WP 3 System Integration and Testing

Dissemination Level: Public

Lead Editor: Johan Kristiansson, ERI

26/03/2015 + adjustments

Status: Submitted

SEVENTH FRAMEWORK PROGRAMME THEME
ICT -2013.1.6
Project full title: COnnected Media and Presence from European Institute of Technology
Grant agreement no.: 611324
Context

WP 3 System Integration and Testing
WPL Seppo Valli (VTT)
Task 3.4 First Prototype
TL Joakim Norrgård (ERI)

Dependencies
This deliverable uses specifically the input of D3.2 Initial System Specification, D4.1 Virtual Cameras 1.0, D4.2 Information Agents. D5.2 Shared Tangible Experience Service and D5.3 Broadcast Presence Studio 1.0 have been added to it as basis for T6.2 First Pilot and T7.2 Formative Evaluation.

The first prototype also includes elements reported in D4.3 Proxies and Devices (multi-device apps) and D4.4 Immersive Spaces (SharedSpace presence mode).

Starting date 24/11/2014
Release date 26/03/2015, adjusted for Year 2 submission 2015-10-27

Author(s) Joakim Norrgård (Ericsson), Nicklas Nyström (LTU), Jimmy Nyström (LTU), Harold Nefs (TUD), Seppo Valli (VTT), Maor Conforti (LiveU), Charlie Gullström (KTH)
Contributor(s) Ericsson, LTU, TUD, LiveU, KTH, VTT
Reviewers Pekka Siltanen (VTT), Joke Kort (TNO)
Approved by: PMT

History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Authors</th>
<th>Sections Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>22/11/2014</td>
<td>Joakim Norrgård</td>
<td>Initial version</td>
</tr>
<tr>
<td>0.2</td>
<td>24/11/2014</td>
<td>Seppo Valli</td>
<td>Responsibilities for editing, writing, and review (tentative)</td>
</tr>
<tr>
<td>0.3</td>
<td>27/11/2014</td>
<td>Harold T Nefs</td>
<td>Add first drafts of Connection Agent, Virtual Camera, and Information Agent</td>
</tr>
<tr>
<td>0.4</td>
<td>27/11/2014</td>
<td>Harold T Nefs</td>
<td>Add first draft of Immersive spaces + small changes + added text from Maor in v0.2 for the BPS</td>
</tr>
<tr>
<td>0.5</td>
<td>28/11/2014</td>
<td>Johan Sjöberg</td>
<td>Editing before review and input from LTU</td>
</tr>
<tr>
<td>0.6</td>
<td>1/12/2014</td>
<td>Johan Sjöberg</td>
<td>Edit after review from Pekka and Joakim</td>
</tr>
<tr>
<td>0.7</td>
<td>2/12/2014</td>
<td>Pekka Siltanen, Seppo</td>
<td>Restructuring and text input</td>
</tr>
<tr>
<td>Revision</td>
<td>Date</td>
<td>Author(s)</td>
<td>Notes</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>-------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>0.8</td>
<td>3/12/2014</td>
<td>Johan Sjöberg</td>
<td>Polishing and added appendix</td>
</tr>
<tr>
<td>0.9</td>
<td>4/12/2014</td>
<td>Johan Sjöberg</td>
<td>Further polishing</td>
</tr>
<tr>
<td>0.10</td>
<td>10/12/2014</td>
<td>Johan Sjöberg</td>
<td>Moving one section, fixing the table of content</td>
</tr>
<tr>
<td>0.11</td>
<td>19/12/2014</td>
<td>Johan Sjöberg</td>
<td>Joakim updated the system diagram</td>
</tr>
<tr>
<td>0.12</td>
<td>19/12/2014</td>
<td>Pekka Siltanen</td>
<td>Architecture diagram turned to landscape.</td>
</tr>
<tr>
<td>0.13</td>
<td>9/3/2015</td>
<td>Joakim Norrgård</td>
<td>Updated document for current prototype</td>
</tr>
<tr>
<td>0.14</td>
<td>10/3/2015</td>
<td>Nicklas Nyström</td>
<td>Added description on shared videos in Immersive Spaces.</td>
</tr>
<tr>
<td>0.15</td>
<td>13/3/2015</td>
<td>Joakim Norrgård, Seppo Valli</td>
<td>Updated according to comments from Joke Kort and Pekka Siltanen</td>
</tr>
<tr>
<td>0.16</td>
<td>16/3/2015</td>
<td>Maor Conforti</td>
<td>Update of Broadcast Presence Studio description</td>
</tr>
<tr>
<td>0.17</td>
<td>22/3/2015</td>
<td>Johan Kristiansson</td>
<td>Polishing, final version.</td>
</tr>
<tr>
<td>0.18</td>
<td>26/3/2015</td>
<td>Charlie Gullström, Seppo Valli</td>
<td>Improvements to Introduction, small changes to text.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Approved by Project Management Team.</td>
</tr>
<tr>
<td>1.0</td>
<td>27/10/2015</td>
<td>Johan E. Bengtsson</td>
<td>Final adjustments for Year 2 submission, linking to other deliverables, slight restructuring and minor improvements. Expanded the Summary.</td>
</tr>
</tbody>
</table>
# Table of Contents

Summary.................................................................................................................................................. 7

1. Introduction........................................................................................................................................... 8

2. System Architecture............................................................................................................................ 9
   2.1. System Structure......................................................................................................................... 9
   2.2. Deployment..................................................................................................................................... 11
      2.2.1. Deployment platform (Docker).......................................................................................... 11
      2.2.2. Mashup deployment (Dashboard)..................................................................................... 11

3. Service components ............................................................................................................................ 13
   3.1. Shared Tangible Experience Service.......................................................................................... 13
   3.2. Broadcast Presence Studio 1.0.................................................................................................... 15
      3.2.1. Clients Implementation........................................................................................................ 16
      3.2.2. Servers Implementation........................................................................................................ 19

4. Mediated Presence components........................................................................................................ 20
   4.1. Virtual Camera.............................................................................................................................. 20
   4.2. Information Agent........................................................................................................................ 20
   4.3. Tangible Interaction ..................................................................................................................... 21
   4.4. Immersive Spaces ....................................................................................................................... 23

5. Platform components.......................................................................................................................... 25
   5.1. Connection Agent ....................................................................................................................... 25
      5.1.1. Client-Side implementation.................................................................................................. 26
      5.1.2. Server-Side implementation................................................................................................ 26
   5.2. Multi-Device System ................................................................................................................... 27
   5.3. COMPEIT Cloud platform ......................................................................................................... 29
   5.4. COMPEIT Media Framework ..................................................................................................... 29

6. Future development............................................................................................................................... 30

Appendix 1 – Users Guide....................................................................................................................... 31
List of Figures

Figure 1. System architecture for the first prototype .......................................................... 10
Figure 2. The Dashboard in the first prototype .................................................................. 12
Figure 3. Shared Tangible Experience Service architecture view ........................................ 13
Figure 4. Shared video viewing, using an Immersive Space ................................................ 14
Figure 5. Multi-device launch menu .................................................................................. 14
Figure 6. Multi-device slave apps (examples) .................................................................... 15
Figure 7. BPS architecture view ....................................................................................... 16
Figure 8. BPS Viewer user interface ................................................................................ 17
Figure 9. LU Smart HD User Interface .............................................................................. 18
Figure 10. LU-Central Web User Interface ....................................................................... 19
Figure 11. Virtual Camera user interface .......................................................................... 20
Figure 12. Information Agent user interface .................................................................... 21
Figure 13. User Agent user interface .............................................................................. 22
Figure 14. Object Agent - adding a new connection between smart devices ....................... 22
Figure 15. Tangiball - displaying a colourful animation ....................................................... 23
Figure 16. Immersive Space with one participant using background removal ................... 24
Figure 17. ConnectionAgent - a typical instantiation .......................................................... 26
Figure 18. Multi-device (Play) applications - launching from the COMPEIT dashboard .... 27
Figure 19. Multi-device (Play) applications - QR-connecting a mobile device .................. 28
<table>
<thead>
<tr>
<th>Abbreviations and Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BPS</strong></td>
</tr>
<tr>
<td><strong>CA</strong></td>
</tr>
<tr>
<td><strong>CDN</strong></td>
</tr>
<tr>
<td><strong>Chroma key</strong></td>
</tr>
<tr>
<td><strong>COMPEIT</strong></td>
</tr>
<tr>
<td><strong>Docker</strong></td>
</tr>
<tr>
<td><strong>DOM</strong></td>
</tr>
<tr>
<td><strong>Flask</strong></td>
</tr>
<tr>
<td><strong>IA</strong></td>
</tr>
<tr>
<td><strong>iBeacon</strong></td>
</tr>
<tr>
<td><strong>ICE</strong></td>
</tr>
<tr>
<td><strong>NAT</strong></td>
</tr>
<tr>
<td><strong>Nginx</strong></td>
</tr>
<tr>
<td><strong>P2P, Peer-to-Peer</strong></td>
</tr>
<tr>
<td><strong>PHP</strong></td>
</tr>
<tr>
<td><strong>Python</strong></td>
</tr>
<tr>
<td><strong>SQLAlchemy</strong></td>
</tr>
<tr>
<td><strong>SQLite</strong></td>
</tr>
<tr>
<td><strong>STUN</strong></td>
</tr>
<tr>
<td><strong>TURN</strong></td>
</tr>
<tr>
<td><strong>VC</strong></td>
</tr>
<tr>
<td><strong>WebRTC</strong></td>
</tr>
<tr>
<td><strong>WebSocket</strong></td>
</tr>
</tbody>
</table>
Summary

This report accompanies the release of the first COMPEIT prototype, documenting its system components. The prototype includes system components for Services (from WP5), Mediated Presence (from WP4), and Platform functionality (from WP3).

The COMPEIT project has produced components that are mostly independent of each other. This report describes how they have then been integrated to represent the technical result from COMPEIT, intended for use in T6.2 First Pilot.

The system architecture has been updated to reflect the latest system components. The principles for deployment using Docker containers are described together with a Dashboard mashup example of how the front-end can be constructed.

Two main services have been implemented by WP5 and then integrated into the First Prototype. Shared Tangible Experience Service is a powerful media consumption service with immersive spaces that lets users experience being in the same virtual place while jointly enjoying playback of videos using multiple devices to control it. Broadcast Presence Studio 1.0 enables broadcasters or informal event producers to choose among live video streams captured by a smartphone app, and pass it on to content distribution networks feeding into YouTube and broadcast networks.

Four Mediated Presence components have been developed by WP4 and integrated into the First Prototype. Virtual Camera enables receivers of video to modify key characteristics of the video such as the viewing position (with strong limitations). Information Agent implements limited information connectedness by enabling users to indicate their current mood. Tangible Interaction enables smart objects and people to inject and receive key information, under authorization control. Immersive Spaces enables users to connect spatially and socially by experiencing being in the same virtual space with the possibility to shroud (pixelate) the own video.

Key platform components from WP3 are the Connection Agent for interconnecting users with streams and objects, the Multi-Device System for enabling shared apps on multiple screens, the Cloud Platform for deployment, and the Media Framework for server-based media processing.

The described first prototype is a snapshot, which will be further developed throughout the project. Through coming iterative development, further integration of (new) components and services will be made resulting in a second prototype. Main areas of focus for further development, in the short term to midterm, are improving the user interface, ease of deployment, and adding more components using APIs for server based media processing.
1. Introduction

Following D2.3 High-Fidelity Mock-up in May 2014, system component development proceeded within WP4 and WP5 resulting in several features that step by step have been integrated to a first prototype as a result of T3.4 Integration and System Testing. The current brief report that accompanies the First Prototype therefore has dependencies on several previous reports where many of the COMPEIT components were already presented e.g. D4.1 Virtual Camera 1.0, D4.2 Information Agent, D4.4 Immersive Spaces, D4.3 Proxies and Devices, D5.1 User Profiler, D5.2 Shared Tangible Experience Service, and D5.3 Broadcast Presence Studio 1.0.

In a first version, ready by December 2014, the First Prototype included some of the above features: Virtual Camera 1.0 based on Kinect, the Information Agent and some Proxies and Devices. The first version was not submitted, mainly because its set of components and functionalities was considered too restricted for piloting. Continued system integration and testing resulted in changes to the user interface to improve the usability. Further, the Broadcast Presence Studio service was added, alongside several improvements relating to the structure, functionalities, and user interface of the first prototype, including among other things user authentication and system management.

More recently, other features resulting from the technical work packages have been added. Parallel work in WP4 in 2014 towards D4.4 Immersive Spaces, for example resulted in novel features that incorporated chromakey technologies and that were tested in a service component co-developed by T4.2 Mediated Spaces and T5.1 Shared Media Distribution throughout the autumn. The main features are called SharedSpace, a virtual 3D environment which users can redesign and populate by scaling and rearranging their avatar-like live video stream representations; and PixelPresence which is an overlay technique that offers a pixelated (shrouded) view of a live video stream, thereby adding an ambient and discrete presence filter, which indicates movement and subtly prompts users to enter into mediated interaction. An initial version of the above are now integrated to the First Prototype.

By February 2015, a second version of the first prototype was deemed mature enough to be tested through expert evaluations in WP6, which produced many valuable observations to improve the system. T6.2 First Pilot is expected to provide valuable user feedback for continued system development in WP3-WP5.

As we now submit D3.4 First Prototype, it is important to stress that several of the system components are in a process of iterative development in WP3-WP5, please refer to the specific deliverables describing the platform and framework (WP3), Mediated Presence Components (WP4) and New Media Services (WP5).

An updated system architecture for the first COMPEIT prototype architecture is briefly described in Section 2, updating the architecture described in D3.2 Initial System Specification. The system components included in the first prototype are described in chapters 3-5. To provide deployment flexibility the system is deployed using Docker containers in the COMPEIT Cloud Platform described in D3.3 Platform and Framework 1.0 (non public). The current prototype is deployed at [http://prototype.compeit.eu](http://prototype.compeit.eu). A short introduction and user instructions for the prototype are included in Appendix 1.
2. System Architecture

2.1. System Structure

The first prototype gathers the COMPEIT platform, components and services and integrates them into a generic system, see diagram on the next page. The development results from WP3-WP5 are clearly visible as described below.

The default user interface is a mashup that provides access to the different system components in a single graphical interface, the dashboard as described in D5.1 User Profiler and should not be viewed as the final service/platform interface design which is instead implemented as user services in WP5.

The user interface consists of a set of client side components, written in JavaScript. The client side components communicate with server side components, which have common webpage template rendering, persistent storage database and user authentication and authorizations mechanisms.

The WP3 development results are included as the Frameworks and Databases system components, and the Connection Agent components implementing the Media Framework described in previous WP3 deliverables. The Multi-Device Interaction Framework described in previous WP3 deliverables is a server-side component that includes a session manager and signal servers. The Platform is described in more detail in in section 5.3 and in previous WP3 deliverables.

The WP4 development results are included as the client-side components Virtual Camera (T4.1), Immersive Space (T4.2 Mediated Spaces), Object Agent (T4.3 Tangible Interaction) and Information Agent (T4.4). Most of them also have server-side parts. The Object Agent implements the Tangible Framework described in D3.2 Initial System Specification.

The WP5 development results are included as client side components: Master/Slave applications (T5.1 Shared Media Distribution) and T5.3 Broadcast Presence Studio. T5.4 Mixed Reality Sharing is for future work. Please note that the multi-device functionality is beyond the expected result for the first prototype.

The prototype is deployed using COMPEIT Cloud Platform described in previous WP3 deliverables where the components, if appropriate, have been implemented as Docker containers. This flexibility allows the prototype to be deployed using Docker on a single system or in the cloud. Figure 1 below shows the architecture of one deployed system. The prototype can use an nginx-proxy for security, to provide a proxy from https to http for verified encrypted communication.
Figure 1. System architecture for the first prototype.
2.2. Deployment

2.2.1. Deployment platform (Docker)
Docker is an open platform to build, ship, and run distributed self-contained applications. Docker Engine allows the users to package and run applications. The docker registry service enables sharing of applications. Docker packages the applications into disk images that are given names, e.g. delivery34/dashboard. Images are stored into public or private registry server. The images can be searched from the registry by their name, description, or the user responsible for the image. The images can then be downloaded and run using the Docker Engine.

A docker image comprises just the application and its dependencies. It runs as an isolated process in the user space of the host operating system, sharing the kernel with other Docker containers. However, the images can communicate with each other e.g. using an HTTP interface, making it possible to integrate different images into a larger application.

COMPEIT has established an own private Docker registry, located at https://docker-registry.compeit.verkstad.net:443.

All the COMPEIT components described in this document are stored in the COMPEIT Docker registry and the component names are mentioned at the end of each component description in the following sections.

2.2.2. Mashup deployment (Dashboard)
Dashboard is an upper level mashup web application that integrates the COMPEIT components. The Dashboard consists of two parts: the user authentication and the mashup user interface. Please note that the COMPEIT services from WP5 are alternative ways for a user to enter the COMPEIT system, potentially representing each service as a COMPEIT room type (see below).

The Dashboard authentication keeps track of users logged into the system and provides this information using a WebSocket interface to the Dashboard UI. The Dashboard UI can signal user login and log out events to other components.

Dashboard provides the concepts of rooms where people can meet. The rooms have a specific type that indicates which system component that provides the user interface for that room.

Authorization is provided using a Circles concept for representing groups of specific users, or other users being in an active service session with the user giving authorization. The user can assign other users to Circles and then give a specific Circle access to a room (or permission to an object).

Dashboard (Figure 2) is implemented using the Python Flask framework to provide basic user authentication. It uses the SQLAlchemy Object Relational Mapper (ORM) to provide persistent storage of information to the database (currently SQLite), such as the users and their passwords.
Figure 2. The Dashboard in the first prototype.

The Flask rendered templates, which accesses information stored in the SQL database are stored in the directory path `dashboard/app/templates/`. For most of the first prototype components, the dashboard is implemented as room types available in subdirectories in `dashboard/app/templates/components/`. The integrated modules/components are deployed in the static directory path `dashboard/app/static`.

Docker image: `delivery34/dashboard` at https://docker-registry.compeit.verkstad.net:443
3. Service components

3.1. Shared Tangible Experience Service

This generic service enables multiple individuals to e-meet and jointly view media from the Internet on multiple devices and using non-conventional displays and new interaction methods. It is expected that the service will be adapted to different uses both by adding new shared apps and by creating new room types that implement various 3D scenes for interaction. See D5.2 Shared Tangible Experience Service for more details on the service and its architecture and software modules.

The prototype implementation includes the following service components (Figure 3):

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETI Engine</td>
<td>A JavaScript library used by Master and Slave applications, making it possible for videos to emit events at specific points in time during playback and create interactive content on a web page that listens for those events. It has an Event Emitter for video playback and keeping the video in sync with an event timeline, a Scene Graph for rendering user interfaces of slave applications (typically on secondary screens), and an Action Controller for managing sets of objects that perform some action on a node in the scene graph. Notably, the Action Controller can execute compound actions like Repeat, Sequence and Group.</td>
</tr>
<tr>
<td>Multi-Device System</td>
<td>A system that allows computers, handheld devices and smart objects to communicate. It enables automatic device detection and connection establishment, and provides an interface for transmitting arbitrary data objects between devices. It has now been incorporated as part of the COMPEIT platform, see section 5.2.</td>
</tr>
<tr>
<td>Device Agent</td>
<td>This mechanism can be used to connect tangibles to the main COMPEIT system. More generally, it allows smart objects and applications to listen to events triggered by other smart objects/applications/people, enabling new ways of interacting with one’s environment, as well as with other users. This component was described under the name User Profiler in D5.1 User Profiler, and has since been renamed Object Agent from T4.3 Tangible Interaction, see section 4.3.</td>
</tr>
<tr>
<td>Immersive Spaces</td>
<td>This component allows to remove the background and thus merge video feeds from multiple locations and users. This component was co-developed by T4.2 Mediated Spaces and T5.1 Shared Media Distribution. It is further described in section 4.4.</td>
</tr>
</tbody>
</table>

Figure 3. Shared Tangible Experience Service architecture view.

This service shared video viewing while e-meeting in an immersive space (Figure 4). The background image of the immersive space is replaced with a Youtube video, which is visible for all users currently in the room. The video is automatically synchronized, providing a coherent view for all participants; one can play, pause, rewind, and fast-forward the video, and the changes are mirrored for all connected users.
Using the Multi-Device System described in section 5.2), additional devices can be used as remote controls, or to browse and launch Youtube clips (Figure 5 and Figure 6).

Figure 4. Shared video viewing, using an Immersive Space

Figure 5. Multi-device launch menu.
3.2. Broadcast Presence Studio 1.0

This service enables broadcasters to select among user-generated live video streams, for broadcasting purposes or for joint viewing in COMPEIT sessions. See D5.3 Broadcast Presence Studio 1.0 for more details on BPS use cases, design, and implementation.

Prototype implementation includes the following components (Figure 7):

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPS Tab</td>
<td>A web page in the COMPEIT prototype website that contains the main UI for BPS. It allows the user to select video to watch, watch the video, and communicate with video providers.</td>
</tr>
<tr>
<td>LU-Smart</td>
<td>A mobile app that transmits HD video to a cloud server using a proprietary bonding technology.</td>
</tr>
<tr>
<td>LU-Central</td>
<td>Management web console to control and monitor video transmitting devices and servers.</td>
</tr>
<tr>
<td>Cloud MMH Server</td>
<td>Cloud server that receives a video stream via multiple channels from the LU-Smart and encodes a standard output stream.</td>
</tr>
<tr>
<td>CDN Service</td>
<td>A CDN service provided by LimeLight to distribute a standard video stream over the web.</td>
</tr>
</tbody>
</table>
COMPEIT Prototype Website and COMPEIT Prototype Server are described in section 2.1.

3.2.1. Clients Implementation

The BPS prototype has 3 clients, as depicted in Figure 7.

3.2.1.1. Broadcast Presence Studio – Prototype User Interface

The prototype user interface shows a video player to the right of a static, non-interactive, image of a map (Figure 8). The map image is a placeholder for an interactive map component to be developed.
The map shows the locations of friends and other broadcasting users. We plan to make those pins clickable so that the viewer can select which stream they want to see on the video player that is on the right side. Currently the player shows a single stream coming from a dedicated iPhone running the LU-Smart HD application. It will be possible to make this part of a BPS room type as described in section 2.2.2, enabling distributed production teams to collaborate.

3.2.1.2. LU-Smart HD application running on iPhone 5s

LU-Smart HD is an iOS Application available on AppStore (Figure 9). An Android version is planned.

The LU-Smart HD transmits video to a cloud server using LiveU’s patented bonding technology. On the server the video is transcoded and streamed to browsers as seen above in section 3.2.1.1.
It will be easy to add an icon (QR code) for launching the LU Smart HD as a "slave app" from the PBS room type.

3.2.1.3. LU-Central Web UI

LU Central is a management service for both the LU-Smart and the LiveU Cloud Server (Figure 10).

For COMPEIT PBS the LU-Central is used as a one-time setup configuration tool where the system administrator can direct the stream from mobile units (LU-Smarts) to the dedicated cloud server, and from there to CDN servers.
It is going to be possible to send the selected video contribution to a live YouTube stream that can be viewed by the shared YouTube viewer implemented in D5.2 Shared Media Experience service.

3.2.2. Servers Implementation

3.2.2.1. COMPEIT Prototype Application Server
For BPS the COMPEIT Prototype Application Server described in section 9 provides a framework to run on. Specifically for PBS it provides static resources (HTML/JavaScript/PNG/CSS files). However, it will be easy to integrate PBS as a COMPEIT room type.

3.2.2.2. LU-Central
LU-Central server monitors and controls the LU-Smart and the LiveU Cloud Server. It applies the pre-defined configuration to the LU-Smart units as they are activated. It also monitors their proper operation. The LU-Central has no Docker configuration as there is only a single logical instance of it always running in the cloud.
4. Mediated Presence components

4.1. Virtual Camera

D4.1 Virtual Camera 1.0 gives detailed information about this component. It is implemented as a client-side JavaScript object that gives the user a 3D view of a remote location in (near) real time from any desired viewpoint. This freedom in choosing the camera viewpoint allows for interesting applications such as motion-parallax enabled displays, 3D audio/video communication but also the function of compensating for the parallax in gaze direction with respect to the camera.

The virtual camera requires a ConnectionAgent to communicate 3D data and skeleton data to other users. The virtual camera requires a Kinect for Windows 1.8 camera.

The current prototype displays the virtual camera system as a module (Figure 11).

![Virtual Camera user interface](image)

Figure 11. Virtual Camera user interface.

Standalone Docker Image: `library/VirtualCamera`

4.2. Information Agent

D4.2 Information Agents gives detailed information about this system component. It collects and distributes specific information about the users, such as availability, activity, mood and location. The information can be provided by the users themselves or be extracted automatically from sensor data. The InformationAgent consists of a client-side JavaScript object and a collection of PHP scripts on the server side. The InformationAgent requires a persistent storage space. Further, it requires that users are properly logged into the system and that their usernames are made known to the InformationAgent in order to store information properly in the right place.

The InformationAgent requires a ConnectionAgent to communicate with other users, and a VirtualCamera (optional) for skeleton tracking.
In the current prototype the InformationAgent is not displayed as component but as part of the homepage as shown in the middle column in Figure 12. However, in the near future, we will extend the InformationAgent with remote information from other users as well. The visual output of the InformationAgent object will therefore also be available to other components and shown in other parts of the front-end of the system.

![Figure 12. Information Agent user interface.](image)

Standalone Docker Image: `/library/InformationAgent`
Server Docker image: `/deliver34/InformationAgentServer`

### 4.3. Tangible Interaction

The Object Agent component is currently implemented in the Dashboard, providing a REST-API to objects and the client UI. The objects/components can register themselves and provide information about their inputs and outputs, and then the UI presents the objects to the users and allows the users to connect them (Figure 13 and Figure 14), as described in D5.1 User Profiler. The objects and the connections are persistently stored in a database.
The COMPEIT first prototype supports Tangiball (Figure 15) described in D4.3 Proxies and Devices. It is a smart device with two LED displays: one for direct communication, and one for displaying data in an ambient way. Three touch points enable simple, tangible interaction, while a Web application provides more fine-grained control. The web
application can be used for setting the ambient light on the Tangiball, sending "light smileys", and creating light animations.

Figure 15. Tangiball - displaying a colourful animation

The Tangiball can be connected to other COMPEIT components through the Object Agent; doing so will prompt it to display certain animations in response to triggered events. This is done via the Object Agent user interface.

Docker Image: The Object Agent is integrated in the dashboard image. delivery34/dashboard

4.4. Immersive Spaces

D4.4 Immersive Spaces gives detailed information about this system component. It is made available as a client-side JavaScript object. It uses its own server process for P2P video and relaying messages for calibration and positioning of user videos..

Immersive Spaces has a SharedSpace part. It places all user videos together in the same perceived space, replacing the users' respective backgrounds using Chroma Key filtering with a common static image or even a background video. For the background removal to work properly, each user needs to have a uniformly coloured (preferably green or blue) wall or screen behind them. The application automatically detects what colour range that is to be removed from the source video. This can be further fine-tuned by the users themselves if necessary. Users can resize and move around each video within the immersive space, and these changes are propagated to all participants. Figure 16 below shows the Immersive Space with one user sitting in front of a uniformly coloured wall, and another sitting in a regular office.
Figure 16. Immersive Space with one participant using background removal.

Immersive Spaces also allows the user to pixelate (shroud) their own sent video. See more in D4.4 Immersive Spaces 1.0.

The immersive space room type was co-developed with T5.1 Shared Media Distribution, so that D5.2 Shared Tangible Experience Service is an immersive space with shared video viewing and multi-device apps (section 5.2), and support for tangible interaction (section 4.3).
5. Platform components

This chapter expands on the descriptions given in the non-public D3.3 Platform and Framework 1.0.

5.1. Connection Agent

The ConnectionAgent is a new system component that is closely related to the COMPEIT media framework described in section 5.4. It is provided as a client-side JavaScript object that builds the real-time communication channels between users using WebRTC. The ConnectionAgent first uses a WebSocket server to make the initial connection between users. Then, the users' browsers negotiate over the WebSocket connection a direct peer-to-peer (P2P) connection for audio, video and data communication in real time. The peer-to-peer negotiation uses an ICE (STUN/TURN) protocol to find a direct connection over the Internet. In order to make a peer-to-peer (P2P) connection, the peers must have a public IP address, which is often not the case when the users are behind a NAT router. The ConnectionAgent therefore can also use a TURN server for audio/video/data relay in the case a direct P2P connection cannot be established.

In principle, the ConnectionAgent can handle an unlimited amount of peer connections. However, the available bandwidth and processing power of the client is the limiting factor, since the peer-to-peer communication means that each client receives all the video streams from all other users in the same session - this limitation will be addressed in a future prototype version by automatically moving to server-based video distribution when needed.

The end-user is not aware of the ConnectionAgent, as there is no need for the user to actively interact with it. The ConnectionAgent is however used extensively throughout the system for connecting users and is responsible for communication (including audio/video) between users. The ConnectionAgent dispatches JavaScript customEvents for which other objects can register. For example the InformationAgent may register for events that contain newly arrived information from a remote InformationAgent.
5.1.1. Client-Side implementation

A typical instantiation of a ConnectionAgent is shown in Figure 17. The ConnectionAgent is instantiated as a JavaScript object as usual. One may specify an own user name and a DOM element where the local and remote audio/video streams are displayed in the browser window. Usually you would also specify a callback function for when local and remote media are added or removed.

```html
<html>
  <div id = "videoContainer">
  </div>
</html>

<script src="ConnectionAgent.js"></script>

var aRoomNumber = 1000; //any number
var aUserName = "Logen"; //any name
aDOMVideoContainer = document.getElementById('videoContainer');

//override/implement these callback functions
function localMediaAddedFunc(){};
fuction localMediaRemovedFunc();
function remoteMediaAddedFunc();
function remoteMediaRemovedFunc();

//instantiate now
connectionAgent = new ConnectionAgent();
connectionAgent.setRoom(aRoomNumber);
connectionAgent.setUsername(aUserName);
connectionAgent.setVideoContainer(aDOMVideoContainer);
connectionAgent.onUserMedia(localMediaAddedFunc, localMediaRemovedFunc);
connectionAgent.onRemoteMedia(remoteMediaAddedFunc,
remoteMediaRemovedFunc)
connectionAgent.start();
</script>

Figure 17. ConnectionAgent - a typical instantiation.

5.1.2. Server-Side implementation

The ConnectionAgent runs completely on the client-side. However, it uses two auxiliary servers to make P2P connections. 1) a WebSocket server and 2) a standard ICE (STUN/TURN) server. We have set up a couple of these (non-public) servers. These servers are also made available as Docker images.

Server Docker Image: /delivery34/ConnectionAgentServer

Client is implemented as a software module in dashboard/static/ConnectionAgent
5.2. Multi-Device System

The COMPEIT prototype allows computers, handheld devices and smart objects to communicate using a multi-device interaction framework (described in a previous WP3 deliverable). It enables automatic device detection and connection establishment, and provides an interface for transmitting arbitrary data objects between devices.

The multi-device system comprises a number of components:

1. **Database and management console** - Allows users to register their own web applications with the multi-device system.
2. **Session manager** - Manages a set of signaling servers, multi-device sessions and supplies proxy scripts for developers.
3. **Signaling server** - WebRTC signaling with WebSocket fallback. Also supplies embeddable multi-device popup menu.
4. **Application server** - A number of example multi-device applications.
5. **Service discovery components** - A NodeJS server and an iOS application for service discovery and optional location-based access control using iBeacons.

More detailed descriptions of these components are available in the COMPEIT git repository and will be described in the upcoming D3.6 Platform and Framework 2.0 (non public).

Communicating devices within the multi-device interaction framework are divided into two categories: masters that publish services, and slaves that subscribe to the services. A master application can be any kind of web application, and slaves are auxiliary applications that extend them by acting as additional screens and/or offering additional input capabilities.

![Multi-device System Interface](image)

*Figure 18. Multi-device (Play) applications - launching from the COMPEIT dashboard*
As shown in the lower left corner of Figure 18, multi-device applications can be launched directly from the COMPEIT prototype dashboard, where a “Play panel” presents the user with a number of example applications. In the application a menu presents the user with a set of options for connecting handheld devices to a master application. For example, scanning the QR code in the menu leads to a list of slave applications that are compatible with the master application in question, as seen in Figure 19.

![Figure 19. Multi-device (Play) applications - QR-connecting a mobile device.](image)

Server Docker image: `/delivery34/multidevice`

The multi-device interaction framework will be used extensively by the upcoming D5.5 Multi-Device Tangible Experience Service.
5.3. **COMPEIT Cloud platform**

COMPEIT Cloud Platform is an online service for hosting, deploying and managing Docker containers. The main purpose of the COMPEIT Cloud Platform is to make setup and deployment of COMPEIT applications simple and portable, while at the same time making integration easier.

The COMPEIT Cloud Platform consists of a private Docker registry and a web-based dashboard to deploy and manage Docker containers, as well as a software agent called Farmer that is responsible for actually deploying the containers on a computer somewhere.

After a Docker container has been tested locally, a Docker image can be pushed to the Docker registry and thus become available to the COMPEIT Cloud Platform. The cloud platform dashboard can then be used to deploy a container on a Farmer. Currently, the containers are deployed by the Farmers on virtual machines running on an OpenStack server hosted by Ericsson (compeit.verkstad.net). It is also possible for any COMPEIT partner to checkout a Docker container image from the Docker registry and run it locally on a Docker engine.

5.4. **COMPEIT Media Framework**

COMPEIT media framework is a framework for implementing media services, such as COMPEIT's Mixed Reality Interaction Service. The media framework is built on top of the Kurento framework. A detailed description of the COMPEIT media framework is available in D3.3 Platform and Framework 1.0 (non public).

The media framework provides means for developing multimedia services with functionalities: signalling and media functions, and application development concepts: media elements and media pipeline. Signalling functions include e.g. media negotiation, QoS parameterization, call establishment, user registration and user presence. Media functions include e.g. media transport, media encoding/decoding, and media processing. Media element is a functional unit performing an action on a media stream. Media elements can receive a media stream, send the stream to other elements, or process the media stream. A media pipeline is a sequence of media elements receiving, sending, and processing media.

The COMPEIT Media Framework will be used in the next prototype by the components needing media transformation on the server (e.g. Augmented Reality Service components) or ability to route media connections (e.g. creating multipoint WebRTC conferencing instead of point to point connections). The Kurento framework is still rapidly developing and unstable, and therefore it was not included into the first prototype but it will be included in the very short term.
6. Future development

The technical development of the COMPEIT main prototype continues in the technical workpackages for platform, component and service development, based on feedback from WP6 Piloting, and WP7 Evaluation. It will consist of a further integration of results from WP4 Mediated Presence Components in terms of more deeply integrated functionality. In addition, on-going development of the platform targets to support specific development in WP5 New Media Services. The second (final) version of the prototype will be delivered early 2016.

Main areas for the short to mid-term technical developments are:

- Supporting improvement of the user interface and administration in collaboration with WP5 New Media Services.
- Ease of installation and deployment
- Use of common frameworks
  - to have more components and objects using authentication and authorisation
  - to have more components using server based media processing
- Enrich the Object Agent to support more types of tangible objects (e.g. different types of communication, websockets, proxy REST-requests)
- Enable tangible objects to connect to the Information Agent, since it is currently only displaying mood and activity levels
Appendix 1 – Users Guide

The prototype system is a completely open system where anybody can register as a user.

First use Chrome to browse to http://prototype.compeit.eu

Then click on the “Register new user”-button and enter your user details.
This will generate a mail to your given e-mail address, where you should confirm your account.

Click on the confirmation link (or copy it and paste it into your Chrome browser), and then log in using your newly created user.
You are then presented with your homepage view in the Dashboard. The Dashboard shows the Information Agent, Multi-Device System, and all the rooms you have created or can enter, and all your objects are presented. For a new user only the default room is available.

To use the multi-device system, you can click on the Run application icon in the Play panel, second row and third column.

To create your own room you first need to edit your user authorisation information, i.e. which of your friends are allowed to enter your rooms. So, first click on Circles panel below the video!
Here you can use the + button to add new Circles and assign users to them using the Circle Members table. A check mark indicates that the person is a member of the circle and a cross indicates that the person is a non-member. You can toggle membership by clicking on the check/cross indicator.

The special groups ‘All’ and ‘Registered’ are special groups which includes either All users (including guest) or only the registered with confirmed e-mail addresses.

Go back to your homepage by clicking the COMPEIT logo.

To continue to create a room, click on the heading for the My Rooms panel. Then use the + button to add a new room; give it a name and a type, for example Meeting or Immersive Space.
Then authorize circles to your newly created rooms giving other users access. The “All” circle enables all registered users and also guest users (who do not need to register) access to the room.
When returning to your homepage, you will now have more rooms to enter. Click on the room to enter it.

When entering a room you will be presented with the view tailored for that room type.
If the “All” circle is allowed access to the room then you can send the URL of the room to another user, e.g. http://prototype.compeit.eu/room/2

When the guest user enters the URL, a login page where the user can enter his/her name, is shown

Then the room interface is presented: