D3.6

Software Prototype with Sensor Fusion API Specification and Usage Description

Editors: Roland Klemke (OUNL)
## Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Contributor(s)</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01-02-2016</td>
<td>Roland Klemke</td>
<td>Initial Outline</td>
</tr>
<tr>
<td>2</td>
<td>12-02-2018</td>
<td>Daniele di Mitri, Bibeg Limbu, Roland Klemke, Puneet Sharma</td>
<td>Re-structuring</td>
</tr>
<tr>
<td>3</td>
<td>19-02-2018</td>
<td>Daniele di Mitri</td>
<td>WEKIT.One Hub, WEKIT.One Cloud</td>
</tr>
<tr>
<td>4</td>
<td>20-02-2018</td>
<td>Bibeg Limbu</td>
<td>WEKIT.One Recorder, How To</td>
</tr>
<tr>
<td>5</td>
<td>22-02-2018</td>
<td>Roland Klemke</td>
<td>Introduction, Methodology, Architecture</td>
</tr>
<tr>
<td>6</td>
<td>26-02-2018</td>
<td>Daniele di Mitri, Bibeg Limbu, Roland Klemke, Puneet Sharma</td>
<td>Integrated version</td>
</tr>
<tr>
<td>7</td>
<td>27-02-2018</td>
<td>Daniele di Mitri, Bibeg Limbu, Roland Klemke, Puneet Sharma, Mikhail Fominykh</td>
<td>Final version</td>
</tr>
</tbody>
</table>

Disclaimer: All information included in this document is subject to change without notice. The Members of the WEKIT Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the WEKIT Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.
Software Prototype with Sensor Fusion API Specification and Usage Description

WP 3 | D3.6

Editors:
Roland Klemke (OUNL)

Authors:
Daniele Di Mitri (OUNL), Bibeg Limbu (OUNL), Puneet Sharma (UiT)

Reviewers:
Mikhail Fominykh (EP)

Deliverable number: D3.6
Dissemination level: Public
Version: 1.0
Status: draft
Date: 28.02.2018
Due date: M27
Table of Contents

REVISION HISTORY ..........................................................................................................................2
EXECUTIVE SUMMARY ..................................................................................................................5
1. INTRODUCTION ............................................................................................................................6
2. METHODOLOGY ..............................................................................................................................6
3. SOFTWARE PROTOTYPE DESCRIPTION .........................................................................................6
   3.1. WEKIT.ONE RECORDER ........................................................................................................8
   3.2. WEKIT.ONE HUB ....................................................................................................................12
      3.2.1. Multimodal recording of a learning task ...........................................................................13
      3.2.2. Data Storing ......................................................................................................................14
      3.2.3. Multimodal data synchronization ...................................................................................15
      3.2.4. Immediate Feedback .......................................................................................................15
      3.2.5. Recording a meaningful learning task .............................................................................15
   3.3. WEKIT.ONE CLOUD .............................................................................................................17
      3.3.1. User authentication ..........................................................................................................17
      3.3.2. Storage REST API .........................................................................................................17
4. CONCLUSION ................................................................................................................................20
REFERENCES ......................................................................................................................................21
   ARTICLES .....................................................................................................................................21
   RELATED WEKIT DELIVERABLES ..............................................................................................21
   OTHER REFERENCES ....................................................................................................................22
Executive summary

This deliverable reports on the updated version of the software prototype with sensor fusion API and usage description. It describes all enhancements, improvements and new developments following the first completed cycle of specification, development, and evaluation. This comprises:

- A report on updates on architecture and developments based on feedback gathered after the first iteration of developments and trials (user feedback, technical feedback, issues experienced, design decisions taken)
- An overview over the core components of the WEKIT system and how the WEKIT.One Hub is integrated within
- Details on how the WEKIT.One Recorder utilises the WEKIT.One Hub in its creating, editing, recording, storing of learning experiences
- Examples on how the high-level ARLEM standard format is complemented with a sensor data format for high-frequency sensor data.

The deliverable does not repeat the content of its predecessor (D3.3), but instead highlights the changes and additions. Consequently, it describes the current version of the relevant units such as the WEKIT.One Recorder, WEKIT.one Hub and WEKIT.One Cloud. Supporting informations for usage and further extension of the current versions are also added for each units. Concise information has been provided with examples and infographics where possible for all relevant stakeholders to understand the aim of the deliverable.
1. Introduction

As outlined in the WEKIT project description, the objective of this deliverable (D3.6) is to describe the updated versions of the software prototype and the sensor fusion API. It involves reviewing existing deliverables and transform their outcomes into formal specifications:

- D1.5 WEKIT Framework and Training Methodology
- D1.6 Requirements for Scenarios and Technological platform
- D3.4 Requirement Analysis and Sensor Specifications

The rest of this deliverable is organised as follows. In Section 2, we outline the methodology used. In Section 3, we describe the Software prototype comprising the WEKIT.One Recorder, WEKIT.One Hub, and WEKIT.One Cloud.

2. Methodology

Based on the first instantiation of the WEKIT Framework (D1.3), the first iteration of sensor specification (D3.1), hardware and software development (D3.2 & D3.3) and their integration in the WEKIT.One prototype (D2.4), the WEKIT project performed the first round of pilot evaluations with our three pilot partners (D6.4, D6.5, D6.6). The feedback gathered within these trials led the second iteration of framework (D1.5), requirements (D3.4), and developments (D3.6).

The following new developments, changes, and conceptual enhancements result from the feedback gathered:

- The high level architecture has been adapted to better utilise network connectivity and computing power.
- The WEKIT.One Recorder has been redesigned to reflect usability and functionality feedback gathered (Limbu et al., 2018b). It now follows principles of design patterns for augmented reality (Antonaci, Klemke & Specht, 2015; Emmerich, Klemke & Hummes, 2017).
- The WEKIT.One Recorder also reflects updates in the WEKIT Framework (Limbu et al., 2018a; Guest et al., 2017), leading to new and updated transfer mechanisms.
- The WEKIT.One Hub has been redesigned to reflect experience gathered in sensor communication, scalability and flexibility. This also comprises of changes made to the sensor data storage and communication formats. Furthermore, the updated version of the sensor specification (D3.4) covers a broader range of sensors and a change in the underlying hardware architecture.
- The WEKIT.One Cloud has been designed to cover core use cases of collecting and sharing captured experiences, metadata, and to allow for basic editing functionalities.

3. Software prototype Description

Fig. 1 shows the updated high-level architecture of the WEKIT solution and its connections to the key components of the overall WEKIT system (Hololens, Sensor Processing Unit and Cloud Server). The new version puts more emphasis to utilize the computing power of the sensor processing unit in terms of local data storage, connectivity for communication. For this reason, the connection to the backend repository (WEKIT.One Cloud) is moved from Hololens to the Sensor Processing Unit (SPU), where the Controller component (part of WEKIT.One Hub) handles the connection to the backend.
Furthermore, the WEKIT.One Hub is now also responsible for controlling all connected sensors, storing the captured sensor data locally, and joining the data of different sensors into a common storage format. To minimise communication overheads between Hololens and SPU, data captured utilising native Hololens sensors is stored locally on Hololens. Only, when the complete ARLEM fileset is compiled (upon user request), the different sensor data files, annotations, user entered data, and media files (audio, video, photo) are send to the SPU and processed by the WEKIT.One Hub for upload to the WEKIT.One Cloud.

**Figure 1.** Updated Architecture of the WEKIT solution and its connections to Hololens, Sensor processing Unit and Cloud Server

The WEKIT.One frontend prototype is developed with Unity3D, Vuforia (marker-based image recognition toolkit for augmented reality), and the Microsoft MixedReality Toolkit. The prototype is organised in two main sections: the Learner GUI and the Expert GUI. This deliverable reports on the Expert GUI and its underlying recorder component.

The Expert GUI contains two main functionalities: virtual annotation of objects in the physical space with various annotation types (text, image, video, audio, 3D objects) and capturing of expert activities using multimodal sensor recordings to re-enact expert behavior for the learner.

The recorder allows for two different ways of connecting virtual annotations to the physical space: marker-based and anchor-based. The marker-based approach uses specific marker images, which are bound to the environment and which can be recognised by the camera, to locate all annotations relative to the marker image. The anchor-based approach uses an environmental world model to locate all annotations relative to world anchors. While marker-based annotations require additional markers to be available in the environment, these can also easily be re-used in different locations. However, in cases, where additional markers are not acceptable (e.g. in the narrow space of an aircraft cockpit), the anchor-based approach allows for marker-independent annotations. Fig. 2 shows the core object model implemented to realise these functionalities.
The outcomes of both functionalities, annotating and capturing, are transformed into ARLEM compliant learning experiences, which can be stored locally and in the cloud-based backend storage. The ARLEM standard defines the basic format for creating complex learning experiences using basic building blocks. It is also the exchange format that can be read by the learner component to allow learners to experience the captured expert performance from their perspective.

The following section contains a usage description of the WEKIT.One Recorder tool, followed by a description of the WEKIT.One Hub and the WEKIT.One Cloud solution.

3.1. WEKIT.One Recorder

WEKIT.One Recorder is designed to allow trainers to record their demonstration. In addition to recording sensor data, the recorder allows experts to create “task stations”. Task stations are object of interest in the physical world. Task stations can be annotated different forms of annotation. The recorder currently offers a range of annotations such as text, image, audio and combined sensor readings. The current version also includes video annotations. A task station may contain more than one annotations. Similarly, a recording session contains many task stations which are collectively uploaded to the cloud as a session. Fig. 3 shows, how the the WEKIT.One Recorder object model is instantiated in an editing session.

Figure 2. WEKIT.One Recorder object model
The current version of the recorder is improved based on the feedback and experience collected in the first trails. These improvements also address number of bugs noticed in the trails. The user interface has been improved with contrasting colors for readability and consistent color profile has been applied. Audio based feedback on taps and click provides much better assurance to the user of their actions. In addition audio based assistance are provided as instructions to the trainers in order to flatten the learning curve for the trainers. Visual hints that allow experts to keep track of their progress such as the line connecting the task stations in step - 7 have also been implemented.

Steps for creating a task station with the recorder is described below.

1. Select "Recorder" in the start up screen.

**Figure 3. WEKIT.One Recorder sequence diagram for creating a learning experience**

**Figure 4. Start up screen**
2. Make an "Air tap" gesture to open a pop up menu which allows creation of task station.

![Figure 5. Create task station](image)

3. A Sphere will be placed in the physical environment which represents a task station. Use the cursor on the Hololens to aim at the task station. When an object is hovered by the cursor it will be highlighted with yellow color and the cursor will change into a hand, indicating possibility to tap that object.

![Figure 6. Task station selection](image)

4. When tapped the task station will show a list of possible annotations.
When an annotation is selected, it will be connected to the task station by a thin line indicating that the annotation belongs to that task station. Annotation icons can be tapped to open menus underneath it.

By selecting the corresponding menu options, an annotation can be placed.
7. Once more than 2 task stations are created, the task stations are linked by a thin line in the order they were created to highlight the relevance to the steps in the procedure.

Figure 9. Taking photo with image annotation

8. When the demonstration is complete and all task stations are created, the annotations can be saved for expert. Click on “Save to ARLEM” option in the initial menu (see step-2).

Figure 10. Sequence of task station

3.2. WEKIT.One Hub

The objective of the WEKIT.One Hub (Hub) is to collect data from different sources and generate a unified multimodal digital experience of a learning task. The Hub has two main functions that allow it to contribute to the study and enhancement of the learning process. First it creates multimodal recordings of learning tasks. Second the Hub retrieves critical data from the different sources, and forwards this data to applications designed to provide immediate feedback to learners.
The multimodal data used to capture meaningful learning tasks comes from different sensor-applications. Each of these applications uses different sensors in order to capture different aspects of the learning task. The different sensors measure different properties and operate at different time frequencies; thus each sensor-application generates different types of data and updates it at different frequencies. For instance, Grove ear-clip based sensor is used to measure heart rate, MPU 9250 IMUs are used for estimating the posture of the person, SHT1x sensor measures temperature and humidity in the environment, and two vibration motors are used to provide haptic feedback to the user. The main purpose of the WEKIT.One Hub is to synchronise and fuse the different streams of multimodal data generated by the different sensor-applications while capturing a meaningful learning task.

3.2.1. Multimodal recording of a learning task

It is neither feasible nor desirable to have applications retrieving, recording and analyzing data from learners all the time. The Hub works by recording only specific learning tasks, this means that the user of the Hub needs to manually start and stop a recording.

Communication between Hub and data providers

To start, stop and generate a unified multimodal recording the Hub needs to communicate with the providers of data. The first step into creating a recording is to define the communication channels between the data providers and the Hub. For each source provider one needs to define:

- Name of the provider.
- Path where the provider will be executed (file path in case the provider will run on the same computer or IP address in case the provider will run remotely)
- Remote or local execution of the provider
- Port number for the TCP listener socket that will receive instructions from the provider.
- Port number of the TCP sender socket that will send instructions to the provider.
- Port number for the TCP socket that will receive files from the provider.
- Port number for the UDP socket that will receive critical real-time data from the provider.
- Port number for the UDP socket that will send streams of data to the provider.
- Boolean value stating if the provider will be used for the specific recording.

From the side of the providers the communication can be handled through the ConnectorHub dynamic library that is included with the Hub solution, or by manually programing the socket communication. Currently Hub solution contains a dynamic Library that works with .Net projects and a dynamic library that works with windows universal platform.

In order to be able to communicate between each other the Hub and each provider must agree on the same communication channels. This is handled with a configuration file named portConfig.txt. Each of the first five lines of this file contains the port numbers that were previously defined. Finally the sixth line of the file contains the IP address of the Hub. In the case of providers running in the same computer as the Hub, the Hub automatically creates this configuration file in the right location. In the case of providers running in other computers this configuration file needs be created manually.

After configuring the communication channels one need to run the provider applications. Provider applications running on the same computer as the Hub will be started automatically. At this point the Hub will internally create a handle for each one of the providers in order to communicate with them. Once a provider is ready to start capturing data, it will send an “IamReady” signal to the Hub. At this point through the Hub it is possible to start and stop a Multimodal recording of a learning task.
Sensors connected with the ESP32 microcontroller send the data via USB connection to the SPU in the form of Serial frame data.

The WEKIT.One Hub can also actively communicate with Sensors to control their functionality (e.g. switch on/off or enable feedback channels such as LEDs or vibration motors). See section 3.2.4 Immediate Feedback.

Example on how to use the ConnectorHub Library:

Add the following lines to initialize the connector:

```c
myConnectorHub = new ConnectorHub.ConnectorHub();
myConnectorHub.init();
myConnectorHub.sendReady();
myConnectorHub.startRecordingEvent += MyConectorHub_startRecordingEvent;
myConnectorHub.stopRecordingEvent += MyConectorHub_stopRecordingEvent;
setValuesNames();
```

Then you need to add a list with the attribute names:

```c
myConectorHub.setValuesName(names);
```

3.2.2. Data Storing

As discussed previously each data provider retrieves different type of data and at a different rate. In order to fuse the data coming from different providers in one unified multimodal recording we used the following Recording format:

A multimodal recording is composed by a collection of RecordingObjects. Each provider generates one RecordingObject. A RecordingObject is composed by a recordingId, an applicationName (name of the data provider) and a collection of FrameObjects. Each FrameObject consist of a frameStamp (time passed since the beginning of the recording) and a dictionary containing the name of the attributes stored for each frame and the current values of these attributes. Once the recording stops, all RecordingObjects are collected by the Hub.

```json
{
    "recordingID": "10H5M43S",
    "applicationName": "ESP32application",
    "frames": [{
        "frameStamp": "00:00:05.8550000",
        "frameAttributes": {
            "AcceloremeterX": "0.325943395020413",
            "GyroscopeX": "0.03833008"
        }
    },
    {
        "frameStamp": "00:00:05.8550000",
        "frameAttributes": {
            "AcceloremeterX": "0.325943395020413",
            "GyroscopeX": "0.03833008"
        }
    }]
}
```
"frameAttributes": {{
  "AccelometerX": "0.325943395020413",
  "GyroscopeX": "0.03833008"
}}
}
}

Example code of a JSON serialization of the Sensor

### 3.2.3. Multimodal data synchronization

The first instruction executed by data providers after receiving a StartRecording instruction is to take note of their current time, which is stored as the starting time of the recording. During the recording once a data provider has a frame ready to be stored, it checks for the current time and subtracts from it the starting time. By assuming the clocks from the data providers run at the same speed, and that all data providers received the StartRecording instruction almost at the same time, this strategy allows a good enough synchronization of multimodal data.

### 3.2.4. Immediate Feedback

To keep things simple from the side of the tutor, it is recommended to only transmit critical data from the providers to the Hub. An example of critical data could be the instruction to "Speak Louder" in case a microphone application detects that the learner is speaking too soft during the specific learning task. To keep things as simple as possible this type of instruction is transmitted from the provider to the Hub via UDP sockets.

The Hub can then forward the received instructions to applications design to provide feedback to learners. This applications can be ambient displays, augmented reality glasses, etc. Establishing the communication between the Hub and the immediate feedback applications is a very similar process to the one for establishing the link between the Hub and providers. Before starting a recording one needs to manually select the feedback applications that will be used and define the communication channels. For each feedback application one needs to define:

- The Name of the application
- Path to reach the application (IP address)
- Port number for sending operational instructions via a TCP socket
- Port number for streaming feedback instructions via a UDP socket.

The feedback applications need to open the socket communications in order to receive the information that comes from the Hub. This can be done through the use of the dynamic Libraries included in the Hub solution (the currently supported libraries are for the .Net framework and Windows Universal Platform), or by manually programing the TCP and UDP sockets.

### 3.2.5. Recording a meaningful learning task

For capturing meaningful learning tasks we use the concept of user invoked Action Recordings: in this case the learner invokes the recording of action, declares the meaning of it, and executes it. The Action Recording starts when the WEKIT.One Hub receives the command "Start Recording". This command includes the following parameters: Actor, Verb, Object and List of sensor-applications used for the recording. The recording stops once the "Stop Recording" command is received. Both commands can be executed locally and remotely: locally by manually selecting the sensors and clicking the "Start" button, remotely by sending a TCP message to the WEKIT.One Hub.
Figure 11. WEKIT.One Hub architecture

Figure 12. Proposed WEKIT prototype. Smart glasses are worn on the head and IMUs are placed on the back of the user, heart rate variability can be measured by using an ear clip based sensor (such as by Grove), Myo armband is placed on an arm, vibration motors can be placed on both arms, and finally power bank, SPU, and ESP32 micro-controller for external sensors can be placed in the front.
3.3. WEKIT.One Cloud

The third software component of the WEKIT.One software prototype is the backend infrastructure: WEKIT.One Cloud, that works as Repository of Learning Experiences. The Cloud Repository is the place and retrieved the stored sessions.

The Cloud Repository works in combination with the Recorder and the Hub. It is implemented using a Java RESTful web service using servlets running on a Google App Engine instance (standard environment). It uses Objectify and GSon for a scalable datastore for storing managements of the classes.

3.3.1. User authentication

To implement secure access OAuth authentication provided by the WEKIT Community platform and the Learning Layer OAuth server. For accessing the Repository, a user needs to sign up to the WEKIT community platform. When accessing the Repository home page https://wekitproject.appspot.com/, the user is invited to sign in using the log in.

3.3.2. Storage REST API

The REST API is designed to connect the WEKIT application with the storage backend. It allows to communicate recording sessions between application and cloud storage. A session comprises all data recorded, annotated, edited. The following subsections details the implemented functions for the /storage REST API of WEKIT.One Cloud, which comprise uploading, listing, editing, and downloading sessions.

Uploading a session

A session can be uploaded by an authorised client in two steps.

Step 1) request to Google Blobstore a temporary token for the /storage.

URL: /storage/requestupload
Method: POST/form-data
URL Params:
- Success response:
  Code: 200
  Content: <the_upload_URL>

Step 2) Upload the session compressed folder (e.g. in Zip file) to the Google Blobstore using <the_upload_URL> generated in step 1:

URL: <the_upload_URL>
Method: POST/form-data
URL Params:
- Required:  
  myFile = [file]
- Optional:  
  device = [string]
  author = [string]
  description = [string]
- Success response:
Generating list of sessions

The list of session can be retrieved the following method. A single session can be retrieved with

**URL:** /storage/sessions
**Method:** GET
**URL Params:**
- Optional: id = [integer]
**Success response:**
Code: 200
Content: <JSON_list>

Downloading a session

To download a session the Blob key is necessary. That can be found in each item of the `<JSON_list>` with the attribute "key". The Blob key has the following format /
gs/wekitproject.appspot.com/long_Alphanumeric_Token.

**URL:** /storage/serve/<JSON_list[n].key>
**Method:** GET
**Success response:**
Code: 200
Content: <JSON_list>

List uploaded sessions

Besides the JSON format is also possible to list the Session in rich HTML format. As displayed in figure XX, a Javascript DataTable renders the JSON information easy format. With the help of JQuery and Javascript is also possible to sort each attribute of the table by clicking on the header, search for a specific term and navigate with the buttons on the bottom-right.

**URL:** /storage/list/
**Method:** GET
Figure 13. Screenshot of the Cloud Repository session list.

Edit a Session

With the table proposed is also possible to perform Addition, Editing and Deletion with the buttons on the top-left of the table. In figure XX it is shown the editing form for a particular session.

Figure 14. Screenshot for editing a session.
4. Conclusion

This deliverable presents the final iteration of the sensor fusion API, implemented in form of the WEKIT.One Hub as part of the overall WEKIT.One prototype. The document shows, how the WEKIT.One Hub bridges the WEKIT.One Recorder and the sensor processing unit, which is the hardware component complementing the WEKIT.One Hub. Furthermore, the deliverable also describes the WEKIT.One Cloud solution, which represents the backend storage for captured data.

This deliverable thus fulfilled the following needs as stated in the executive summary:

- It reports on updates on architecture and developments based on feedback gathered after the first iteration of developments and trials (user feedback, technical feedback, issues experienced, design decisions taken). We have shown architectural changes, user interface updates, new functionalities to comply to this.
- It gives an overview over the core components of the WEKIT system and how the WEKIT.One Hub is integrated within. The updated, detailed architecture (Fig.
- It shows, how the WEKIT.One Recorder utilises the WEKIT.One Hub in its creating, editing, recording, storing of learning experiences
- It details, how the high-level ARLEM standard format is complemented with a sensor data format for high-frequency sensor data.
References

Articles


Related WEKIT Deliverables

D1.5 WEKIT Framework and Training Methodology (M20)

D1.6 Requirements for Scenarios and Prototypes (M23)

D3.2 Hardware Prototype with Component Specification and Usage Description (M12)

D3.4 Requirement analysis and sensor specifications - Final version (M25)
Other references

Alex (2016). Alex posture tracker product page: http://alexposture.com/


Leap (2016). Leap Motion product page: https://www.leapmotion.com/


WEKIT project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 687669. http://wekit.eu/