



Project reference: 2015-1-LU01-KA202-001353

Project acronym: BIM4VET

Project title: Standardized Vocational Education and Training for BIM in EU

KA2- Cooperation for Innovation and the exchange of good practices strategic partnerships for vocation education and training

IO5. TANGIBLE TABLETOP DEVICE SUPPORTING THE BIM QUALIFICATION ASSESSMENT AND THE TRAINING SELECTION

Dissemination level	Public
Activity	5 – Tangible tabletop device supporting the BIM qualification assessment and the training selection
WP Leader	LIST
Contributor(s)	CEA, Cardiff University
Author(s)	Nico Mack
Status (F: final, D: draft)	D

Disclaimer

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

The document reflects only the authors' view and the European Commission, and the Anefore are not responsible for any use that may be made of the information it contains.

Copyright

© Copyright 2017 BIM4VET Consortium

This document may not be copied, reproduced, or modified in whole or in part for any purpose without written permission from the BIM4VET Consortium. In addition to such written permission to copy, reproduce, or modify this document in whole or part, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.

All rights reserved.



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/)

History

Version	Description	Lead author	Date
1.0	Initial Draft of the document	Nico MACK (LIST)	27.03.2018
1.1	New version	Annie Guerriero (LIST)	18.04.2018
1.2	Updated content	Nico MACK (LIST)	23.04.2018
1.3	Final version	Annie Guerriero (LIST)	23.04.2018

Diffusion

Version	Sent to	Date
1.0	All consortium partners	27.3.2018
1.1	All consortium partners	18.04.2018
1.2	All consortium partners	23.04.2018
1.3	All consortium partners	23.04.2018

Acknowledgements

BIM4VET has received funding from the Erasmus+ programme (*Key Action: Cooperation for innovation and the exchange of good practices, Action Type: Strategic Partnerships for vocational education and training*).

The partners of the project are LIST, Cardiff University, and Commissariat à l'Energie Atomique et aux Energies Alternatives.

Acronym

API	Application Programming Interface
BIM	Building Information Modelling
DIY	Do It Yourself
GIS	Geographical Information System
GIT	A distributed version control system
GUI	Graphical User Interface
HDCE	Human Dynamics in Cognitive Environments
IR	Infrared (Light with a wavelength between 780 nm and 1450 nm)
ITIS	Information Technologies for Innovative Services
JDBC	Java Database Connectivity
LGPL	GNU Lesser General Public Licence
MQTT	Message Queuing Telemetry Transport
REST	Representational State Transfer
TUI	Tangible User Interface
TUIO	An open framework that defines a common protocol and API for tangible multi-touch surface
TULIP	A framework for tangible user interfaces developed by ITIS/HDCE Research Unit of LIST.
XML	Extensible Markup Language

Table of contents

1. TANGIBLE USER INTERFACE	5
1.1. INTRODUCTION	5
1.2. TANGIBLE TABLES	5
1.2.1. THE HARDWARE	6
1.3. BACKGROUND	6
1.3.1. TULIP FRAMEWORK	6
1.3.2. TULIP SOFTWARE ARCHITECTURE	7
1.3.2.1. Tulip-CPS	8
1.3.2.2. Tulip-GIS	8
1.3.2.3. Tulip-IOT	8
1.4. FOREGROUND	8
1.4.1. MULTIPLE CONTEXTS	8
1.4.1.1. Implementation	9
1.4.2. DISTRIBUTED SYSTEM ARCHITECTURE	9
1.4.2.1. Implementation	10
1.4.3. PROFILEWIDGET	11
1.4.3.1. ModalWidget	11
1.4.3.2. Profile Modes	11
1.4.4. VISUALISATION OF RECOMMENDATIONS	14
1.4.4.1. Force Directed Graphs	14
1.4.4.2. Implementation	16
1.5. CONCLUSION	16

Table of figures

Figure 1 - Tangible Table-top Device	5
Figure 2 - Examples of Markers (left Multitaction – right Reactivision)	6
Figure 3 - Mapping between Physical and Virtual Objects	7
Figure 4 - Tulip Architecture	7
Figure 5 - Multiple Contexts	9
Figure 6 - Bim4Vet System Architecture	10
Figure 7 - ProfileMode Class Diagram	12
Figure 8 - Visual representation of ProfileWidget.....	12
Figure 11 - Training Subscription	14
Figure 12 - Force Directed Graph	15
Figure 13 - Recommended Trainings with Fitness Scores per Participant	16

1. Tangible User Interface

1.1. Introduction

The field of human-computer interactions relies heavily on the use of metaphors. As an example, in the field of Graphical User Interfaces (GUI), the desktop metaphor in combination with the folder and file metaphor of the filesystem mimics a physical office environment. Graphical symbols or Icons represent known objects from that environment thus implying the functionality of the given user interface element. The use of efficient metaphors renders a user interface easy to manipulate and reduces the time and effort required by a user to get accustomed to using the interface¹.

A decade ago, GUIs mainly relied on point-and-click devices (mice, touchpad) for interaction and keyboards for data capture. With the advent of touch-based devices like smart phones and tablets, those physical devices have slowly become obsolete or have been replaced by virtual counterparts like onscreen keyboards. The introduction of multi-touch interfaces, i.e. touch interfaces which were able to detect more than one finger simultaneously, enabled more natural interactions by allowing gestures such as pinching, swiping and rotating, gestures which have already become second nature to users, even to the least tech-savvy ones.

Tangible User Interfaces (TUI) are an attempt to further lower the accessibility and acceptance hurdle of computer systems, leveraging the fact that human beings are used to manipulate physical objects. Where GUIs rely solely on the users' visual capacities, TUIs are able to convey information on a haptic and topological level as well, mixing both physical and digital elements to interact with the interface. GUIs in general are very structured interfaces, meaning that control and visualisation elements are *rigidly* laid out, following a strict organisation pattern in order for the user to easily find them once he's accustomed with the interface. TUIs on the other hand break this paradigm by giving designers the freedom to *de-structure* the interface either completely or partially, thus requiring a more efficient set of metaphors to imply functionality and guarantee usability. A dedicated work-package, *IO6 – Assessment Report on Experimentations*, comprises all the tasks and outputs required for validating and verifying the design choices and assumptions being made in terms of usability and user acceptance.

1.2. Tangible Tables

TUI applications required a specific input output device called a **tangible table-top device** or simply a **tangible table**. Figure 1 shows the working principal of such a device.

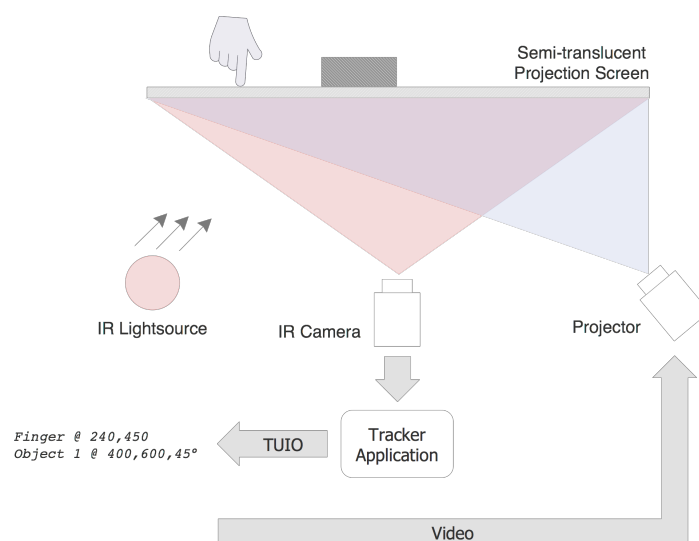


Figure 1 - Tangible Table-top Device

¹ E.Tobias, V.Maquil, T.Latour, 'Method for providing data input using a tangible user interface', PT03427EP

All components depicted below the **semi-translucent projection screen** are installed inside an opaque enclosure, the projection screen being the top face of the enclosure. An **Infrared light source** floods the enclosure with infrared light. **Objects** and **fingers** placed on the semi-translucent projection screen reflect some of the light. A **video camera** with an infrared filter matching the wavelength of the emitting light source, captures those reflections and feeds the captured frames to a **tracker application**. The tracker application processes the received frames and extracts the x and y positions of the captured reflections. For the tracker to be able to identify individual objects and their respective orientation on the table, it is important to tag each individual object with a visual black and white marker.

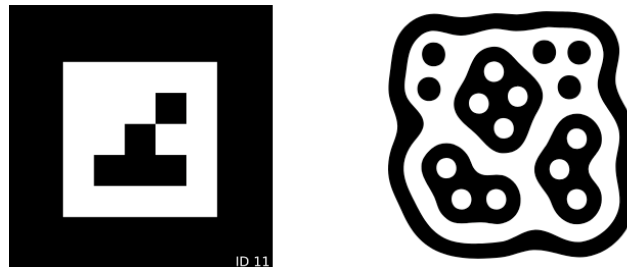


Figure 2 - Examples of Markers (left Multitaction – right Reactivision)

The type of marker to affix is governed by the choice of the tracker application. Figure 2 shows two markers representing both the identifier number 11, the one on the left is for the Multitaction proprietary tracker and the one on the right for the ReactiVision open-source tracker. The information extracted by both tracker applications is transmitted as events using a de-facto standard protocol called TUIO.

A connected computer reads the incoming TUIO events, interprets them and acts upon them. In most cases the resulting action consists in drawing information and data at the respective object positions. The video output of the connected computer is fed to a video projector build into the enclosure, projecting the drawn items on the semi-translucent projection screen, reconciling both the physical object and its virtual representation.

1.2.1. The Hardware

The choice of the hardware being used is mostly influenced by the cost factor of the respective solution, but also by criteria such as transportability, ruggedness and reliability. The cheapest solution in terms of costs is the DIY approach. We, that is the ITIS/HDCE unit of LIST, have built a number of those tables ourselves and helped schools to build their own tables. The price tag of such a table is below 5k€, depending on the chosen components. On the downside, those tables are not easy to transport and require a time-consuming calibration process after moving them in order to realign the camera and the projector. If transportability and ruggedness are key, then a commercial table such as the MT-Cells (<https://www.multitaction.com/hardware>) from the finish manufacturer Multitaction is a possible candidate. They come in the shape of a 1215 x 686 x 200mm Full-HD Display, integrating all the required hardware and software (Tracker Application) in one single unit. They come with a rugged flight-case for easy and safe shipping. All this comes with a price of course, which is approximately 20k€ per unit.

1.3. Background

1.3.1. Tulip Framework

The TUI developed in the scope of work package IO5 by the ITIS/HDCE unit (LIST, LU), is based on our existing TULIP framework, a JAVA software framework for developing tangible applications, progressively developed in house and extended since its gestation in 2014. In order to establish TULIP as a de-facto standard for TUI applications, the framework is available under an LGPL version 3 open-source licence on our GIT server (<https://git.list.lu/nui/tulip/wikis/home>).

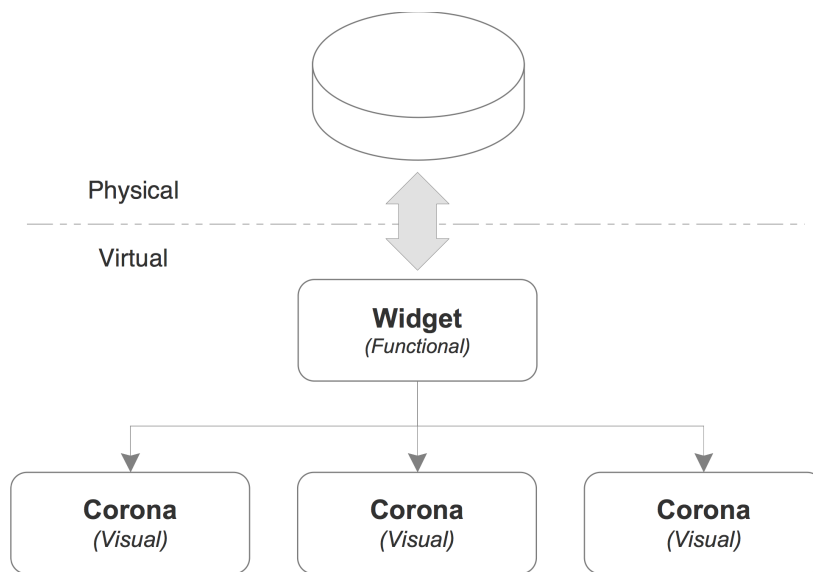


Figure 3 - Mapping between Physical and Virtual Objects

The TULIP framework is a collection of customizable building blocks, mapping the physical objects on the table to a virtual representation on the screen. Each physical object, identified by its marker, is mapped to a dedicated software representation, called a **Widget**. Widgets define the **functional** behaviour of the object. Each widget has one or more associated **Coronas**, defining the **visual** representation of the object on the table. The mapping of markers to widgets and subsequently of coronas to widgets is defined in an external XML file called a **Scenario**. The scenario file encapsulates all design related aspects of the application to build, i.e. graphical and functional attributes of coronas, configuration of data sources, chaining of data processing steps.

1.3.2. Tulip Software Architecture

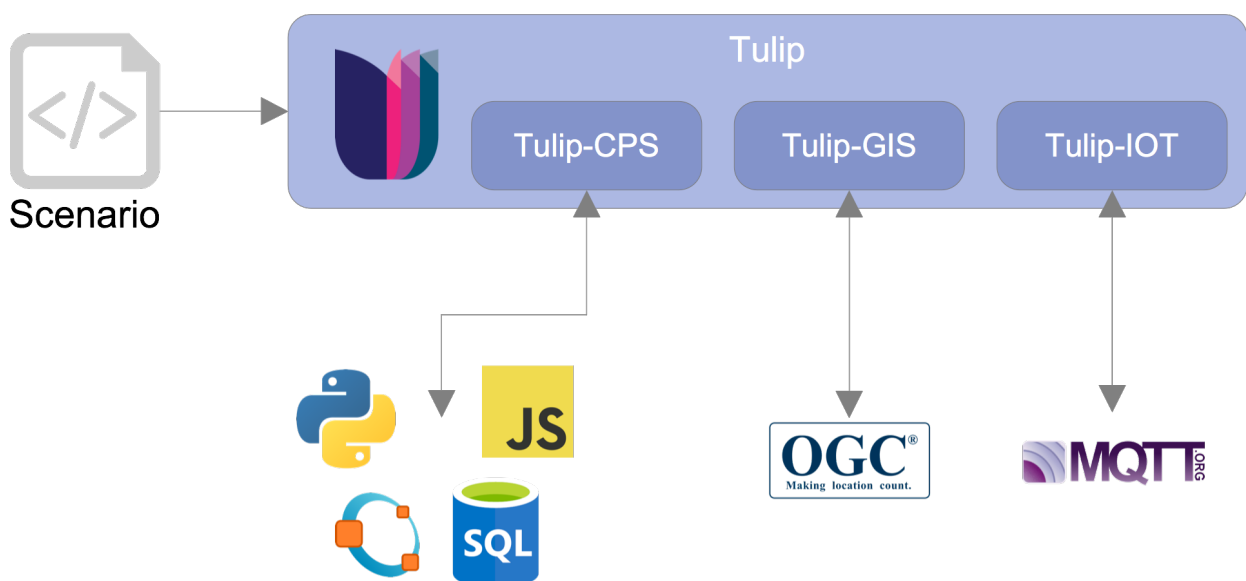


Figure 4 - Tulip Architecture

From a software architecture point of view, the TULIP framework consists of the open-source Tulip core package and a number of domain specific extensions. Each extension not only adds domain specific widgets and coronas, but also adds connectivity to external systems and environments.

1.3.2.1. Tulip-CPS

CPS stands for Complex Problem Solving. The extension adds the concept of **variables** and provides widgets and coronas to manipulate and visualise those variables. On top of that, Tulip-CPS adds the concept of **equation systems** to the scenario, mapping variables either as parameters or results to individual equations. Tulip-CPS provides a number of specific **Executors** for those equations, enabling the author to either externalise the processing body of the equation in third party languages like Python, JavaScript or Octave, or to directly query SQL databases.

1.3.2.2. Tulip-GIS

Tulip-GIS adds geographical information to Tulip. The extension adds interfaces for connecting with OGC (Open Geospatial Consortium) compliant GIS systems and brings along widgets and coronas for manipulating geographical maps.

1.3.2.3. Tulip-IOT

Tulip-IOT is the latest addition to the Tulip family, adding IOT (Internet-Of-Things) connectivity to Tulip. This is achieved by adding MQTT messaging, a lightweight messaging protocol for small sensors and mobile devices, to the scenario. By adding the concept of **messaging systems** to the scenario, the author has the possibility, in combination with the Tulip-CPS extension, to retrieve sensor values and to control actuators.

1.4. Foreground

Most of the work done in the scope of the BIM4VET project was focused on the design and the implementation of the scenario, leveraging already existing building blocks and functionality to do so. In an ideal world, with a feature-complete framework, no actual coding would be required. The complexity of the BIM4VET scenario however entailed a minimum amount of development work, mostly concentrated on the implementation of a dedicated `ProfileWidget`. Before going into details, I would like to introduce some of the design challenges we faced during the implementation of the BIM4VET scenario.

1.4.1. Multiple Contexts

The equation system introduced by the Tulip-CPS extension allows the definition of a set of independent (parameters) and of dependent (results) variables. This set of variables and their respective values are called a context. The scenarios implemented before the BIM4VET project were exclusively single context applications, i.e. one set of variables was sufficient to describe the overall state of the application. The requirement that the BIM4VET scenario shall allow each participant to have its own set of competence objectives, filter criteria and training recommendations, very quickly revealed that the BIM4VET scenario would require on one hand the ability to handle and distinguish multiple individual contexts in parallel, and on the other be able to merge variables from different contexts into a combined view.

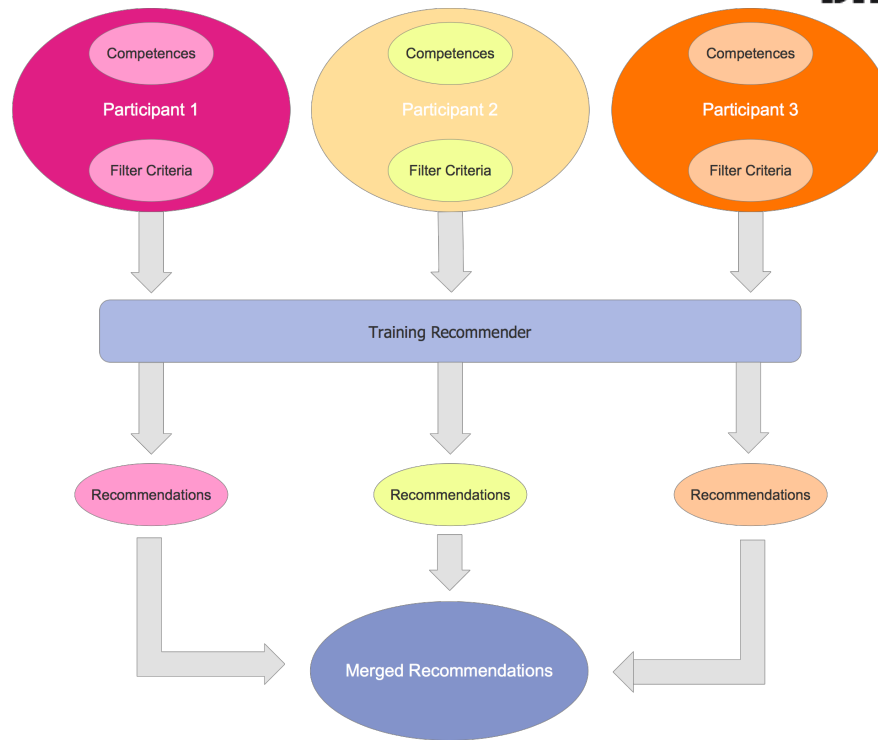


Figure 5 - Multiple Contexts

1.4.1.1. Implementation

Managing multiple contexts is very similar to encapsulating attributes in object-oriented programming. Looking at Figure 5, it's easy to see that each participant can be seen as an instance of a virtual Participant class and that their respective competences and filter criteria are merely private class members. Once this similarity was understood, the next logical step was to figure out how non object-oriented programming languages added object-oriented features. The **Perl** programming language does this in a very simple and transparent way, an object instance is a special kind of reference to a **hash table**, the class members being the keys into the hash and the respective instance variables the corresponding values.

The Tulip-CPS extension already supported a range of scalar and array variables, including hash variables. By using those hash variables, we were able to model each participant as an instance of a virtual class.

1.4.2. Distributed System Architecture

The architecture of the BIM4VET system is both distributed and heterogeneous. This design choice was motivated in equal parts by technical and organisational constraints. In fact, the three main components making up the BIM4VET system, i.e. the **Database**, the **Recommender** engine and the **tangible tabletop device** were developed individually by the three consortium partners involved in the project. With this kind of setup, it becomes clear that the database, being the projects' central repository for trainings and participant profiles, is best located in the cloud. It is also clear, that some of the components are not developed from scratch but are developed on top of or with already existing technology assets.

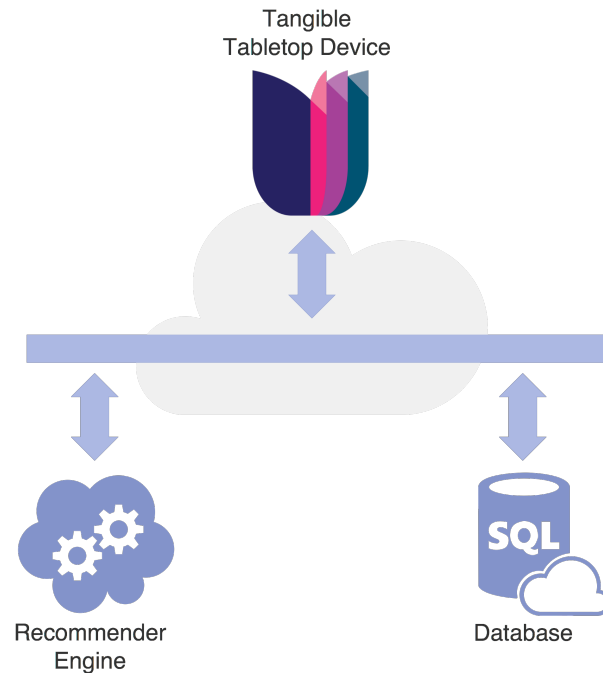


Figure 6 - BIM4VET System Architecture

1.4.2.1. Implementation

The Tulip-CPS extension already comes with a vendor agnostic database connector. As long as a JDBC driver is available for the target database, a condition satisfied for most database engines, Tulip-CPS will be able to access the database either locally or on a remote server. The cloud based (Microsoft Azure) BIM4VET database is a Microsoft SQL Server database and its technical integration was straightforward.

The Recommender engine developed by CEA-LIST was build using their proprietary *Express-If* Rule-engine, which in itself is an application developed in .NET. A direct API integration was not possible because the Tangible Tabletop Device and the Recommender were developed using different development environments (Java vs. .NET). Also, a direct API integration would have been the least desirable form of integration because of its tight coupling and the loss of flexibility this would have entailed. We choose to integrate the Recommender engine as a REST web service, thus maintaining the loose coupling required by the distributed architecture illustrated in Figure 6. The Tulip-CPS extension already comes with a REST Web service connector thus allowing a seamless integration of the recommender into the scenario. Marshalling the required parameters for the web service call and extracting data from the service answer was handled in a dedicated Python script (`Recommender.py`) integrated via Tulip-CPS.

1.4.3. ProfileWidget

Each participant shall be able to manage her or his competence profile, filter criteria and training subscription. As already mentioned in 1.4.1, Tulip-CPS already has the means to handle those profiles. What was missing however, were the means for the user to interact with the different aspects of her or his profile. Even though Tulip already contains a number of basic and high-level widgets, it became quickly clear that a custom widget implementation would be required to satisfy the needs set forth by the scenario requirements.

1.4.3.1. ModalWidget

A participants' profile can be divided into three different groups or *modes*:

- **Responsibilities** – Summarises all things related to maturity levels *already acquired* for available BIM responsibilities as well target levels to be acquired through training.
- **Filter Criteria** – Explicit criteria like cost, duration, form of delivery for narrowing down the list of recommended trainings.
- **Training Subscriptions** – Overview of all recommended trainings the participant has already subscribed to.

The widget to be developed shall be able to display different aspects of the participants' profile depending on the currently selected mode. The Tulip framework already comes with a modal widget, able to distinguish between multiple user defined modes with a dedicated set of coronas each, and thus already fulfils a part of the requirements for the `ProfileWidget` to be build. As a consequence, we decided to build the new `ProfileWidget` on top of the beforementioned `ModalWidget`.

1.4.3.2. Profile Modes

Despite the three modes representing and influencing different aspects of the participants' profile, they all have in common the fact that they share the profile of a participant. All functionality pertaining to the manipulation of participant profile data has been encapsulated in a base `ProfileMode` java class from which more specialised classes were derived for each individual mode (see Figure 7 - `ProfileMode` Class Diagram). Those specialised classes encapsulate the particular functional requirements for the respective mode. By doing so we were able to render the `ProfileWidget` agnostic about those particularities, the widget thus becoming merely a switch board, forwarding method calls to the currently selected profile mode. A dedicated `ProfileChangeListener` interface allows communicating profile changes done by the participant to the application.

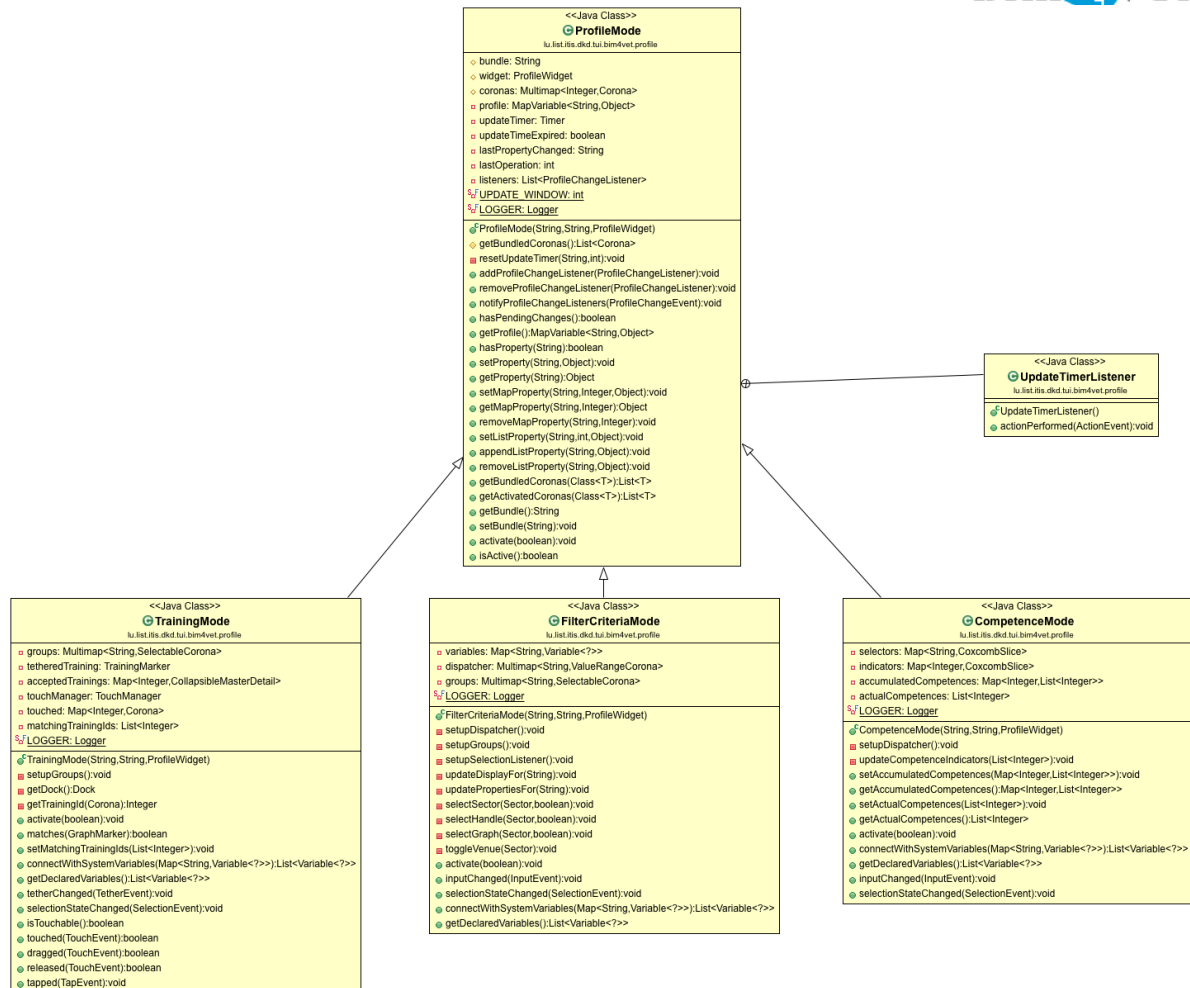


Figure 7 - ProfileMode Class Diagram

At Scenario level, the ProfileWidget is constituted, as described earlier, by the widget itself and a set of three selectable coronas of the type [Sector](#), each sector representing one profile mode. Sectors are arc shaped coronas specifically tailored for presenting multiple options arranged like petals around a circular shape. They are also part of a family of touch-sensitive or touchable coronas, i.e. coronas which can be selected by simply touching them with a finger.



Figure 8 - Visual representation of ProfileWidget

Since the ProfileWidget implements the `SelectionListener` interface, the widget is notified whenever the user touches one of the sector coronas and is thus able to activate the respective profile mode.

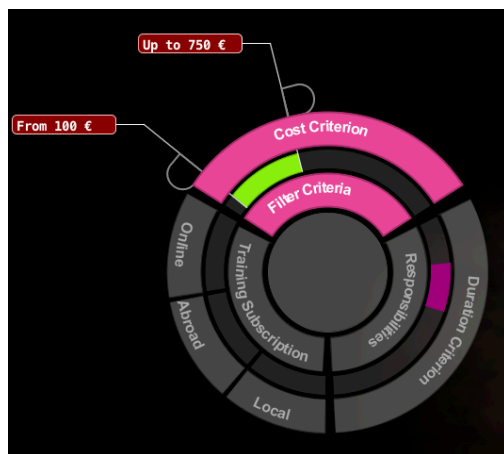


Figure 9 - Filter Criteria Mode



Figure 10 - Responsibilities Mode

Figure 10 and Figure 9 show the visual representation of the ProfileWidget depending on the currently selected mode. As already mentioned earlier, the switching of coronas being shown is handled by functionality provided by the ModalWidget that ProfileWidget inherits from.

Filter Criteria Mode

In Filter Criteria Mode, the widget shows an additional layer or ring of sector coronas, each sector representing a criterion that the participant is able to set. The current scenario distinguishes between **value range criteria**, i.e. criteria where a lower and an upper value needs to be defined, and **binary criteria**, i.e. criteria which are either on or off. The following criteria are available for the participant to set:

- **Cost** (Value Range): specifies both the minimum and the maximum amount of money an eligible training shall/may cost.
- **Duration** (Value Range): specifies both the minimum and the maximum duration in days an eligible training shall/may have.
- **Local** (Binary): specifies whether or not trainings given in the country of residence of the participant shall be included in the recommendations.
- **Abroad** (Binary): specifies whether or not trainings given in another country than the country of residence of the participant shall be included in the recommendations.
- **Online** (Binary): specifies whether or not trainings given online shall be included in the recommendations.

It is important to note that the filter criteria listed above are so called *explicit* criteria, i.e. criteria which needs to be defined by the participant. Other attributes of the participants' profile however are used as *implicit* criteria, such as country of residence, languages spoken, etc.

Responsibilities Mode

The Responsibilities mode combines two functions. On one hand it gives the participant the possibility to view her/his current maturity levels for the respective BIM Responsibilities defined in the participants' profile. Maturity levels are shown as a radial plot, each responsibility represented by a dedicated slice whose length is governed by the corresponding responsibility' maturity level.

The second function available in this mode consists in giving the participant the ability to define target maturity levels for already acquired or not yet acquired responsibilities, thus defining the overall learning outcome she or he is looking for in offered trainings.

Training Subscription Mode

Training Subscription mode also combines two functions in one mode. Its first function consists in giving the participant the possibility to subscribe to a recommended training.

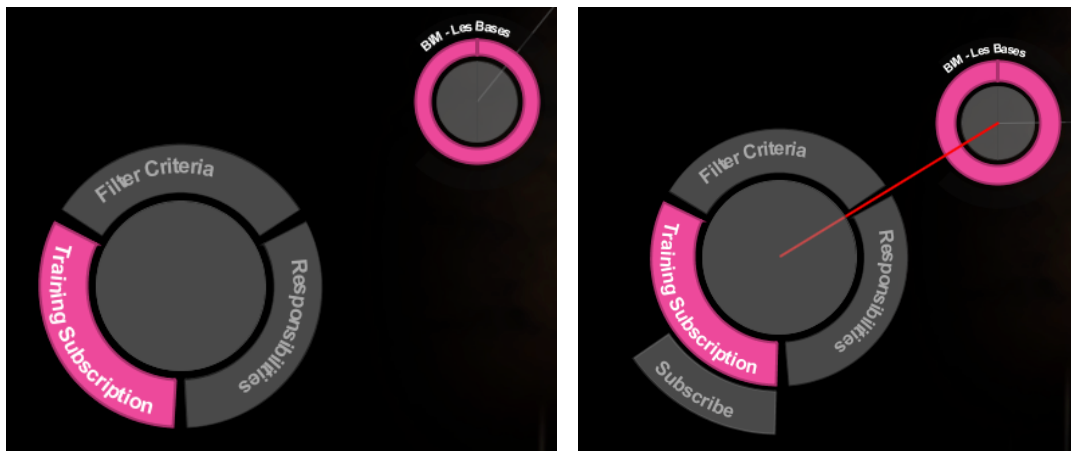


Figure 11 - Training Subscription

Figure 11 shows the subscription process. At the bottom left we see the `ProfileWidget` in Training Subscription mode. In the top right corner, we see a recommended training (The next section will provide more details about how recommended trainings are visualized). By moving the `ProfileWidget` closer to the training of interest, the participant will see at some point a red link between the widget and the training and a button labeled **Subscribe** will pop-up on the widget (right picture). The red link is part of a Tulip mechanism called *tethering* and which allows entities, known to each other as potential tethering candidates, to establish a link if the distance between both entities is inferior to a configurable distance called *tethering distance*. If the participant decides to subscribe to the *tethered* training, a touch on the Subscribe button is all it takes to do so.

1.4.4. Visualisation of Recommendations

The first versions of the Tulip framework strongly differentiated between **tangible objects**, i.e. widgets and coronas, and **content**. Content mostly consisted of images used as static backdrops, which could be switched depending on the state of the scenario. With the introduction of the Tulip-GIS extension, adding the concept of geographical maps and layers as content to the framework, content became more interactive. It was also around that time that the concept of **markers** was introduced, functional entities similar to widgets and hence using the same coronas for visualisation but with static positions bound to the content. In the Tulip-GIS extension, markers are mainly used to display information at specific geographical positions.

The visualisation of recommended trainings presented a number of interesting challenges:

1. The number of recommended trainings is variable and mainly depends on the filter criteria set by each individual participant.
2. We need to be able to display meta-data about the given recommendation, for instance the fitness score determined by the recommender engine.
3. We need to be able to convey a sense of organisational structure, i.e. being able to group trainings by hosting institution, language, etc.

The challenges mostly influencing the choice of a visualisation structure are challenges #1 and #3. It becomes clear from challenge #3 that the structure of choice will be an undirected graph of some kind. Looking for potential layout algorithms for such graphs, we decided to use a class of algorithms called *force directed*².

1.4.4.1. Force Directed Graphs

² Handbook of Graph Drawing and Visualization – Roberto Tamassia – CRC Press

In such graphs, repulsive forces between nodes and attractive forces between vertices ensure that the resulting graph is aesthetically pleasing, shows symmetry and tends towards crossing-free layouts. The main algorithm consists of a main loop applying repulsive and attractive forces until the graph reaches a local state of minimum kinetic energy, point at which the graph reaches a state of *equilibrium* and the algorithm stops.

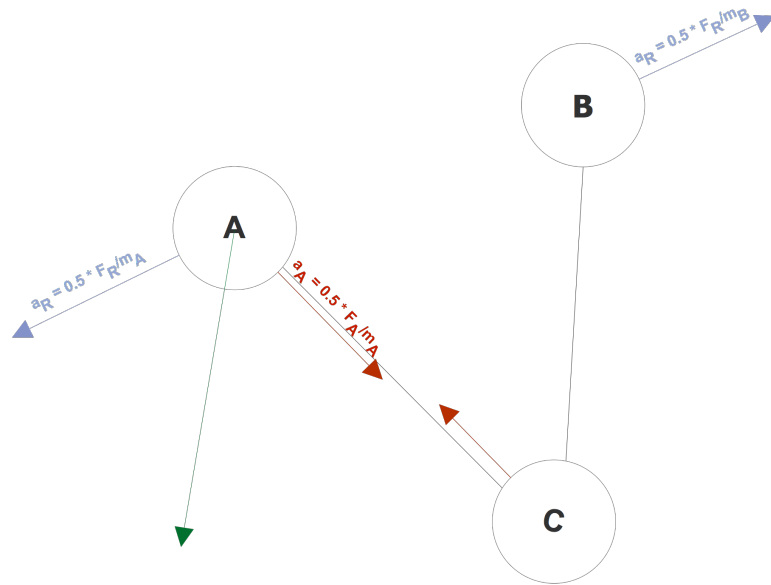


Figure 12 - Force Directed Graph

Figure 12 depicts how the different forces act upon nodes and vertices. A, B and C are *nodes*, the links between A & C and B & C are *vertices*. The repulsive force F_R between nodes A and B is determined according to *Coulombs' Law*,

$$F_R = \frac{q_A \cdot q_B}{d^2} = \frac{(m_A \cdot \rho) \cdot (m_B \cdot \rho)}{d^2}$$

Equation 1 - Coulombs' Force exerted on nodes A and B

with m_A and m_B being the mass of the respective nodes, ρ being a configurable **repulsion** constant and d being the distance between both nodes. The attractive force caused by the vertex between nodes A and C is governed by *Hooke's law*.

$$F_A = k \cdot (d - l)$$

Equation 2 - Hooke's law

with d being the distance between nodes A and C, l being a configurable **spring length** (relaxed state) and k being a configurable hooks **spring constant**. Both forces impart an acceleration to the respective nodes (second law of motion), the sum of both acceleration (green arrow) determining the magnitude and the angle of the resulting motion.

The Tulip framework already included the general concept of force interactions by defining a generic *ForceReactive* and *ForceGenerator* interface. Those concepts were added at a time when our research group was experimenting with *robotized* tangibles, i.e. tangible objects able to move on their own via build in motors and wheels. We also implemented back then the general concept of a graph, further specialised to implement a *k-d tree*, a *space-partitioning data structure for organizing points in a k-dimensional space*³, intended in our case for implementing a collision avoidance algorithm as presented in⁴. Along the same line, we started looking into force directed graphs as a mean to *slave* robotized tangibles to the position of human manipulated tangibles. Even though those ideas never came to be

³ https://en.wikipedia.org/wiki/K-d_tree

⁴ J. Snape, J. van den Berg, S. J. Guy and D. Manocha, "Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, 2009, pp. 5917-5922

implemented, the basic data structures and algorithms were already in place and could be leveraged for implementing the BIM4VET scenario.

1.4.4.2. Implementation

Looking at visualisation challenge #2, i.e. the ability to display meta-data about the given recommendation, it turns out that a Tulip Marker is a good candidate to embody this missing link. We used a dedicated type of Tulip Marker, a *GraphMarker*, whose position is governed by the accelerations imparted by repulsive and attractive forces generated by the force directed algorithm. The inherent capability of markers to display coronas allowed us on one hand to display the training title but also to display the fitness score for individual participants in the shape of a radial plot, the length of each slice being proportional to the respective fitness score, shown in the same colour as the colour of the user profile.



Figure 13 - Recommended Trainings with Fitness Scores per Participant




Since the graph of recommended trainings is a combined view of individual recommendations, this kind of visualisation allows the user to identify the trainings for him in particular, and which of those trainings are the most suitable ones.

1.5. Conclusion

The BIM4VET scenario is the most complex Tulip scenario we've build so far. As described in section 1.4.1, most scenarios so far were single context and to some extent single user. By single user we do not imply that only a single user could interact with the scenario, all scenarios can be manipulated by multiple users simultaneously. What we mean by single user is that it didn't matter whether one or more users manipulated the scenario, the outcome was always the same. Not so with the BIM4VET scenario. Even though the scenario could well be used by a single user, the collaborative nature of the task to be performed, i.e. deciding at team level whose responsibilities are to evolve to meet the requirements set forth by the BIM project, encourages this scenario to be used by all involved participants in collaboration.

Even though far from feature-complete, the Tulip framework and its available extensions allowed us to implement the BIM4VET scenario with a minimum amount of coding. As outlined in 1.4.3, most of the coding effort went into the `ProfileWidget` and the respective `ProfileModes` the widget is managing.

BIM4VET Partners

LIST / Luxembourg Institute of Science and Technology	
Cardiff University	
LIST / CEA tech / Commissariat à l'Energie atomique et aux énergies alternatives	

BIM4VET contact & website

Contact:

annie.guerriero@list.lu

sylvain.kubicki@list.lu

Website:

<http://www.bim4vet.eu/>