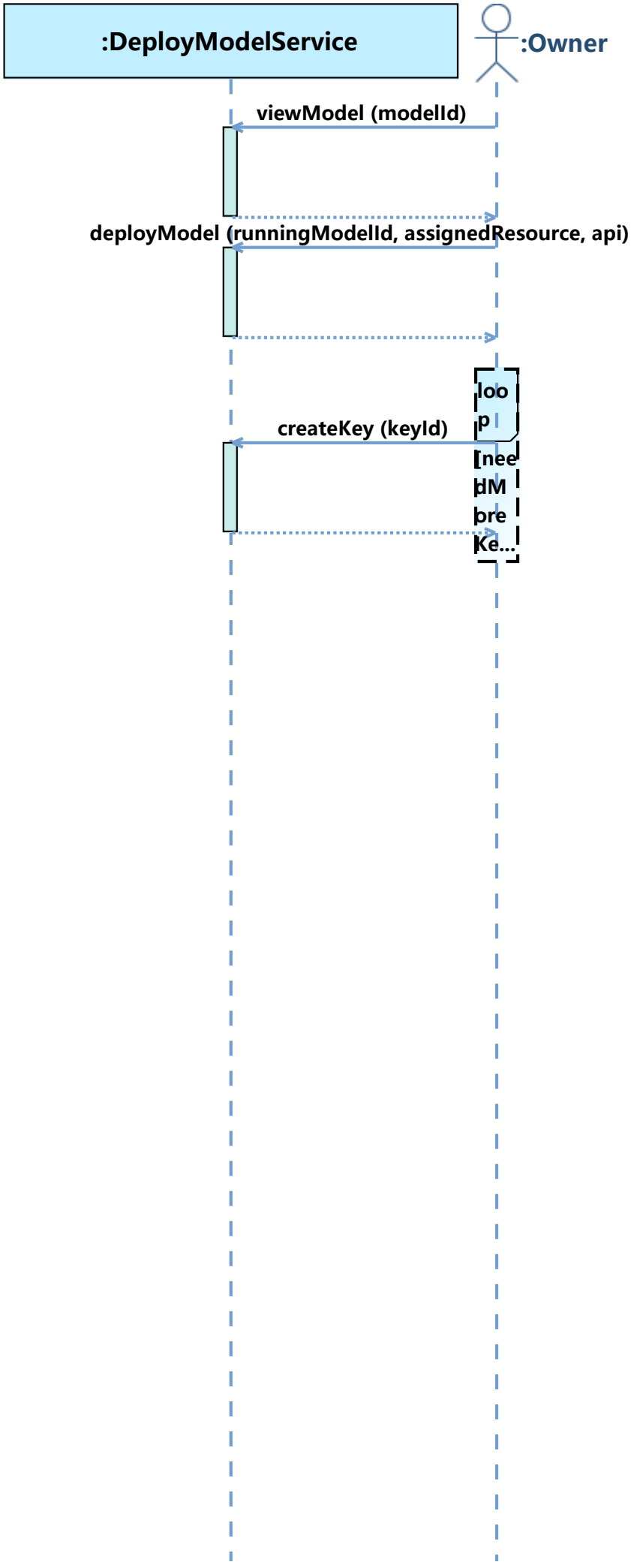

:UploadModelService
:Owner
uploadModelInformation (id, title, keyword, version, description)
uploadModelAttachment ()
uploadDataSet (id, content)

Use Case Description:

UseCase Name:	uploadModel
UseCase ID:	UC1
Brief Description:	
Involved Actor:	Owner
Preconditions:	
Postconditions:	
Basic Path:	1. Owner clicks to execute the operation uploadModelInformation, with entering id, title, keyword, version, description 2. Owner clicks to execute the operation uploadModelAttachment 3. Owner clicks to execute the operation uploadDataSet, with entering id, content
Alternative Path:	

UC2 - deployModel

System Sequence Diagram:

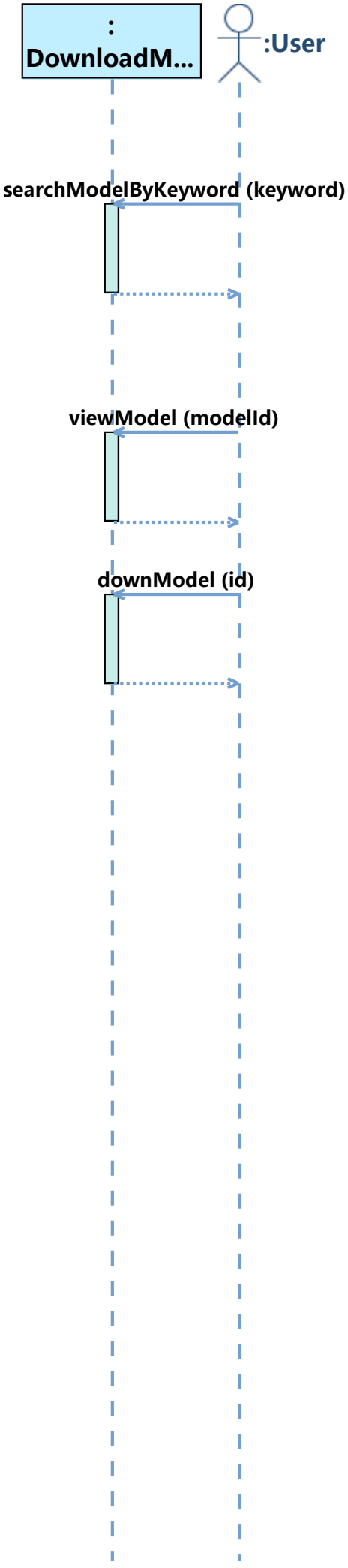

:DeployModelService
:Owner
viewModel (modelId)
deployModel (runningModelId, assignedResource, api)
loop
createKey (keyId)
[needMoreKe...]

Use Case Description:

UseCase Name:	deployModel
UseCase ID:	UC2
Brief Description:	
Involved Actor:	Owner
Preconditions:	
Postconditions:	
Basic Path:	1. Owner clicks to execute the operation viewModel, with entering modelId 2. Owner clicks to execute the operation deployModel, with entering runningModelId, assignedResource, api 3. Owner clicks to execute the operation createKey, with entering keyId *If needMoreKeys, repeat the step(s) 3*
Alternative Path:	

UC3 - downloadModel

System Sequence Diagram:


:
DownloadM...
:User
searchModelByKeyword (keyword)
viewModel (modelId)
downModel (id)

Use Case Description:

UseCase Name:	downloadModel
UseCase ID:	UC3
Brief Description:	
Involved Actor:	User
Preconditions:	
Postconditions:	
Basic Path:	1. User clicks to execute the operation searchModelByKeyword, with entering keyword 2. User clicks to execute the operation viewModel, with entering modelId 3. User clicks to execute the operation downModel, with entering id
Alternative Path:	

UC4 - callModel

Use Case Description:

UseCase Name:	callModel
UseCase ID:	UC4
Brief Description:	
Involved Actor:	User
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC5 - manageModel

Use Case Description:

UseCase Name:	manageModel
UseCase ID:	UC5
Brief Description:	
Involved Actor:	Owner
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

UC6 - manageRunningModel

Use Case Description:

UseCase Name:	manageRunningModel
UseCase ID:	UC6
Brief Description:	
Involved Actor:	Owner
Preconditions:	
Postconditions:	
Basic Path:	
Alternative Path:	

3.1.2 System Operation

OP1 - uploadModelInformation

Operation Name:	uploadModelInformation
Operation ID:	OP1
Description:	
Service:	UploadModelService
Input:	1. name: *id*, type: String 2. name: *title*, type: String 3. name: *keyword*, type: String 4. name: *version*, type: String 5. name: *description*, type: String
Output Type:	Boolean
Definition:	*a* is the object *b* in the instance set of class Model. *b* represents an object of class Model, and *b* meets: The attribute *Id* of the object *b* is equal to *id*
Preconditions:	The object *a* doesn't exist
Postconditions:	1. *e* represented the object of class Model 2. The object *e* was created 3. The attribute *Id* of the object *e* became *id* 4. The attribute *Title* of the object *e* became *title* 5. The attribute *Keyword* of the object *e* became *keyword* 6. The attribute *Version* of the object *e* became *version* 7. The attribute *Description* of the object *e* became *description* 8. The attribute *LastUpdateTime* of the object *e* was equal to *Now* 9. The object *e* was put into the instance set of class Model 10. The object CurrentModel became *e* 11. The return value was **true**

Contract of **uploadModelInformation**:

```
Contract UploadModelService::uploadModelInformation(id : String, title : String,
keyword:String, version : String, description : String) : Boolean {
        definition:
            a:Model = Model.allInstance()->any(b:Model| b.Id = id)
        precondition:
            a.oclIsUndefined() = true
```

```
        postcondition:
            let e:Model in
            e.oclIsNew() and
            e.Id = id and
            e.Title = title and
            e.Keyword = keyword and
            e.Version = version and
            e.Description = description and
            e.LastUpdateTime.isEqual(Now) and
            Model.allInstance()->includes(e) and
            self.CurrentModel = e and
            result = true
}
```

OP2 - uploadModelAttachment

Operation Name:	uploadModelAttachment
Operation ID:	OP2
Description:	
Service:	UploadModelService
Input:	None
Output Type:	Boolean
Preconditions:	1. The object *CurrentModel* exists 2. The object *CurrentModel* doesn't exist
Postconditions:	1. *c* represented the object of class Attachment 2. The object *c* was created 3. The attribute *URL* of the object *c* became the return value of system operation generateURL 4. The object *CurrentModel* was linked to the object *c* by *ModeltoAttachment* 5. The object *c* was put into the instance set of class Attachment 6. The return value was **true**

Contract of **uploadModelAttachment**:

```
Contract UploadModelService::uploadModelAttachment() : Boolean {
        precondition:
            CurrentModel.oclIsUndefined() = false and
            CurrentModel.ModeltoAttachment.oclIsUndefined() = true
        postcondition:
            let c:Attachment in
            c.oclIsNew() and
            c.URL = generateURL() and
            CurrentModel.ModeltoAttachment = c and
            Attachment.allInstance()->includes(c) and
            result = true
}
```

OP3 - uploadDataSet

Operation Name:	uploadDataSet
Operation ID:	OP3
Description:	
Service:	UploadModelService
Input:	1. name: *id*, type: String 2. name: *content*, type: String
Output Type:	Boolean
Definition:	*a* is the object *b* in the instance set of class DataSet. *b* represents an object of class DataSet, and *b* meets: The attribute *Id* of the object *b* is equal to *id*
Preconditions:	1. The object *a* doesn't exist 2. The object *CurrentModel* exists
Postconditions:	1. *c* represented the object of class DataSet 2. The object *c* was created 3. The attribute *Id* of the object *c* became *id* 4. The attribute *Content* of the object *c* became *content* 5. The attribute *LastUpdateTime* of the object *c* was equal to *Now* 6. The object *c* was put into the instance set of class DataSet 7. The object *CurrentModel* was linked to the object *c* by *ContainedDataSet* 8. The return value was **true**

Contract of **uploadDataSet**:

```
Contract UploadModelService::uploadDataSet(id : String, content : String) :
Boolean {
        definition:
            a:DataSet = DataSet.allInstance()->any(b:DataSet| b.Id = id)
        precondition:
            a.oclIsUndefined() = true and
            CurrentModel.oclIsUndefined() = false
        postcondition:
            let c:DataSet in
            c.oclIsNew() and
            c.Id = id and
            c.Content = content and
            c.LastUpdateTime.isEqual(Now) and
            DataSet.allInstance()->includes(c) and
            CurrentModel.ContainedDataSet->includes(c) and
            result = true
}
```

OP4 - deployModel

Operation Name:	deployModel
Operation ID:	OP4
Description:	
Service:	DeployModelService
Input:	1. name: *runningModelId*, type: String 2. name: *assignedResource*, type: String 3. name: *api*, type: String
Output Type:	Boolean
Definition:	*a* is the object *e* in the instance set of class RunningModel. *e* represents an object of class RunningModel, and *e* meets: The attribute *Id* of the object *e* is equal to *runningModelId*
Preconditions:	1. The object *CurrentModel* exists 2. The object *a* doesn't exist
Postconditions:	1. *b* represented the object of class RunningModel 2. The object *b* was created 3. The attribute *Id* of the object *b* became *runningModelId* 4. The attribute *AssignedResource* of the object *b* became *assignedResource* 5. The attribute *API* of the object *b* became *api* 6. The object *b* was put into the instance set of class RunningModel 7. The object *b* was linked to the object *CurrentModel* by *BelongedModel* 8. The object *CurrentModel* was linked to the object *b* by *ContainedRunningModel* 9. The object CurrentRunningModel became *b* 10. The return value was **true**

Contract of **deployModel**:

```
Contract DeployModelService::deployModel(runningModelId : String,
assignedResource : String, api : String) : Boolean {
        definition:
            a:RunningModel = RunningModel.allInstance()->any(e:RunningModel|
e.Id = runningModelId)
        precondition:
            CurrentModel.oclIsUndefined() = false and
```

```
            a.oclIsUndefined() = true
        postcondition:
            let b:RunningModel in
            b.oclIsNew() and
            b.Id = runningModelId and
            b.AssignedResource = assignedResource and
            b.API = api and
            RunningModel.allInstance()->includes(b) and
            b.BelongedModel = CurrentModel and
            CurrentModel.ContainedRunningModel->includes(b) and
            self.CurrentRunningModel = b and
            result = true
}
```

OP5 - createKey

Operation Name:	createKey
Operation ID:	OP5
Description:	
Service:	DeployModelService
Input:	name: *keyId*, type: String
Output Type:	Boolean
Definition:	*key* is the object *e* in the instance set of class Key. *e* represents an object of class Key, and *e* meets: The attribute *Id* of the object *e* is equal to *keyId*
Preconditions:	The object *key* doesn't exist
Postconditions:	1. *b* represented the object of class Key 2. The object *b* was created 3. The attribute *Id* of the object *b* became *keyId* 4. The attribute *IsValid* of the object *b* became **true** 5. The object *b* was put into the instance set of class Key 6. The object *b* was linked to the object *CurrentRunningModel* by *BelongedRunningModel* 7. The object *CurrentRunningModel* was linked to the object *b* by *ContainedKey* 8. The return value was **true**

Contract of **createKey**:

```
Contract DeployModelService::createKey(keyId : String) : Boolean {
        definition:
            key:Key = Key.allInstance()->any(e:Key| e.Id = keyId)
        precondition:
            key.oclIsUndefined() = true
        postcondition:
            let b:Key in
            b.oclIsNew() and
            b.Id = keyId and
            b.IsValid = true and
            Key.allInstance()->includes(b) and
            b.BelongedRunningModel = CurrentRunningModel and
            CurrentRunningModel.ContainedKey->includes(b) and
            result = true
}
```

OP6 - callModel

Operation Name:	callModel
Operation ID:	OP6
Description:	
Service:	DLModelMS1System
Input:	name: *keyId*, type: String
Output Type:	String
Definition:	*key* is the object *e* in the instance set of class Key. *e* represents an object of class Key, and *e* meets: The attribute *Id* of the object *e* is equal to *keyId*
Preconditions:	1. The object *key* exists 2. The attribute *IsValid* of the object *key* is equal to **true**
Postconditions:	The return value was the attribute *API* of the object *key*

Contract of **callModel**:

```
Contract DLModelMS1System::callModel(keyId : String) : String {
        definition:
            key:Key = Key.allInstance()->any(e:Key| e.Id = keyId)
//          ,a:RunningModel = key.BelongedRunningModel
        precondition:
            key.oclIsUndefined() = false and
//          a.oclIsUndefined() = false and
            key.IsValid = true
        postcondition:
            result = key.BelongedRunningModel.API
}
```

OP7 - searchModelByKeyword

Operation Name:	searchModelByKeyword
Operation ID:	OP7
Description:	
Service:	DownloadModelService
Input:	name: *keyword*, type: String
Output Type:	Set of Model
Preconditions:	The *keyword* is not equal to **null**
Postconditions:	The return value was the set of class Model, including all *m* in the instance set of class Model. *m* represented an object of class Model, and *m* meet: The attribute *Keyword* of the object *m* was equal to *keyword*

Contract of **searchModelByKeyword**:

```
Contract DownloadModelService::searchModelByKeyword(keyword : String) :
Set(Model) {
        precondition:
            keyword <> ""
        postcondition:
            result = Model.allInstance()->select(m:Model| m.Keyword = keyword)
}
```

OP8 - viewModel

Operation Name:	viewModel
Operation ID:	OP8
Description:	
Service:	DLModelMS1System
Input:	name: *modelId*, type: String
Output Type:	Model
Definition:	*a* is the object *b* in the instance set of class Model. *b* represents an object of class Model, and *b* meets: The attribute *Id* of the object *b* is equal to *modelId*
Preconditions:	The object *a* exists
Postconditions:	1. The object CurrentModel became *a* 2. The return value was *a*

Contract of **viewModel**:

```
Contract DLModelMS1System::viewModel(modelId : String) : Model {
        definition:
            a:Model = Model.allInstance()->any(b:Model| b.Id = modelId)
        precondition:
            a.oclIsUndefined() = false
        postcondition:
            self.CurrentModel = a and
            result = a
}
```

OP9 - downModel

Operation Name:	downModel
Operation ID:	OP9
Description:	
Service:	DownloadModelService
Input:	name: *id*, type: String
Output Type:	Boolean
Definition:	*a* is the object *b* in the instance set of class Model. *b* represents an object of class Model, and *b* meets: The attribute *Id* of the object *b* is equal to *id*
Preconditions:	The object *a* exists
Postconditions:	1. The system operation download was executed 2. The return value was **true**

Contract of **downModel**:

```
Contract DownloadModelService::downModel(id : String): Boolean {
        definition:
            a:Model = Model.allInstance()->any(b:Model| b.Id = id)
        precondition:
            a.oclIsUndefined() = false
        postcondition:
            download(a.ModeltoAttachment.URL) and
            result = true
}
```

OP10 - updateModelInformation

Operation Name:	updateModelInformation
Operation ID:	OP10
Description:	
Service:	ManageModelService
Input:	1. name: *id*, type: String 2. name: *title*, type: String 3. name: *keyword*, type: String 4. name: *version*, type: String 5. name: *description*, type: String
Output Type:	Boolean
Definition:	*e* is the object *b* in the instance set of class Model. *b* represents an object of class Model, and *b* meets: The attribute *Id* of the object *b* is equal to *id*
Preconditions:	The object *e* exists
Postconditions:	1. The attribute *Id* of the object *e* became *id* 2. The attribute *Title* of the object *e* became *title* 3. The attribute *Keyword* of the object *e* became *keyword* 4. The attribute *Version* of the object *e* became *version* 5. The attribute *Description* of the object *e* became *description* 6. The attribute *LastUpdateTime* of the object *e* was equal to *Now* 7. The return value was **true**

Contract of **updateModelInformation**:

```
Contract ManageModelService::updateModelInformation(id : String, title : String,
keyword : String, version : String, description : String) : Boolean {
        definition:
            e:Model = Model.allInstance()->any(b:Model| b.Id = id)
        precondition:
            e.oclIsUndefined() = false
        postcondition:
            e.Id = id and
            e.Title = title and
            e.Keyword = keyword and
            e.Version = version and
            e.Description = description and
            e.LastUpdateTime.isEqual(Now) and
```

```
            result = true
}
```

OP11 - updateDataSet

Operation Name:	updateDataSet
Operation ID:	OP11
Description:	
Service:	ManageModelService
Input:	1. name: *id*, type: String 2. name: *content*, type: String
Output Type:	Boolean
Definition:	*c* is the object *b* in the instance set of class DataSet. *b* represents an object of class DataSet, and *b* meets: The attribute *Id* of the object *b* is equal to *id*
Preconditions:	The object *c* exists
Postconditions:	1. The attribute *Id* of the object *c* became *id* 2. The attribute *Content* of the object *c* became *content* 3. The attribute *LastUpdateTime* of the object *c* was equal to *Now* 4. The return value was **true**

Contract of **updateDataSet**:

```
Contract ManageModelService::updateDataSet(id : String, content : String) :
Boolean {
        definition:
            c:DataSet = DataSet.allInstance()->any(b:DataSet| b.Id = id)
        precondition:
            c.oclIsUndefined() = false
        postcondition:
            c.Id = id and
            c.Content = content and
            c.LastUpdateTime.isEqual(Now) and
            result = true
}
```

OP12 - modifyRunningModel

Operation Name:	modifyRunningModel
Operation ID:	OP12
Description:	
Service:	ManageRunningModelService
Input:	1. name: *runningModelId*, type: String 2. name: *assignedResource*, type: String 3. name: *api*, type: String
Output Type:	Boolean
Definition:	*a* is the object *e* in the instance set of class RunningModel. *e* represents an object of class RunningModel, and *e* meets: The attribute *Id* of the object *e* is equal to *runningModelId*
Preconditions:	The object *a* exists
Postconditions:	1. The attribute *Id* of the object *a* became *runningModelId* 2. The attribute *AssignedResource* of the object *a* became *assignedResource* 3. The attribute *API* of the object *a* became *api* 4. The return value was **true**

Contract of **modifyRunningModel**:

```
Contract ManageRunningModelService::modifyRunningModel(runningModelId : String,
assignedResource : String, api : String) : Boolean {
        definition:
            a:RunningModel = RunningModel.allInstance()->any(e:RunningModel|
e.Id = runningModelId)
        precondition:
            a.oclIsUndefined() = false
        postcondition:
            a.Id = runningModelId and
            a.AssignedResource = assignedResource and
            a.API = api and
            result = true
}
```

OP13 - modifyKey

Operation Name:	modifyKey
Operation ID:	OP13
Description:	
Service:	ManageRunningModelService
Input:	1. name: *keyId*, type: String 2. name: *isValid*, type: Boolean
Output Type:	Boolean
Definition:	*key* is the object *e* in the instance set of class Key. *e* represents an object of class Key, and *e* meets: The attribute *Id* of the object *e* is equal to *keyId*
Preconditions:	The object *key* exists
Postconditions:	1. The attribute *Id* of the object *key* became *keyId* 2. The attribute *IsValid* of the object *key* became *isValid* 3. The return value was **true**

Contract of **modifyKey**:

```
Contract ManageRunningModelService::modifyKey(keyId : String, isValid : Boolean)
: Boolean {
        definition:
            key:Key = Key.allInstance()->any(e:Key| e.Id = keyId)
        precondition:
            key.oclIsUndefined() = false
        postcondition:
            key.Id = keyId and
            key.IsValid = isValid and
            result = true
}
```

OP14 - generateURL

Operation Name:	generateURL
Operation ID:	OP14
Description:	
Service:	ThirdPartyServices
Input:	None
Output Type:	String
Preconditions:	None
Postconditions:	The return value was **null**

Contract of **generateURL**:

```
Contract ThirdPartyServices::generateURL() : String {
        precondition:
            true
        postcondition:
            result = ""
}
```

OP15 - download

Operation Name:	download
Operation ID:	OP15
Description:	
Service:	ThirdPartyServices
Input:	name: *url*, type: String
Output Type:	Boolean
Preconditions:	The *url* is not equal to **null**
Postconditions:	The return value was **true**

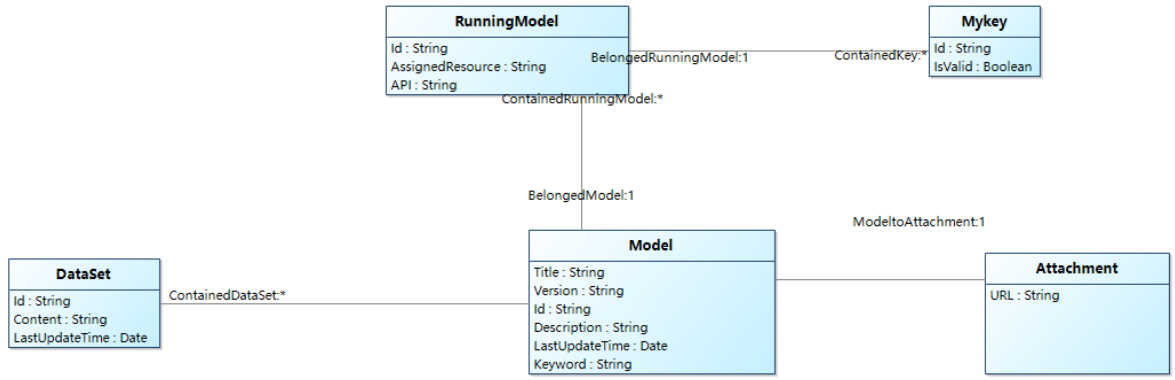
Contract of **download**:

```
Contract ThirdPartyServices::download(url : String) : Boolean {
        precondition:
            url <> ""
        postcondition:
            result = true
}
```

3.2 Database requirements

3.2.1 Entity Analysis

Conceptual Class Diagram



E1 - Model

Entity Name:	Model	
Entity ID:	E1	
Entity Description:		
Attribute Name	**Attribute Type**	**Attribute Description**
Title	String	The Title of Model
Version	String	The Version of Model
Id	String	The Id of Model
Description	String	The Description of Model
LastUpdateTime	LocalDate	The LastUpdateTime of Model
Keyword	String	The Keyword of Model
Relationship Name	**Related Entity**	**Relationship Type**
ContainedRunningModel	RunningModel	Association: One-to-Many
ContainedDataSet	DataSet	Association: One-to-Many
ModeltoAttachment	Attachment	Association: One-to-One

E2 - RunningModel

Entity Name:	RunningModel	
Entity ID:	E2	
Entity Description:		
Attribute Name	**Attribute Type**	**Attribute Description**
Id	String	The Id of RunningModel
AssignedResource	String	The AssignedResource of RunningModel
API	String	The API of RunningModel
Relationship Name	**Related Entity**	**Relationship Type**
ContainedKey	Key	Association: One-to-Many
BelongedModel	Model	Association: One-to-One

E3 - Key

Entity Name:	Key	
Entity ID:	E3	
Entity Description:		
Attribute Name	**Attribute Type**	**Attribute Description**
Id	String	The Id of Key
IsValid	Boolean	The IsValid of Key
Relationship Name	**Related Entity**	**Relationship Type**
BelongedRunningModel	RunningModel	Association: One-to-One

E4 - DataSet

Entity Name:	DataSet	
Entity ID:	E4	
Entity Description:		
Attribute Name	**Attribute Type**	**Attribute Description**
Id	String	The Id of DataSet
Content	String	The Content of DataSet
LastUpdateTime	LocalDate	The LastUpdateTime of DataSet

E5 - Attachment

Entity Name:	Attachment	
Entity ID:	E5	
Entity Description:		
Attribute Name	**Attribute Type**	**Attribute Description**
URL	String	The URL of Attachment

3.2.2 Other database requirements

This should specify the logical requirements for any information that is to be placed into a database. This may include the following:

- a) Types of information used by various functions;
- b) Frequency of use;
- c) Accessing capabilities;
- d) Integrity constraints;
- e) Data retention requirements.

3.3 Performance requirements

3.3.1 Static numerical requirements

This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole. Static numerical requirements may include the following:

- a) The number of terminals to be supported;
- b) The number of simultaneous users to be supported;
- c) Amount and type of information to be handled.

3.3.2 Dynamic numerical requirements

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these requirements should be stated in measurable terms.

For example,

- *95% of the transactions shall be processed in less than 1 s.*

rather than,

- *An operator shall not have to wait for the transaction to complete.*

NOTE:Numerical limits applied to one specifific function are normally specifified as part of the processing subparagraph description of that function.

3.4 Usability requirements

Define usability and quality in use requirements and objectives for the software system that can include measurable effectiveness, efficiency, satisfaction criteria and avoidance of harm that could arise from use in specific contexts of use.

3.5 Interface requirements

3.5.1 User interfaces

This should specify the following:

- a) The logical characteristics of each interface between the software product and its users. This includes those configuration characteristics (e.g., required screen formats, page or window layouts, content of any reports or menus, or availability of programmable function keys) necessary to accomplish the software requirements.
- b) All the aspects of optimizing the interface with the person who must use the system. This may simply comprise a list of do's and don'ts on how the system will appear to the user. One example may be a requirement for the option of long or short error messages. Like all others, these requirements should be verifiable, e.g., "a clerk typist grade 4 can do function X in Z min after 1 h of training" rather than "a typist can do function X." (This may also be specified in the Software System Attributes under a section titled Ease of Use.)

3.5.2 Hardware interfaces

This should specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics (number of ports, instruction sets, etc.). It also covers such matters as what devices are to be supported, how they are to be supported, and protocols. For example, terminal support may specify full-screen support as opposed to line-by-line support.

3.5.3 Software interfaces

This should specify the use of other required software products (e.g., a data management system, an operating system, or a mathematical package), and interfaces with other application systems (e.g., the linkage between an accounts receivable system and a general ledger system). For each required software product, the following should be provided:

- a) Name;
- b) Mnemonic;
- c) Specification number;
- d) Version number;
- e) Source.

For each interface, the following should be provided:

- a) Discussion of the purpose of the interfacing software as related to this software product.
- b) Definition of the interface in terms of message content and format. It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.

3.5.4 Communications interfaces

This should specify the various interfaces to communications such as local network protocols, etc.

3.6 Design constraints

Specify constraints on the system design imposed by external standards, regulatory requirements or project limitations.

3.6.1 Standards compliance

This subsection should specify the requirements derived from existing standards or regulations. They may include the following:

- a) Report format;
- b) Data naming;
- c) Accounting procedures;
- d) Audit tracing.

For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.

3.7 Software system attributes

3.7.1 Reliability

This should specify the factors required to establish the required reliability of the software system at time of delivery.

3.7.2 Availability

This should specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart.

3.7.3 Security

This should specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to

- a) Utilize certain cryptographical techniques;
- b) Keep specific log or history data sets;
- c) Assign certain functions to different modules;
- d) Restrict communications between some areas of the program;
- e) Check data integrity for critical variables.

3.7.4 Maintainability

This should specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices.

3.7.5 Portability

This should specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include the following:

- a) Percentage of components with host-dependent code;
- b) Percentage of code that is host dependent;
- c) Use of a proven portable language;
- d) Use of a particular compiler or language subset;
- e) Use of a particular operating system.

3.8 Supporting information

Additional supporting information to be considered includes:

- a) sample input/output formats, descriptions of cost analysis studies or results of user surveys;
- b) supporting or background information that can help the readers of the SRS;
- c) a description of the problems to be solved by the software; and
- d) special packaging instructions for the code and the media to meet security, export, initial loading or other requirements.

The SRS should explicitly state whether or not these information items are to be considered part of the requirements.

4 Verification

Provide the verification approaches and methods planned to qualify the software. The information items for verification are recommended to be given in a parallel manner with the information items in Section 3.

5 Appendices

5.1 Assumptions and dependencies

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

5.2 Apportioning of requirements

Apportion the software requirements to software elements. For requirements that will require implementation over multiple software elements, or when allocation to a software element is initially undefined, this should be so stated. A cross-reference table by function and software element should be used to summarize the apportionments.

Identify requirements that may be delayed until future versions of the system (e.g., blocks and/or increments).

5.3 Acronyms and abbreviations

This subsection should provide the acronyms and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.