Affiliation line 1
Affiliation line 2
Author line 1
Author line 2

MyProjectName : Your Title
Messir Analysis Document
- v 0.0 -

(*Report type: Specification*)

Tuesday 7th November, 2017 - 16:48

Contents

List of Figures

Listings

Chapter 1

Introduction

1.1 Overview

1.2 Purpose and recipients of the document

1.3 Application Domain

1.4 Definitions, acronyms and abbreviations

1.5 Document structure

Chapter 2

General Description

2.1 Domain Stakeholders

2.2 System's Actors

The objective of this section is not to provide the full requirement elicitation document in this section but to reuse a part of this document to provide a informal introduction to the Messir specification of the system under development. The use case model is made of a use case diagrams modelling abstractly and informally the actors and their use cases together with a set of use cases descriptions. In addition, those diagrams and description tables are adapted to the Messir specification since actor and messages names together with parameters are partly adapted to be consistent with the specification identifiers (see [1] for more details).

2.3 Use Cases Model

This section contains the use cases elicited during the requirements elicitation phase. The use cases are textually described as suggested by the Messir method and inspired by the standard Cokburn template [2].

2.3.1 Use Cases

2.3.1.1 usergoal-ugAddRoom

The goal is to add a new Room to the room database.

Use-Case Description	
Name	ugAddRoom
Scope	system
Level	usergoal
Primary actor(s)	
1	`actManager[active]`
Goal(s) description	
The goal is to add a new Room to the room database.	
Reuse	
1	`sfEnterFields [0..*]`
2	`sfAddRoom [0..*]`
Protocol condition(s)	
1	
Pre-condition(s)	
1	The user needs to be a manager to be able to add a new room.
Main post-condition(s)	
1	The new room is added to the database.
Main Steps	
a	the actor `actManager` executes the `sfEnterFields` use case
b	the actor `actManager` executes the `sfAddRoom` use case
Additional Information	
none	

Figure 2.1 The manager has the ability to add a room.

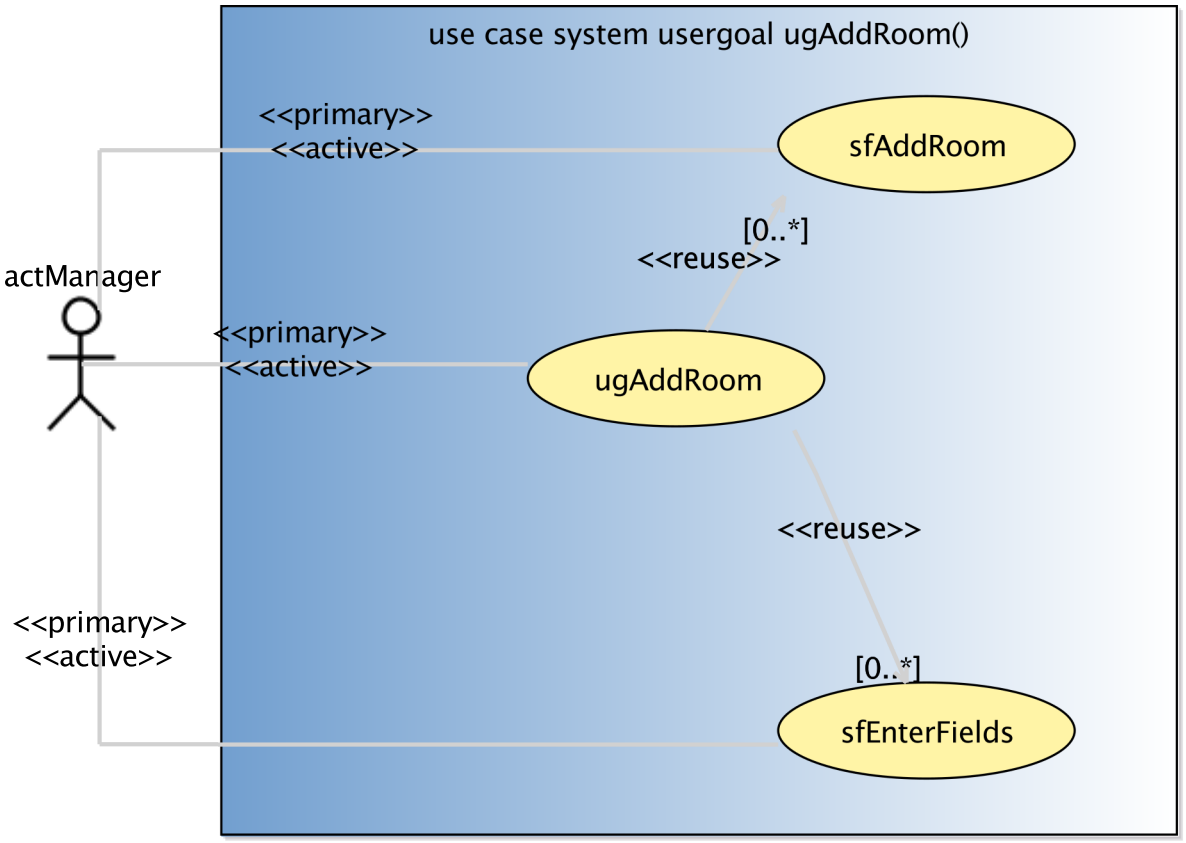

Figure 2.1: Add Room Function for the Manager Actor

2.3.1.2 usergoal-ugAskForFruitOrVegtable

The gardener has the ability to ask for a new fruit or vegetable

USE-CASE DESCRIPTION	
Name	ugAskForFruitOrVegtable
Scope	system
Level	usergoal
Primary actor(s)	
1	`actGardener[active]`
Goal(s) description	
The gardener has the ability to ask for a new fruit or vegetable	
Protocol condition(s)	
1	
Pre-condition(s)	
1	The gardener has to be logged in and has to be on his home screen.
Main post-condition(s)	
1	The request has been send to the manger.
Additional Information	
There are no constraints on the demand but there is a pop up which appears which has to be accepted to successfully send the request.	

2.3.1.3 usergoal-ugAskForSensor

The Technician goal is to request a sensor type to the manager.

USE-CASE DESCRIPTION	
Name	ugAskForSensor
Scope	system
Level	usergoal
Primary actor(s)	
1	`actTechnician[active]`
Goal(s) description	
The Technician goal is to request a sensor type to the manager.	
Reuse	
1	`oeEntersFields [1..1]`
2	`suRequestItem [1..1]`
Protocol condition(s)	
1	
Pre-condition(s)	
1	The gardener has to be logged in the system.
Main post-condition(s)	
1	The table request from manager is updated
Main Steps	
a	the actor `actTechnician` executes the `oeEntersFields` use case
b	the actor `actTechnician` executes the `suRequestItem` use case
Additional Information	
none	

Figure 2.2 The technician has the ability to request a sensor

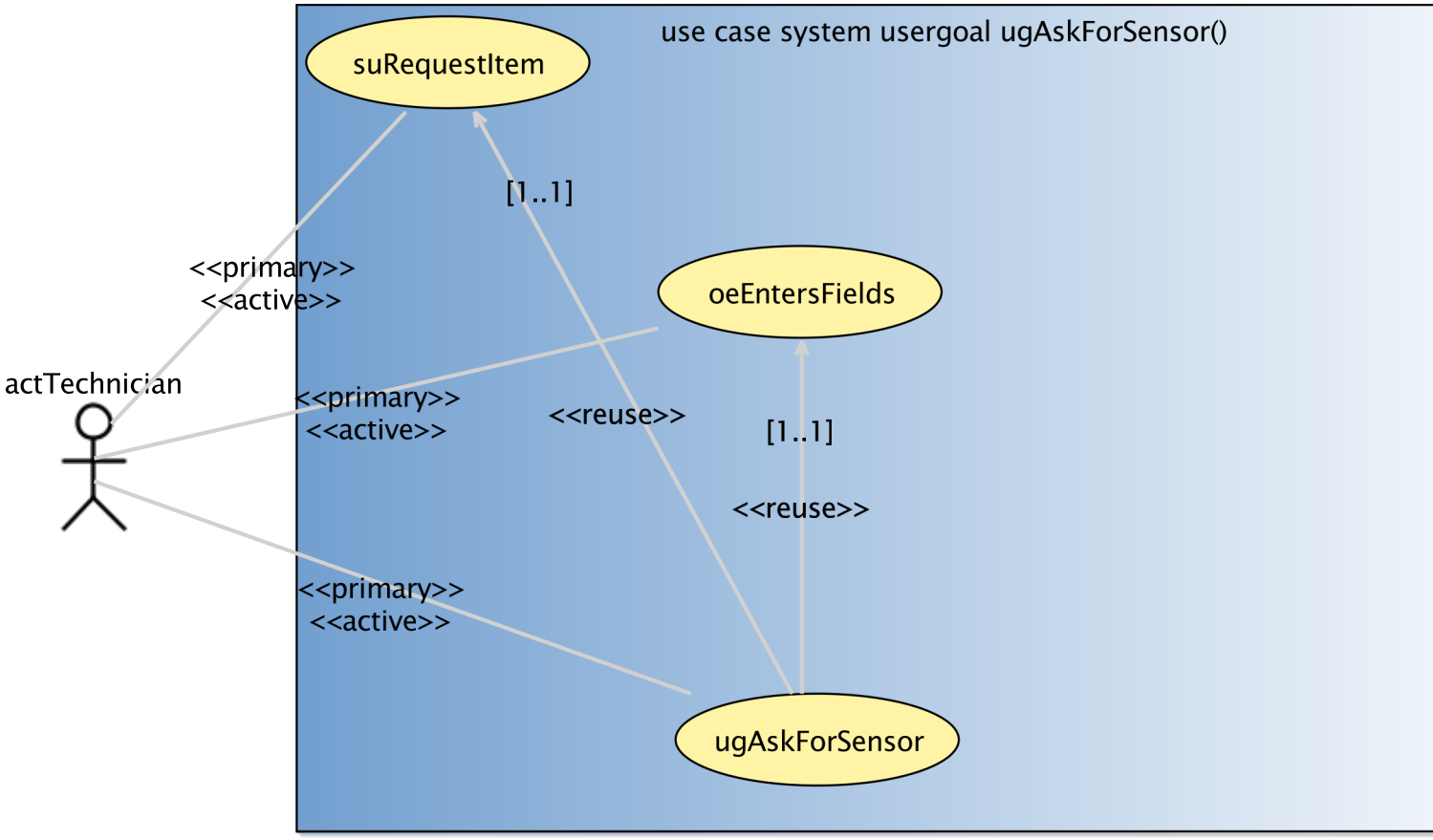

Figure 2.2: Technician requests sensor

2.3.1.4 usergoal-ugFillingGardnerSchedule

The goal is it to qdd tasks to the schedule

USE-CASE DESCRIPTION	
Name	ugFillingGardnerSchedule
Scope	system
Level	usergoal
Primary actor(s)	
1	`actManager[active]`
Goal(s) description	
The goal is it to qdd tasks to the schedule	
Reuse	
1	`sfFillingTextFields [0..*]`
2	`sfChooseImportance [0..1]`
Protocol condition(s)	
1	

continues in next page . . .

... Use-Case Description table continuation

Pre-condition(s)	
1	Manager must be securly logined and be on the manager screen.
Main post-condition(s)	
1	Schedule has an additonal task more tasks
Main Steps	
a	the actor `actManager` executes the `sfEnterFields` use case
b	the actor `actManager` executes the `sfChooseImportance` use case
Additional Information	
The input has to be write no sanity checks are done.	

Figure 2.3 The goal is it to add a task to the schedule

2.3.1.5 usergoal-ugSecurelyUseSystem

Every actor has the possibility to connect to the system but no other person from outside

USE-CASE DESCRIPTION	
Name	ugSecurelyUseSystem
Scope	system
Level	usergoal
Primary actor(s)	
1	`actUser[active]`
Goal(s) description	
Every actor has the possibility to connect to the system but no other person from outside	
Reuse	
1	`oeLogin [1..1]`
2	`oeLogout [1..1]`
Protocol condition(s)	
1	
Pre-condition(s)	
1	No actor has to be logged in.
Main post-condition(s)	
1	The given actor can execute he's tasks and can logout after.
Main Steps	
a	the actor `actUser` executes the `oeLogin` use case
b	the actor `actUser` executes the `oeLogout` use case
Steps Ordering Constraints	
1	step (a) must always precede step (b).
Additional Information	
The given actor has to logout him self.	

Figure 2.4 A given actor can securely login in.

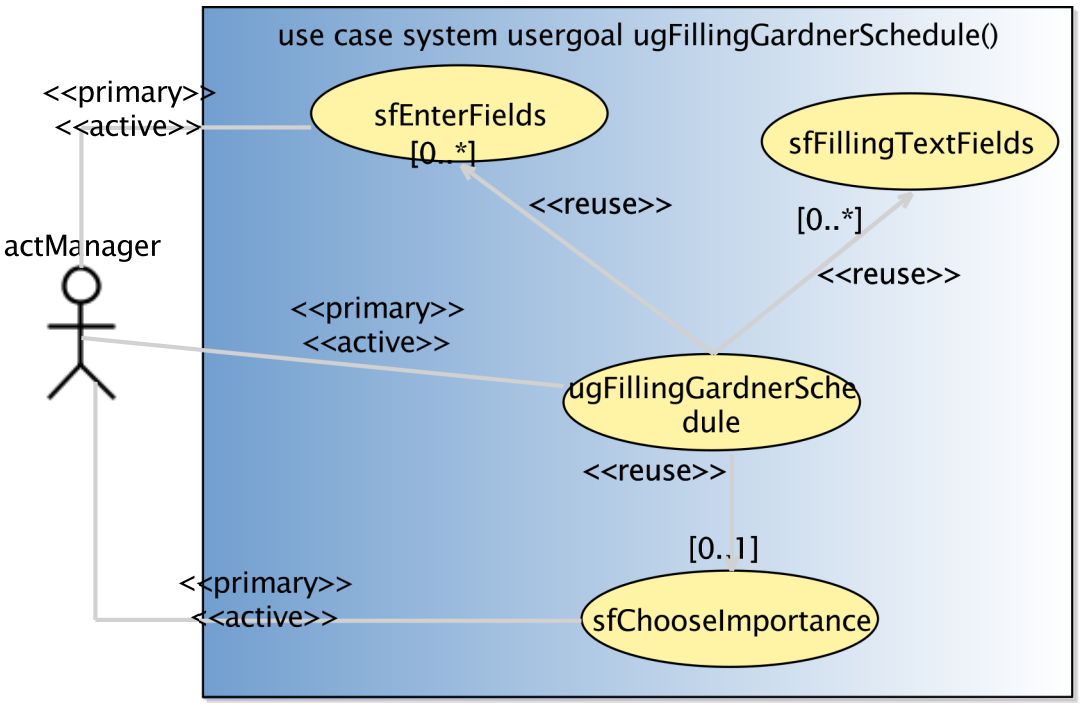

Figure 2.3: Adding task to schedule

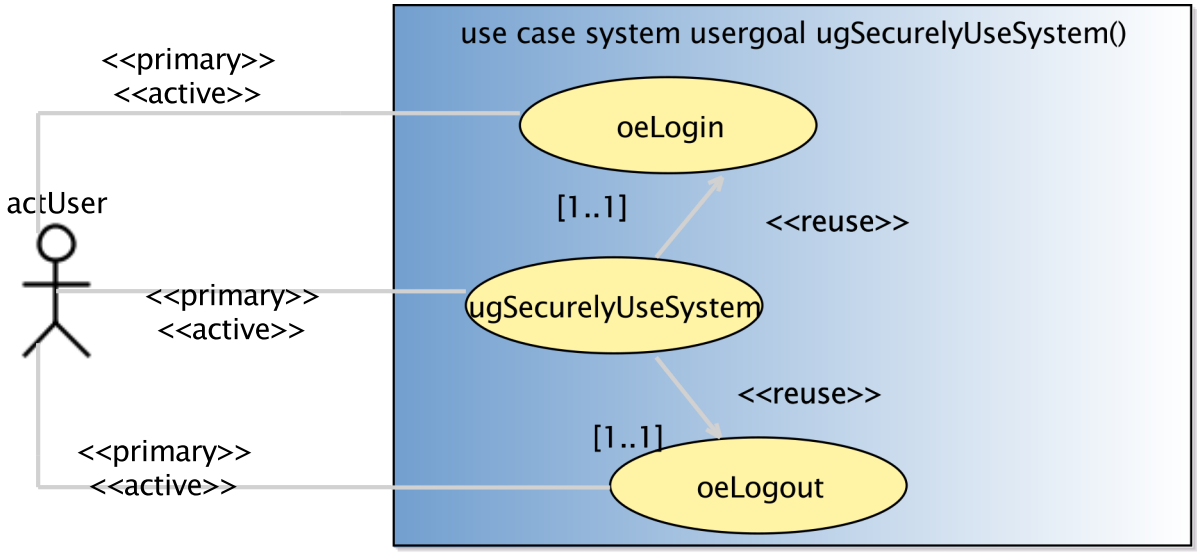

Figure 2.4:

2.3.1.6 subfunction-oeLogout

The currently signed in user is signed out by calling this function.

Use-Case Description	
Name	oeLogout
Scope	system
Level	subfunction
Primary actor(s)	
1	`actUser[active]`
Goal(s) description	
The currently signed in user is signed out by calling this function.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	Actor is logged in.
Main post-condition(s)	
1	Actor is logged out.
Additional Information	
The actor has to be logged in.	

2.3.1.7 subfunction-sfAddRoom

This function add the entry into the room database.

Use-Case Description	
Name	sfAddRoom
Scope	system
Level	subfunction
Primary actor(s)	
1	`actManager[active]`
Goal(s) description	
This function add the entry into the room database.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	The input formular needs to be correctly filled.
Main post-condition(s)	
1	The new room is now inside the room database.
Additional Information	
none	

2.3.1.8 subfunction-sfEnterFields

The Manager needs to enter the different information for the new room to be added.

Use-Case Description	
Name	sfEnterFields

continues in next page . . .

. . . Use-Case Description table continuation

Scope	system
Level	subfunction
Primary actor(s)	
1	`actManager[active]`
Goal(s) description	
The Manager needs to enter the different information for the new room to be added.	
Protocol condition(s)	
1	
Pre-condition(s)	
1	All fields must be filled out correctly.
Main post-condition(s)	
1	The input fields get cleared and the new room is now added to the database
Additional Information	
Room name must be unique. All other data must be numeric (Length, width, soil height)	

2.3.1.9 subfunction-sfcheckSensor

checksIfTheSensors are properly installed CHECK IF IT WORKS

USE-CASE DESCRIPTION	
Name	sfcheckSensor
Scope	system
Level	subfunction
Primary actor(s)	
1	`actTechnician[active]`
Goal(s) description	
checksIfTheSensors are properly installed CHECK IF IT WORKS	
Protocol condition(s)	
1	
Pre-condition(s)	
1	
Main post-condition(s)	
1	here is a post condition
Additional Information	
none	

2.3.1.10 subfunction-sfplating

plating the plants

USE-CASE DESCRIPTION	
Name	sfplating
Scope	system
Level	subfunction
Primary actor(s)	
1	`actGardener[active]`

continues in next page . . .

. . . Use-Case Description table continuation

Goal(s) description
plating the plants
Protocol condition(s)
1
Pre-condition(s)
1 Here is a preCondition
Main post-condition(s)
1
Additional Information
none

2.3.2 Use Case Instance(s)

2.3.2.1 Use-Case Instance - uciSecurelyUseSystem:ugSecurelyUseSystem

The view of the user

USERGOAL USE-CASE INSTANCE
Instantiated Use Case ugSecurelyUseSystem
Instance ID uciSecurelyUseSystem

Chapter 3

Environment Model

3.1 Environment model view(s)

There are no view(s) for the Messir environment model.

3.2 Actors and Interfaces Descriptions

We provide for the given views the description of the actors together with their associated input and output interface descriptions.

3.2.1 `actEcMeter` Actor

Actor
actEcMeter
The EC Meter is a sensor to control the water quality and gives information on the current nutrients inside the water. The EC Meter actor sends the information to the system.

3.2.2 `actGardener` Actor

Actor
actGardener
The gardener is a low authorized user able to view all data corresponding to their tasks or create requests which need to be approved by a higher authorized user.

3.2.3 `actHumiditySensor` Actor

Actor
actHumiditySensor
The humidity sensor controls the current humidity inside the soil. This information are concurrently sent to the system and saved inside a log.

3.2.4 `actLightSensor` Actor

Actor
actLightSensor
The light sensor measures the current luminosity level for the plants inside the greenhouse. The current spectrql level is concurrently sent to the system and saved inside a log.

3.2.5 `actManager` Actor

Actor
actManager
The manager is a high authorized user able to access and modify any information inside the software.

3.2.6 `actMotionSensor` Actor

Actor
actMotionSensor
The motion sensor recognizes movements of animals, humans or possible intruders. The information are sent to the system.

3.2.7 `actPhMeter` Actor

Actor
actPhMeter
The PH Meter is a sensor to measure the current Ph value from the water tank. This information are sent to the system.

3.2.8 `actTechnician` Actor

Actor
actTechnician
The technician is a low authorized user able to view all data corresponding to their tasks or create requests which need to be approved by a higher authorized user.

3.2.9 `actTemperatureSensor` Actor

Actor
actTemperatureSensor
The temperature sensor measures the current temperature inside the room. This information is sent to the system.

3.2.10 `actUser` Actor

Actor
actUser
The actor user is a user which represents all the Human actors inside our system

Chapter 4

Concept Model

4.1 Concept Model view(s)

There are no view(s) for the Messir concept model.

4.2 Concept Model Types Descriptions

This section provides the textual descriptions of all the types defined in the concept model and that can be part of the graphical views provided.

4.2.1 Primary types - Class types descriptions

There are no elements in this category in the system analysed.

4.2.2 Primary types - Datatypes types descriptions

The table below is providing comments on the graphical views given for the datatype types of the primary types.

Datatypes
dtExecussion executes an action
dtMessage Is a type of string/text

4.2.3 Primary types - Association types descriptions

There are no association types for the primary types.

4.2.4 Primary types - Aggregation types descriptions

There are no aggregation types for the primary types.

4.2.4.1 Primary types - Composition types descriptions

There are no composition types for the primary types.

4.2.5 Secondary types - Class types descriptions

There are no elements in this category in the system analysed.

4.2.6 Secondary types - Datatypes types descriptions

There are no elements in this category in the system analysed.

4.2.7 Secondary types - Association types descriptions

There are no association types for the secondary types.

4.2.8 Secondary types - Aggregation types descriptions

There are no aggregation types for the secondary types.

4.2.9 Secondary types - Composition types descriptions

There are no composition types for the secondary types.

Chapter 5

Operation Model

This section contains the operation schemes of each operation defined in either an actor, its output interface, in a primary or secondary type (class, datatype or enumeration types). The Messir OCL code listing is joined to the comment table.

5.1 Environment - Out Interface Operation Schemes

There are no elements in this category in the system analysed.

5.2 Environment - Actor Operation Schemes

There are no elements in this category in the system analysed.

5.3 Primary Types - Operation Schemes for Classes

There are no elements in this category in the system analysed.

5.4 Primary Types - Operation Schemes for Datatypes

There are no elements in this category in the system analysed.

5.5 Primary Types - Operation Schemes for Enumerations

There are no elements in this category in the system analysed.

5.6 Secondary Types - Operation Schemes for Classes

There are no elements in this category in the system analysed.

5.7 Secondary Types - Operation Schemes for Datatypes

There are no elements in this category in the system analysed.

5.8 Secondary Types - Operation Schemes for Enumerations

There are no elements in this category in the system analysed.

Chapter 6

Test Model(s)

There are no elements in this category in the system analysed.

Chapter 7

Additional Constraints

Appendix A

Undocumented Messir Specification Elements

A.1 Undocumented Use Cases

A.1.1 Undocumented Summary Level Use Cases

- lu.uni.lassy.excalibur.group01.excalibur.usecases.suRequestItem

A.1.2 Undocumented User-Goal Level Use Cases

- lu.uni.lassy.excalibur.group01.excalibur.usecases.ugAddANewSensorToTheRoom

A.1.3 Undocumented Subfunction Level Use Cases

- lu.uni.lassy.excalibur.group01.excalibur.usecases.oeEntersFields
- lu.uni.lassy.excalibur.group01.excalibur.usecases.oeLogin
- lu.uni.lassy.excalibur.group01.excalibur.usecases.oeRequestPushedToTable
- lu.uni.lassy.excalibur.group01.excalibur.usecases.sfChecksInputFields
- lu.uni.lassy.excalibur.group01.excalibur.usecases.sfChooseImportance
- lu.uni.lassy.excalibur.group01.excalibur.usecases.sfFillingTextFields
- lu.uni.lassy.excalibur.group01.excalibur.usecases.sfSendsRequest

A.1.4 Undocumented Use Case Views

- uc-sfplating

A.2 Undocumented Use Case Instances

A.2.1 Undocumented Use Case Instance Views

- uci-uciSecurelyUseSystem

A.3 Undocumented Actors

- lu.uni.lassy.excalibur.group01.excalibur.environment.actSystem

A.4 Undocumented Primary Types

A.4.1 Undocumented Primary Classe Types

- lu.uni.lassy.excalibur.group01.excalibur.concepts.primarytypes.classes.ctState

A.5 Undocumented Concept Model Views

- LionelSchroeder
- Marc

Appendix B

Messir Specification Files Listing

B.1 File ./src-gen/messir-spec/.views.msr

```
1 //
2 //DON'T TOUCH THIS FILE !!!
3 //
4 package uuid38ab2d5ea46c4d358fba679e4f14fa42 {
5  Concept Model {}
6 }
```

Listing B.1: Messir Spec. file .views.msr.

B.2 File ./src-gen/messir-spec/environment/environment.msr

```
1 /*
2 * @author Gaetan1991
3 * @date Wed Oct 11 14:39:06 CEST 2017
4 */
5
6 package lu.uni.lassy.excalibur.group01.excalibur.environment {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 Environment Model {
14
15 actor actTechnician role rnactTechnician cardinality[0 .. *] {
16    input interface inactTechnician {
17    }
18    output interface outactTechnician {
19    }
20   }
21  actor actGardener role rnactGardener cardinality[0 .. *] {
22    input interface inactGardener {
23    }
24    output interface outactGardener {
25    }
26   }
27  actor actSystem role rnactSystem cardinality[0 .. *] {
28    input interface inactSystem {
29    }
30    output interface outactSystem {
31    }
32   }
33  actor actManager role rnactManager cardinality[0 .. *] {
34    input interface inactManager {
35    }
36    output interface outactManager {
```

```
37    }
38   }
39  actor actUser role rnactUser cardinality[0 .. *] {
40    input interface inactUser {
41    }
42    output interface outactUser {
43    }
44   }
45  actor actTemperatureSensor role rnactTemperatureSensor cardinality[0 .. *] {
46    input interface inactTemperatureSensor {
47    }
48    output interface outactTemperatureSensor {
49    }
50   }
51  actor actHumiditySensor role rnactHumiditySensor cardinality[0 .. *] {
52    input interface inactHumiditySensor {
53    }
54    output interface outactHumiditySensor {
55    }
56   }
57  actor actLightSensor role rnactLightSensor cardinality[0 .. *] {
58    input interface inactLightSensor {
59    }
60    output interface outactLightSensor {
61    }
62   }
63  actor actPhMeter role rnactPhMeter cardinality[0 .. *] {
64    input interface inactPhMeter {
65    }
66    output interface outactPhMeter {
67    }
68   }
69  actor actEcMeter role rnactEcMeter cardinality[0 .. *] {
70    input interface inactEcMeter {
71    }
72    output interface outactEcMeter {
73    }
74   }
75  actor actMotionSensor role rnactMotionSensor cardinality[0 .. *] {
76    input interface inactMotionSensor {
77    }
78    output interface outactMotionSensor {
79    }
80   }
81  }
82 }
```

Listing B.2: Messir Spec. file environment.msr.

B.3 File ./src-gen/messir-spec/concepts/primarytypes-associations/primarytypes-associations.msr

```
1 /*
2 * @author Gaetan1991
3 * @date Wed Oct 11 14:39:06 CEST 2017
4 */
5
6 package lu.uni.lassy.excalibur.group01.excalibur.concepts.primarytypes.associations {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 Concept Model {
14
15  Primary Types {
16
```

```
17  }
18 }
19 }
```

Listing B.3: Messir Spec. file primarytypes-associations.msr.

B.4 File ./src-gen/messir-spec/concepts/primarytypes-classes/primarytypes-classes.msr

```
1 /*
2 * @author Gaetan1991
3 * @date Wed Oct 11 14:39:06 CEST 2017
4 */
5
6 package lu.uni.lassy.excalibur.group01.excalibur.concepts.primarytypes.classes {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 import lu.uni.lassy.messir.libraries.primitives
14
15 Concept Model {
16
17  Primary Types {
18
19   state class ctState {
20    attribute vpStarted: ptBoolean
21
22    operation init(AvpStarted:ptBoolean): ptBoolean
23   }
24
25  }
26 }
27 }
```

Listing B.4: Messir Spec. file primarytypes-classes.msr.

B.5 File ./src-gen/messir-spec/concepts/primarytypes-datatypes/primarytypes-datatypes.msr

```
1 /*
2 * @author Gaetan1991
3 * @date Wed Oct 11 14:39:06 CEST 2017
4 */
5
6 package lu.uni.lassy.excalibur.group01.excalibur.concepts.primarytypes.datatypes {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 Concept Model {
14
15  Primary Types {
16
17  datatype dtMessage {
18   attribute value : ptString
19    }
20  datatype dtExecussion {
21   attribute value : ptInteger
22    }
23   }
24 }
```

```
25 }
```

Listing B.5: Messir Spec. file primarytypes-datatypes.msr.

B.6 File ./src-gen/messir-spec/concepts/secondarytypes-associations/secondarytypes-associations.msr

```
1 /*
2 * @author Gaetan1991
3 * @date Wed Oct 11 14:39:06 CEST 2017
4 */
5
6 package lu.uni.lassy.excalibur.group01.excalibur.concepts.secondarytypes.associations {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 Concept Model {
14
15  Secondary Types {
16
17  }
18 }
19 }
```

Listing B.6: Messir Spec. file secondarytypes-associations.msr.

B.7 File ./src-gen/messir-spec/concepts/secondarytypes-classes/secondarytypes-classes.msr

```
1 /*
2 * @author Gaetan1991
3 * @date Wed Oct 11 14:39:06 CEST 2017
4 */
5
6 package lu.uni.lassy.excalibur.group01.excalibur.concepts.secondarytypes.classes {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 Concept Model {
14
15  Secondary Types {
16
17  }
18 }
19 }
```

Listing B.7: Messir Spec. file secondarytypes-classes.msr.

B.8 File ./src-gen/messir-spec/concepts/secondarytypes-datatypes/secondarytypes-datatypes.msr

```
1 /*
2 * @author Gaetan1991
3 * @date Wed Oct 11 14:39:06 CEST 2017
4 */
5
6 package lu.uni.lassy.excalibur.group01.excalibur.concepts.secondarytypes.datatypes {
```

```
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13 Concept Model {
14
15  Secondary Types {
16
17  }
18
19 }
20 }
```

Listing B.8: Messir Spec. file secondarytypes-datatypes.msr.

B.9 File ./src-gen/messir-spec/tests/tests.msr

```
1 /*
2 * @author Gaetan1991
3 * @date Wed Oct 11 14:39:06 CEST 2017
4 */
5
6 package lu.uni.lassy.excalibur.group01.excalibur.tests {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13  Test Model {
14
15  }
16
17 }
```

Listing B.9: Messir Spec. file tests.msr.

B.10 File ./src-gen/messir-spec/usecases/usecaseinstance-ugSecurelyUseSystem-uciSecurelyUseSystem.msr

```
1 package usecases.uci {
2  import lu.uni.lassy.excalibur.group01.excalibur.usecases
3
4  Use Case Model {
5
6   use case instance uciSecurelyUseSystem : ugSecurelyUseSystem{
7    actors {
8     bill:lu.uni.lassy.excalibur.group01.excalibur.environment.actUser
9
10    }
11    use case steps {
12     bill
13     executed instanceof subfunction
14     oeLogin("Greenhouse","GreenCODE"){
15      ieMessage('You are logged in.')returned to bill
16     }
17     bill
18     executed instanceof subfunction
19     oeLogout{
20      ieMessage('You are logged out!')returned to bill
21     }
22
23    }
24   }
25  }
```

```
26 }
```

Listing B.10: Messir Spec. file usecaseinstance-ugSecurelyUseSystem-uciSecurelyUseSystem.msr.

B.11 File ./src-gen/messir-spec/usecases/usecases.msr

```
1 /*
2 * @author Gaetan1991
3 * @date Wed Oct 11 14:39:06 CEST 2017
4 */
5
6 package lu.uni.lassy.excalibur.group01.excalibur.usecases {
7
8 import lu.uni.lassy.messir.libraries.calendar
9 import lu.uni.lassy.messir.libraries.math
10 import lu.uni.lassy.messir.libraries.primitives
11 import lu.uni.lassy.messir.libraries.string
12
13  Use Case Model {
14
15  use case system subfunction sfcheckSensor() {
16    actor lu.uni.lassy.excalibur.group01.excalibur.environment.actTechnician[primary, active]
17
18   }
19  use case system subfunction sfplating() {
20    actor lu.uni.lassy.excalibur.group01.excalibur.environment.actGardener[primary, active]
21   }
22  use case system subfunction oeLogin() {
23    actor lu.uni.lassy.excalibur.group01.excalibur.environment.actUser[primary, active]
24    returned messages{
25     ieMessage returned to lu.uni.lassy.excalibur.group01.excalibur.environment.actUser
26    }
27   }
28
29  use case system subfunction oeLogout() {
30    actor lu.uni.lassy.excalibur.group01.excalibur.environment.actUser[primary, active]
31    returned messages{
32     ieHLoggedOut returned to lu.uni.lassy.excalibur.group01.excalibur.environment.actUser
33    }
34   }
35
36  use case system subfunction oeEntersFields() {
37    actor lu.uni.lassy.excalibur.group01.excalibur.environment.actTechnician[primary, active]
38
39   }
40  use case system subfunction sfSendsRequest() {
41    actor lu.uni.lassy.excalibur.group01.excalibur.environment.actSystem[primary, active]
42    returned messages{
43
44    }
45   }
46  use case system subfunction sfChecksInputFields() {
47    actor lu.uni.lassy.excalibur.group01.excalibur.environment.actSystem[primary, active]
48   }
49  use case system subfunction oeRequestPushedToTable() {
50    actor lu.uni.lassy.excalibur.group01.excalibur.environment.actSystem[primary, active]
51   }
52  use case system summary suRequestItem() {
53   actor lu.uni.lassy.excalibur.group01.excalibur.environment.actTechnician[primary,active]
54  }
55  use case system usergoal ugSecurelyUseSystem() {
56   actor lu.uni.lassy.excalibur.group01.excalibur.environment.actUser[primary,active]
57
58   reuse oeLogin[1..1]
59   reuse oeLogout[1..1]
60
61   step a: lu.uni.lassy.excalibur.group01.excalibur.environment.actUser
62    executes oeLogin
63   step b: lu.uni.lassy.excalibur.group01.excalibur.environment.actUser
```

```
64    executes oeLogout
65
66   ordering constraint
67   "step (a) must always precede step (b)."
68   }
69
70  use case system usergoal ugAskForSensor() {
71   actor lu.uni.lassy.excalibur.group01.excalibur.environment.actTechnician[primary,active]
72   reuse oeEntersFields[1..1]
73   reuse suRequestItem[1..1]
74   step a: lu.uni.lassy.excalibur.group01.excalibur.environment.actTechnician
75   executes oeEntersFields
76   step b: lu.uni.lassy.excalibur.group01.excalibur.environment.actTechnician
77   executes suRequestItem()
78   }
79
80  use case system usergoal ugAskForFruitOrVegtable() {
81   actor lu.uni.lassy.excalibur.group01.excalibur.environment.actGardener[primary,active]
82
83   }
84  use case system usergoal ugAddANewSensorToTheRoom() {
85   actor lu.uni.lassy.excalibur.group01.excalibur.environment.actGardener[primary,active]
86   }
87  use case system usergoal ugAddRoom() {
88   actor lu.uni.lassy.excalibur.group01.excalibur.environment.actManager[primary, active]
89
90   reuse sfEnterFields[0..*]
91   reuse sfAddRoom[0..*]
92
93   step a:lu.uni.lassy.excalibur.group01.excalibur.environment.actManager
94   executes sfEnterFields()
95   step b:lu.uni.lassy.excalibur.group01.excalibur.environment.actManager
96   executes sfAddRoom()
97   }
98  use case system subfunction sfAddRoom() {
99    actor lu.uni.lassy.excalibur.group01.excalibur.environment.actManager[primary, active]
100  }
101 use case system subfunction sfEnterFields() {
102   actor lu.uni.lassy.excalibur.group01.excalibur.environment.actManager[primary, active]
103  }
104 use case system usergoal ugFillingGardnerSchedule() {
105  actor lu.uni.lassy.excalibur.group01.excalibur.environment.actManager[primary,active]
106
107  reuse sfFillingTextFields[0..*]
108  reuse sfChooseImportance[0..1]
109
110  step a: lu.uni.lassy.excalibur.group01.excalibur.environment.actManager
111  executes sfEnterFields()
112  step b: lu.uni.lassy.excalibur.group01.excalibur.environment.actManager
113  executes sfChooseImportance()
114
115  }
116 use case system subfunction sfFillingTextFields() {
117   actor lu.uni.lassy.excalibur.group01.excalibur.environment.actManager[primary, active]
118  }
119 use case system subfunction sfChooseImportance() {
120   actor lu.uni.lassy.excalibur.group01.excalibur.environment.actManager[primary, active]
121  }
122 }
123
124 }
```

Listing B.11: Messir Spec. file usecases.msr.

Bibliography

[1] Guelfi, N.: Messir: A Scientific Method for the Software Engineer. to be published (2017)

[2] Armour, F., Miller, G.: Advanced Use Case Modeling: Software Systems. Addison-Wesley (2001)