DOCUMENT ON SOFTWARE ANALYSIS AND DESIGNING

Project Title: “Multimodal Interface Directed by Action Sentences (MIDAS) – The use of Natural User Interaction (NUI) Devices”.

Software analysis and design were drafted by:

Name	Surname	E-mail
Grigorios	**Kalliatakis**	gkalliatakis@yahoo.gr
Efthimios	**Syntychakis**	pepemakis@hotmail.com
Petros	**Varchalamas**	sonor_7@hotmail.com
Anastasios	**Vlasopoulos**	anvlasop@hotmail.com

Day & Time of Course:

Friday 9:15-14:00
Winter Semester 2013-2014
M.Sc in Informatics and Multimedia

Under the supervision of Prof. Dr Vidakis Nikolaos

Table of Contents

Chapter 1 – Textual Analysis-A tool to design connectivity with NUI Devices

Textual analysis is a tool for recording customers' needs. Furthermore, it lets you extract key terms from a passage you recorded, and transform the terms to model elements or put them into glossary to build a project - based dictionary. It consists of:

i. Recording requirements

Document customers' needs by performing textual analysis.

ii. Identifying important terms

Shows you how to identify glossary term from a passage recorded by textual analysis.

iii. Identifying candidate objects

Shows you how to identify candidate model element from a passage. You selectively convert candidate to actual model element and visualize it in diagram.

iv. Forming diagram from candidate objects

Shows you how to visualize candidate elements in a diagram.

v. Candidate pane view

A view that shows candidate elements in visualized form - as boxes.

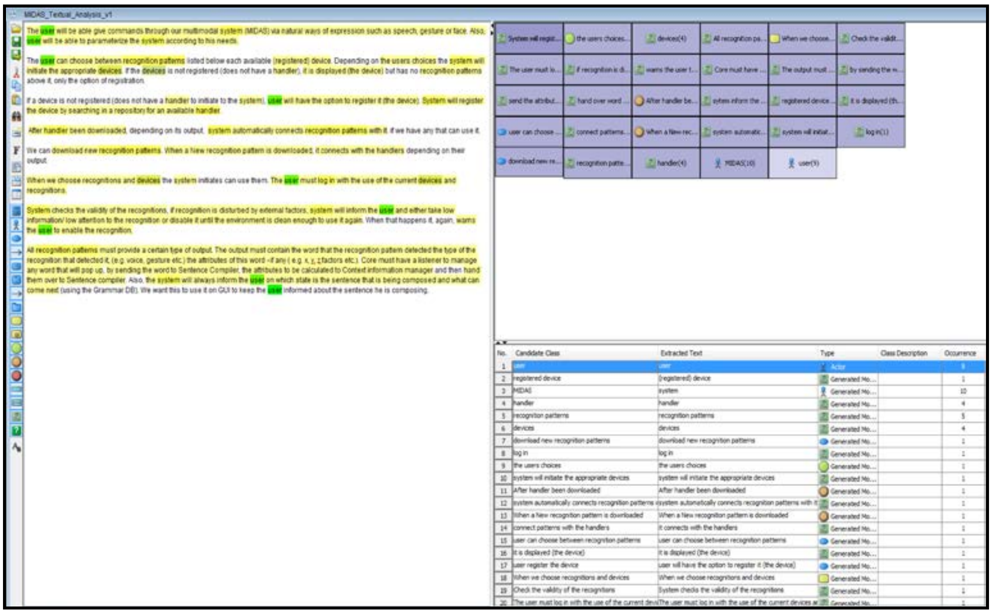
The system will enable the user to choose between recognition patterns listed below each available (registered) device. Depending on the users choices the system will initiate the appropriate devices. If the device is not registered (does not have a handler), it is displayed but has no recognition patterns above it, only the option of registration.

If a device is not registered (does not have a handler to initiate to the system) we can search online for an available. Depending on the handlers output, system automatically connects recognition patterns if we have any that can use it. We can download new recognition patterns. When a New recognition pattern is downloaded, it connects with the handlers depending on their output.

When we choose recognitions and devices the system initiates. The user must log in with the use of the current devices and recognitions.

System checks the validity of the recognitions, if recognition is disturbed by external factors, system will inform the user and either take low information/ low attention to the recognition or disable it until the environment is clean enough to use it again. When that happens it, again, warns the user to enable the recognition.

All recognition patterns will provide a certain type of output. The output will have the word that the recognition pattern detected the type of the recognition that detected it, (e.g. voice, gesture etc.) the attributes of this word –if any (e.g. x, y, z factors etc.). Core will have a listener to manage any word that will pop up, by sending the word to Sentence Compiler, the attributes to be calculated to Context information manager and then hand them over to Sentence compiler. Also, Sentence compiler will always inform the system on which state is the sentence that he's composing and what can come next (using the Grammar DB). We want this to use it on GUI to keep the user informed about the sentence he is composing.



MIDAS_Textual_Analysis_v1

The user will be able give commands through our multimodal system (MIDAS) via natural ways of expression such as speech, gesture or face. Also, user will be able to parameterize the system according to his needs.

The user can choose between recognition patterns listed below each available (registered) device. Depending on the users choices the system will initiate the appropriate devices. If the devices is not registered (does not have a handler), it is displayed (the device) but has no recognition patterns above it, only the option of registration.

If a device is not registered (does not have a handler to initiate to the system), user will have the option to register it (the device). System will register the device by searching in a repository for an available handler.

After handler been downloaded, depending on its output, system automatically connects recognition patterns with it if we have any that can use it.

We can download new recognition patterns. When a New recognition pattern is downloaded, it connects with the handlers depending on their output.

When we choose recognitions and devices the system initiates can use them. The user must log in with the use of the current devices and recognitions.

System checks the validity of the recognitions, if recognition is disturbed by external factors, system will inform the user and either take low information/ low attention to the recognition or disable it until the environment is clean enough to use it again. When that happens it, again, warns the user to enable the recognition.

All recognition patterns must provide a certain type of output. The output must contain the word that the recognition pattern detected the type of the recognition that detected it, (e.g. voice, gesture etc.) the attributes of this word –if any (e.g. x, y, z factors etc.). Core must have a listener to manage any word that will pop up, by sending the word to Sentence Compiler, the attributes to be calculated to Context information manager and then hand them over to Sentence compiler. Also, the system will always inform the user on which state is the sentence that is being composed and what can come next (using the Grammar DB). We want this to use it on GUI to keep the user informed about the sentence he is composing.

No.	Candidate Class	Extracted Text	Type	Class Description	Occurrence
1	user	user	Actor		9
2	registered device	(registered) device	Generated Mo...		1
3	MIDAS	system	Generated Mo...		10
4	handler	handler	Generated Mo...		4
5	recognition patterns	recognition patterns	Generated Mo...		5
6	devices	devices	Generated Mo...		4
7	download new recognition patterns	download new recognition patterns	Generated Mo...		1
8	log in	log in	Generated Mo...		1
9	the users choices	the users choices	Generated Mo...		1
10	system will initiate the appropriate devices	system will initiate the appropriate devices	Generated Mo...		1
11	After handler been downloaded	After handler been downloaded	Generated Mo...		1
12	system automatically connects recognition patterns	system automatically connects recognition patterns with it	Generated Mo...		1
13	When a New recognition pattern is downloaded	When a New recognition pattern is downloaded	Generated Mo...		1
14	connect patterns with the handlers	it connects with the handlers	Generated Mo...		1
15	user can choose between recognition patterns	user can choose between recognition patterns	Generated Mo...		1
16	it is displayed (the device)	it is displayed (the device)	Generated Mo...		1
17	user register the device	user will have the option to register it (the device)	Generated Mo...		1
18	When we choose recognitions and devices	When we choose recognitions and devices	Generated Mo...		1
19	Check the validity of the recognitions	System checks the validity of the recognitions	Generated Mo...		1
20	The user must log in with the use of the current devi	The user must log in with the use of the current devices a	Generated Mo...		1

Figure 1 Textual Analysis

Chapter 2 – Requirements Capturing

[Section 1] Relation between the user and the system

The user can interact with the computer by using Natural User Interaction devices, which support different recognition patterns.

- ➢ The user will be allowed to choose the recognition pattern to be used
- ➢ The system should inform the user for the registered (having a handler) devices, thus indicating what kind of recognition may be implemented
- ➢ There must be the capability by the system to register an unknown device by downloading (from the Net), the suitable handlers
- ➢ The user will be able to perform system log in by using the preinstalled NUI devices, or in a different way (if no NUI devices are installed)

[Section 2] Relation between Recognition Library and Handlers

The system will be able to attach recognition patterns with NUI devices.

- ➢ The system will automatically connect available handlers (registered devices) with the competent recognition pattern.
- ➢ Newly downloaded handlers should be able to automatically connect to appropriate recognition patterns, in the same way.

[Section 3] Relation between Environment and the System

The system should be able to take into consideration, potential changes that may occur in the environment.

- Environmental changes affecting recognitions made by the system should be marked in a semantic way.
- The system could be adaptive in a way to prevent the user from choosing low-quality recognitions.

Below, Image 2 shows the requirements as identified from the textual analysis which was preceded.

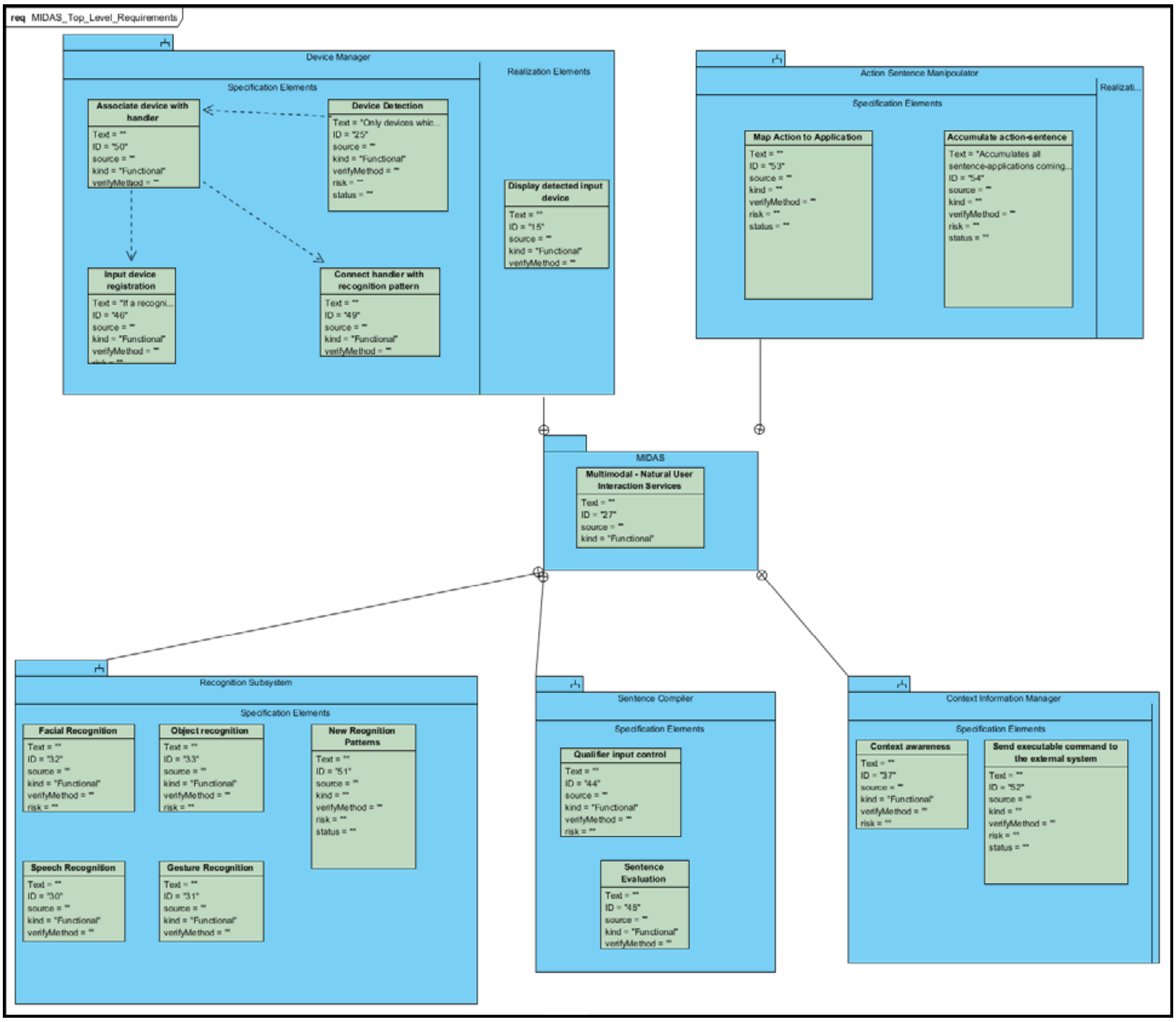


Figure 2 Top-level Requirements resulting from Textual Analysis

Chapter 3 – Use Cases

The use case modeling approach keeps software developments focused on what user wants to do (user goal) rather than what features is going to develop. Use case diagram provides a crystal clear presentation for system analyst to see all user goals (use cases) and related end-users (actors).

For this purpose, we came up with several use cases being part of the overall system usage example. The first case has to do with how devices are managed by the system. So the user in order to register a new (to the system) NUI device, device detection is required from the system, before completing the association with the corresponding handler. Another case would be the actual management of registered devices by the user. In such a case, the user would be able to select the desired recognition pattern. Furthermore, the user could view all available recognitions patterns and devices, as our last use case of the managing device process.

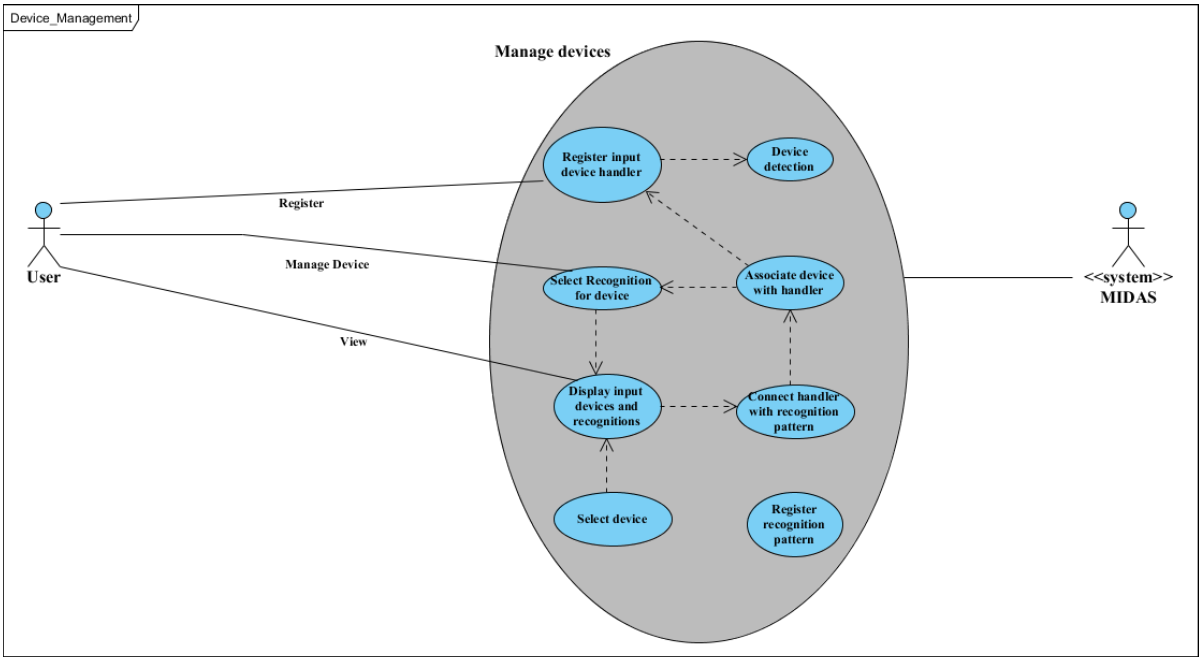

Device_Management
Manage devices
Register input device handler
Device detection
User
Register
Manage Device
View
Select Recognition for device
Associate device with handler
<<system>>
MIDAS
Display input devices and recognitions
Connect handler with recognition pattern
Select device
Register recognition pattern

Figure 3 Device Management Use Case

Another case is the natural interaction the user can have with the system. The use case illustrated here is when the user logs in to the system, using registered NUI devices and installed pattern recognitions.

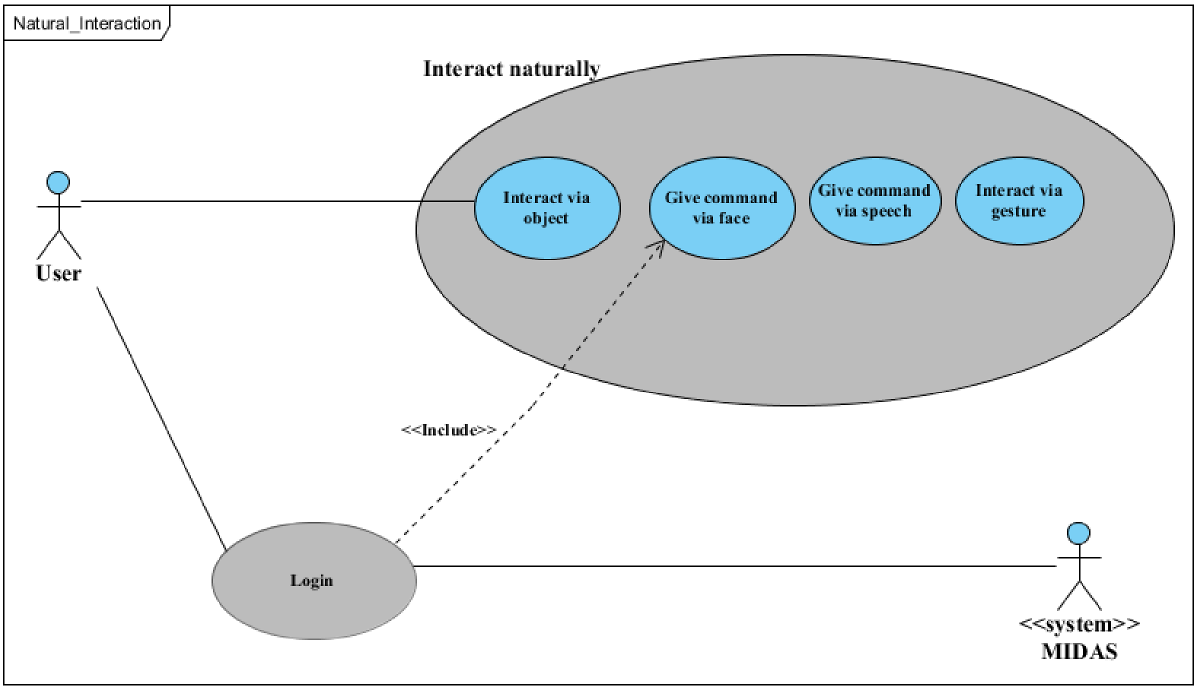


Figure 4 Natural Interaction Use Case

Below we present the overall system's use case which is built upon the aforementioned sub-use cases.

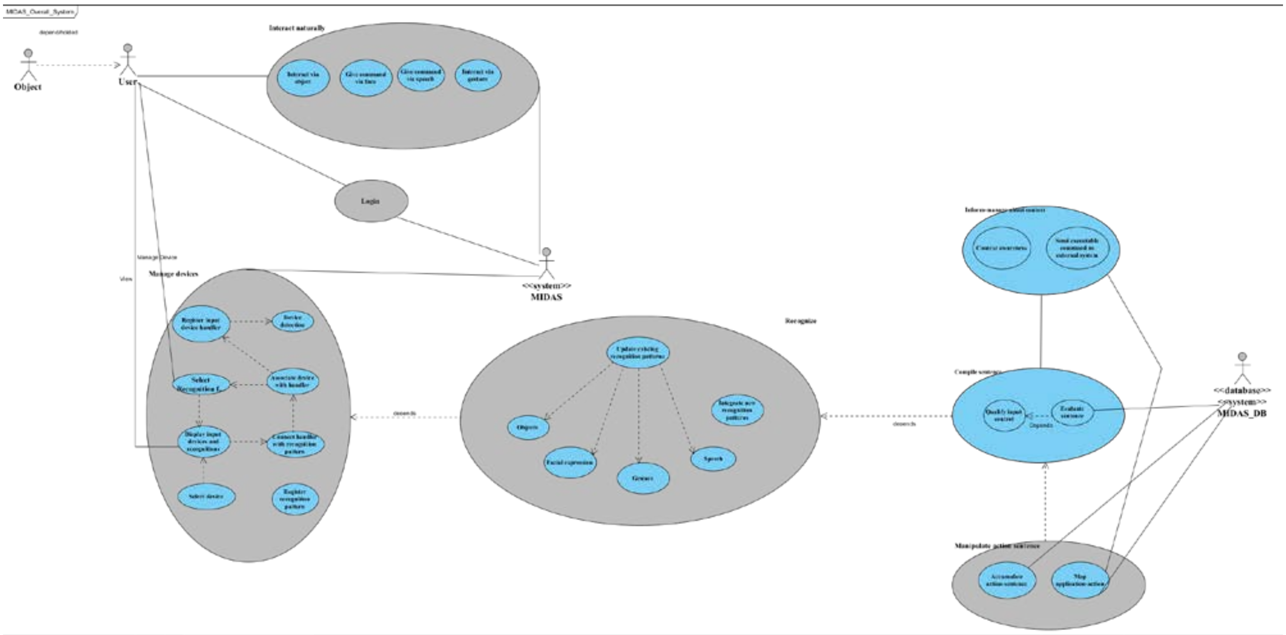


Figure 5 An overall system's Use Case

Chapter 4 – Class Diagram

The resulting class diagram contains three different packages, the “Device Manager” , the “Recognition Library” and the “MIDAS Core”. The model shown in the diagram below expresses that in order to manage the different NUI devices that will be imported to the system, they need to be connected to an appropriate handler.
Moreover, the recognition library indicates the different recognitions patterns that can be connected with different NUI devices as well as new recognition patterns that can be downloaded by the users. Finally, the association between MIDAS core and the operation system is shown in the last package, which contains the main operation to connect with the operating system the context information manager Also it is shown how recognition manager is connected with MIDAS core. All relationships outlined above are presented in Figure 6.

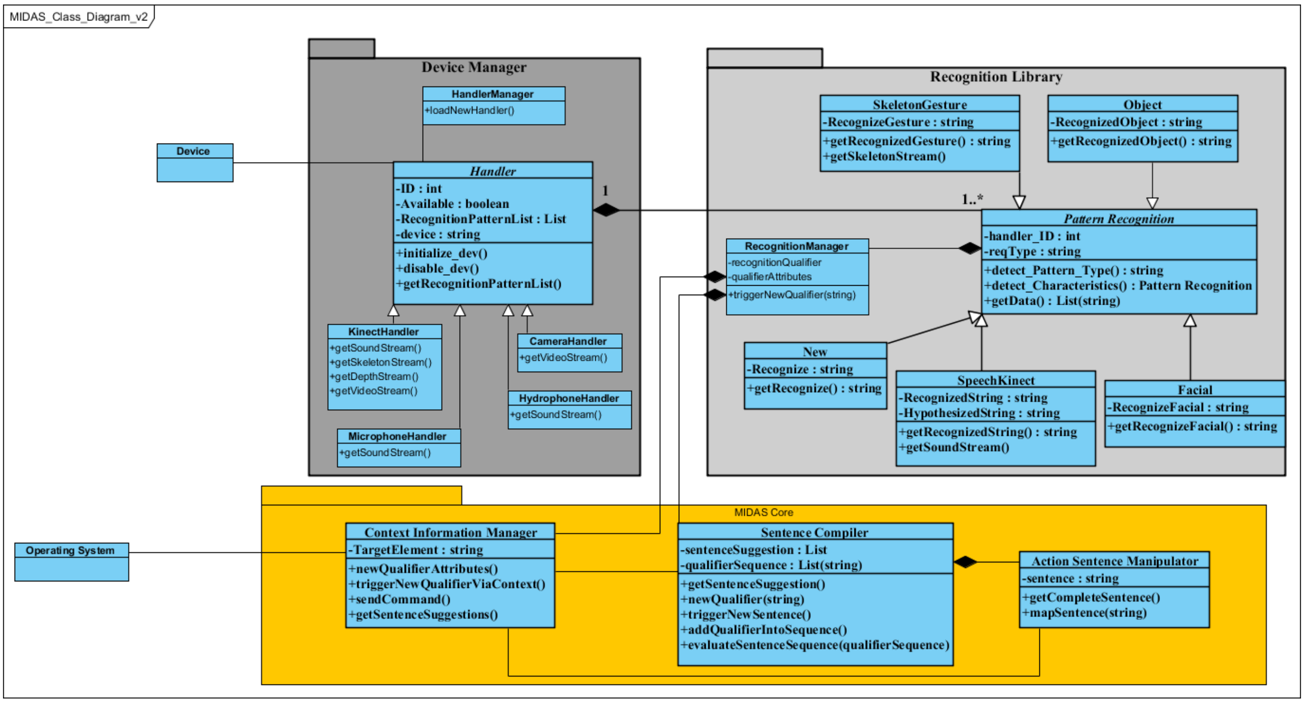


Figure 6 Class Diagram of the system's architecture

Chapter 5 – Sequence Diagram

In this sequence diagram is presented the procedure for the physical interaction of the user with the MIDAS system (Multimodal Interface Directed by Action Sentences) and by extension to the operating system, based on the use case of «Put That There», using Microsoft Kinect.

Here is a description of the steps followed from 1 to 46.

In steps 1-4, the initialization of the multimodal input device Kinect, starts through the Kinect_Handler (each type of device connected to MIDAS is initialized through its custom handler). Sound, depth and skeleton stream pass through the handler in order to be transferred in recognition patterns as input. In steps 5 and 6 speech recognition receives sound stream, as gesture recognition the skeleton stream.

The user as an actor of the procedure can start the interaction by saying the word "This", in step 7.

This command is a bit tricky because it contains attributes from both speech and gesture recognition, speaking and pointing. Specifically, speech recognition informs recognition manager that the word "This" is detected in step 8. Recognition manager requests data from gesture recognition that assembles the qualifier and its attributes in step 9. In step 10 the new qualifier is sent to the sentence compiler and the attributes to context information manager (CIM) as shown in step 11. Continuing in step 12, CIM makes a request in the operating system to get the item's context that the user pointed at.

In steps 13 the qualifier from step 10 is added into the sentence and in step 14 a check of the completion of the sentence is done.

The attributes from the previous request, in step 12, in this case the x, y coordinates of the item that the user pointed at, are granted from the operating system and the CIM gets them in step 15, to send them as a new qualifier in sentence compiler in step 16.

During steps 17 and 18 the qualifier is added in sentence and then continues with the check of the completion of the sentence. Since the user hasn't complete a sentence, that part of the sentence with its attributes is sent to the CIM in step 19 ,which finally updates the interface of the operating system.

The whole procedure from step 21 to 34 is repeated for the word "There" and from 35 to 46 for the word "Put" with the difference that the word "Put" doesn't require any other attributes so recognition manager sends the new qualifier to sentence compiler and null qualifier attributes to context information manager.

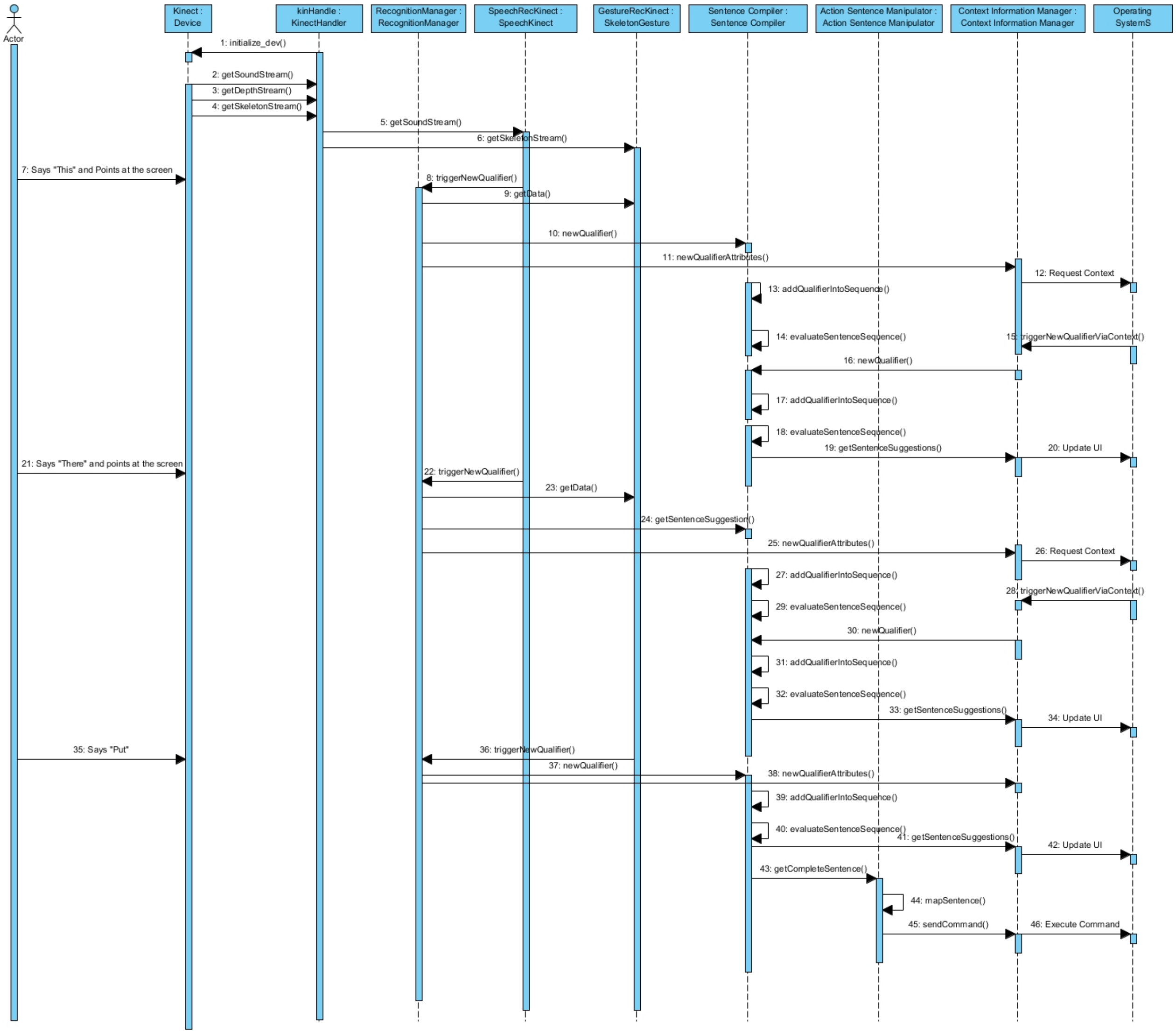

Actor
Kinect : Device
kinHandle : KinectHandler
RecognitionManager : RecognitionManager
SpeechRecKinect : SpeechKinect
GestureRecKinect : SkeletonGesture
Sentence Compiler : Sentence Compiler
Action Sentence Manipulator : Action Sentence Manipulator
Context Information Manager : Context Information Manager
Operating SystemS
1: initialize_dev()
2: getSoundStream()
3: getDepthStream()
4: getSkeletonStream()
5: getSoundStream()
6: getSkeletonStream()
7: Says "This" and Points at the screen
8: triggerNewQualifier()
9: getData()
10: newQualifier()
11: newQualifierAttributes()
12: Request Context
13: addQualifierIntoSequence()
14: evaluateSentenceSequence()
15: triggerNewQualifierViaContext()
16: newQualifier()
17: addQualifierIntoSequence()
18: evaluateSentenceSequence()
19: getSentenceSuggestions()
20: Update UI
21: Says "There" and points at the screen
22: triggerNewQualifier()
23: getData()
24: getSentenceSuggestion()
25: newQualifierAttributes()
26: Request Context
27: addQualifierIntoSequence()
28: triggerNewQualifierViaContext()
29: evaluateSentenceSequence()
30: newQualifier()
31: addQualifierIntoSequence()
32: evaluateSentenceSequence()
33: getSentenceSuggestions()
34: Update UI
35: Says "Put"
36: triggerNewQualifier()
37: newQualifier()
38: newQualifierAttributes()
39: addQualifierIntoSequence()
40: evaluateSentenceSequence()
41: getSentenceSuggestions()
42: Update UI
43: getCompleteSentence()
44: mapSentence()
45: sendCommand()
46: Execute Command

Figure 7 Physical Interaction Sequence Diagram

Chapter 6 – Component Diagram

Component diagrams are different in terms of nature and behavior. Component diagrams are used to model physical aspects of a system. Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node.
So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities. So from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files etc.
Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.
So the purpose of the component diagram can be summarized as:

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

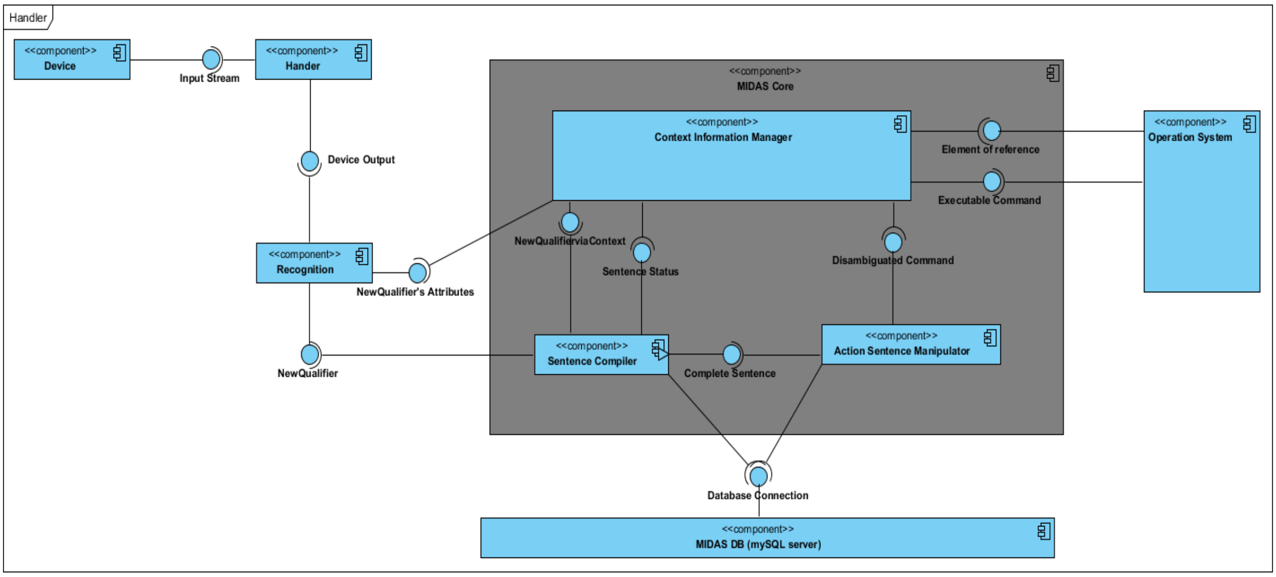


Figure 8 Handler Component Diagram

Chapter 7 – Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related. Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.

UML is mainly designed to focus on software artifacts of a system. But these two diagrams are special diagrams used to focus on software components and hardware components. So most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on hardware topology of a system. Deployment diagrams are used by the system engineers.

The purpose of deployment diagrams can be described as:

- Visualize hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe runtime processing nodes.

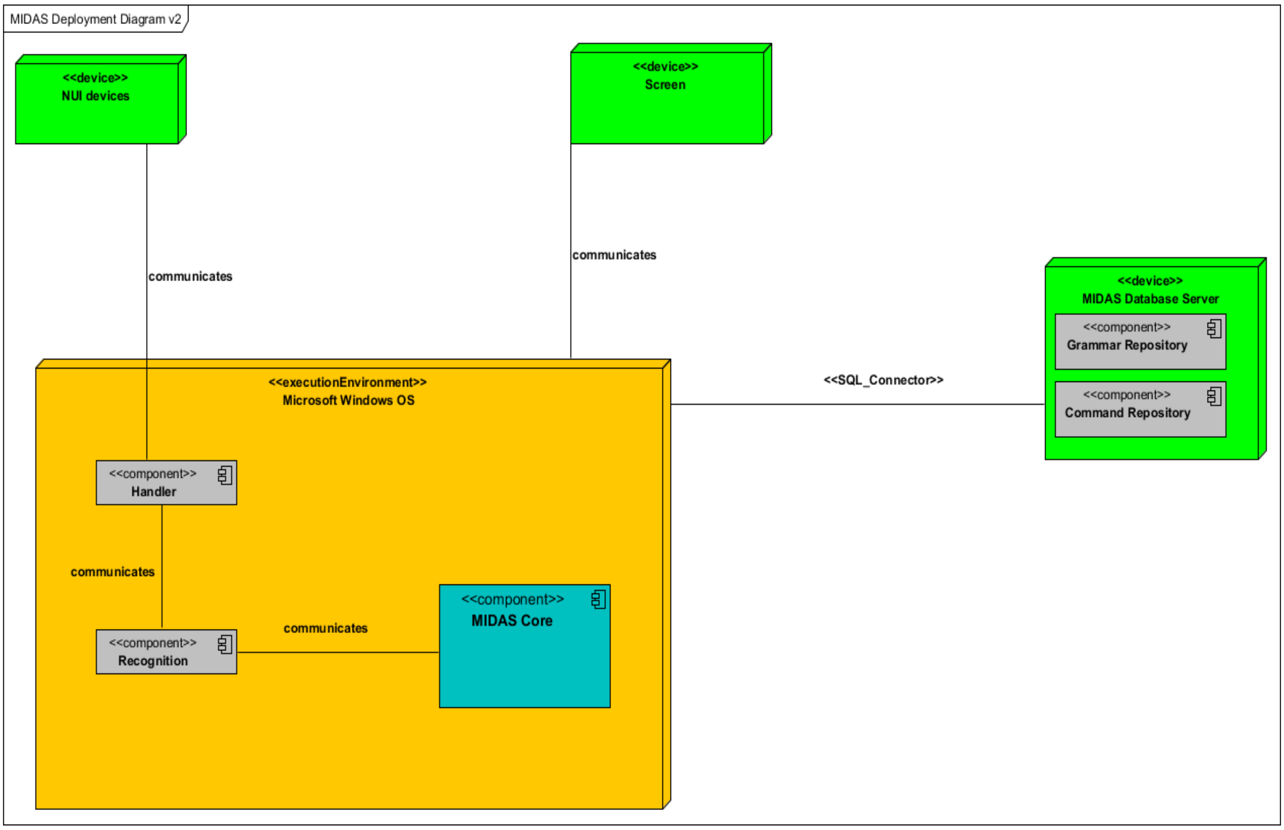


Figure 9 Deployment Diagram

Chapter 8 – Package Diagram

Package diagram visualizes packages and depicts the dependency, import, access, generalization, realization and merge relationships between them. Package diagram enables you to gain a high level understanding of the collaboration among model elements through analyzing the relationships among their parent package. This also helps explain the system's architecture from a broad view.

UML package helps to organize and arrange model elements and diagrams into logical groups, through which you can manage a chunk of project data together. You can also use packages to present different views of the system's architecture. In addition, developers can use package to model the physical package or namespace structure of the application to build. Package diagrams can use packages containing use cases to illustrate the functionality of a software system.

Package diagrams can use packages that represent the different layers of a software system to illustrate the layered architecture of a software system. The dependencies between these packages can be adorned with labels / stereotypes to indicate the communication mechanism between the layers.

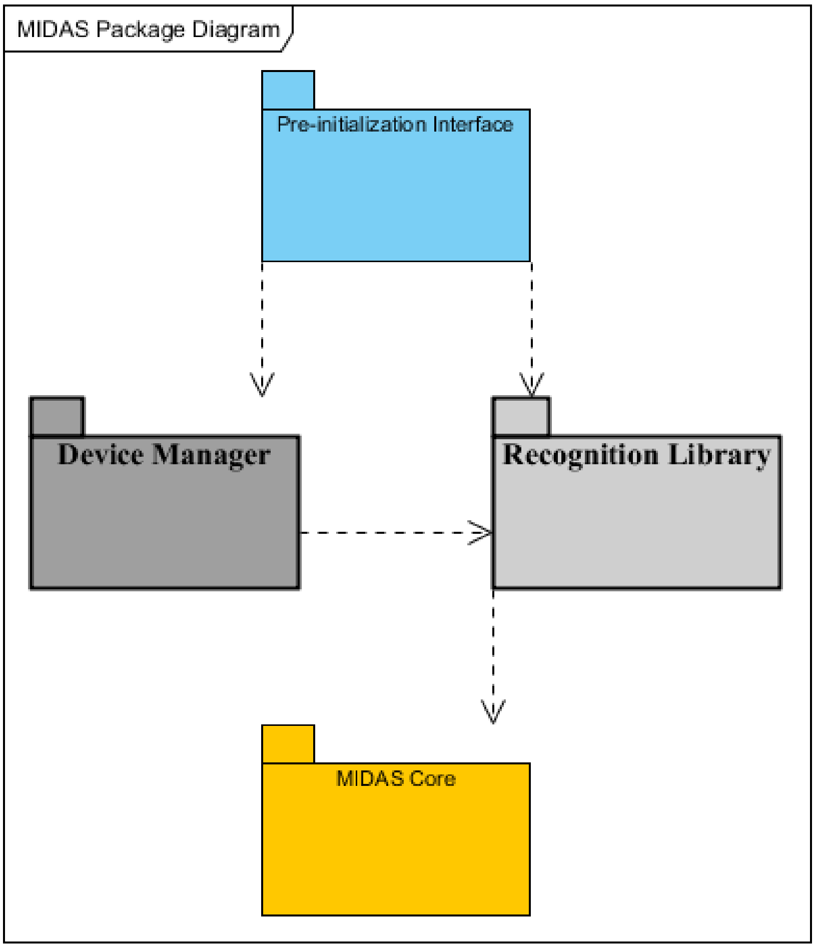


Figure 10 Package Diagram