PrismArch

Deliverable No D5.1

First version of architectural design and integration protocol

Project Title:	PrismArch - Virtual reality aided design blending cross-disciplinary aspects of architecture in a multi-simulation environment
Contract No:	952002 - PrismArch
Instrument:	Innovation Action
Thematic Priority:	H2020 ICT-55-2020
Start of project:	1 November 2020
Duration:	24 months
Due date of deliverable:	31 May 2021
Actual submission date:	4 July 2021
Version:	1.0
Main Authors:	Dimitrios Ververidis (CERTH), Vittorio Bava (Mindesk), Gabriele Sorrento (Mindesk)

Project funded by the European Community under the H2020 Programme for Research and Innovation.

Deliverable title	**First version of architectural design and integration protocol**
Deliverable number	D5.1
Deliverable version	Final
Contractual date of delivery	31 May 2021
Actual date of delivery	4 July 2021
Deliverable filename	Prismarch_D5.1_FirstV_ArchDesign_Integration.pdf
Type of deliverable	Report
Dissemination level	PU
Number of pages	38
Workpackage	WP5
Task(s)	T5.1
Partner responsible	MINDESK. Contributors: CERTH
Author(s)	Dimitrios Ververidis (CERTH), Vittorio Bava (Mindesk), Gabriele Sorrento (Mindesk)
Editor	Spiros Nikolopoulos (CERTH)
Reviewer(s)	Edoardo Tibuzzi (AKT), Arun Selvaraj (SWECO)

Abstract	A roadmap for the developments of PrismArch platform, including a draft of the envisioned system architecture, a technical overview of the different modules (WP2, WP4), a timeline, and dependency map.

Keywords	System design, integration roadmap, components definition

Copyright

Deliverable history

Version	Date	Reason	Revised by
0.1	17/03/2021	Table of Contents	Dimitrios Ververidis (CERTH)
0.2	14/5/2021	Study on the technical requirements (Section 2)	Dimitrios Ververidis (CERTH)
0.3	5/6/2021	Technical requirements dependency map (Section 2)	Vittorio Bava (Mindesk)
0.5	20/6/2021	Architecture Design (Chapter 3)	Gabriele Sorrento (Mindesk)
0.9	30/6/2021	Integration and Testing (Chapter 4)	Dimitrios Ververidis (CERTH)
1.0	3/7/2021	Review and edits	Spiros Nikolopoulos (CERTH), Edoardo Tibuzzi (AKT), Arun Selvaraj (SWECO)

List of abbreviations and Acronyms

Abbreviation	Meaning
AB	Advisory Board
CA	Consortium Agreement
DR	Deliverable Responsible
EC	European Commission
GA	Grant Agreement
IP	Intellectual Property
IPR	Intellectual Property Rights
PC	Project Coordinator
PHP	PHP: Hypertext Pre-processor
SBM	Supervisory Board Member
ToC	Table of Contents
UML	Unified Modelling Language
QMR	Quarterly Management Report
WP	Work package
WPL	WP Leaders
AEC	Architecture, Engineering and Construction
AR	Augmented Reality
BIM	Building Information Modelling
CAD/CAM	Computer-Aided Design & Computer-Aided Manufacturing
ICT	Information and communication technology
NDA	Non-Disclosure Agreements
SME	Small and Medium-sized Enterprises
UG	User Group
VR	Virtual Reality

Executive Summary

The role of D5.1 is to transform the user requirements posed by D1.1 and the scenarios described in D6.1 into a unified and sound technical blueprint. The user requirements have been studied and classified into categories that allows for their correlation and their positioning in an inter-dependency graph. The additional user requirements that were not foreseen in the Grant Agreement have been added to the overall plan. Third party software libraries are selected and inserted into the overall architecture of the system. A technical design with a topology of servers and services is proposed, and it will be used as a blueprint for the implementation of the system. Apart from the partners involved in the project, we have collaborated with Speckle Systems, an EU company that offers a platform that achieves asynchronous collaboration for several AEC disciplines across the most popular onscreen CAD, CAE and BIM software via a common web-based database. Towards this end, PrismArch is developing and integrating a VR solution with the help of Mindesk in the pipeline of Speckle so that VR can be also a ubiquitous real-time synchronous design and collaboration tool.

Table of Contents

1. Introduction

D5.1 aims to analyse the technical requirements, provide an architectural design of the platform, and define the integration procedure and unit testing protocol. A novel architecture system design is proposed that divides AEC disciplines collaboration in two modalities, namely the Synchronous and Asynchronous Collaboration modalities. These two modalities are expanding what it is considered today as collaboration with IT technologies in the field of 3D design for CAD, CAE and BIM.

The outline of this deliverable is as follows. First, in Section 2, the requirements posed by AEC disciplines in Deliverable D1.1 and D6.1 are categorized, prioritized, and correlated to find synergies among requirements. In Section 3, various diagrams are overlaying the logic, the technologies, the architecture and the timeplan for the developments of the system. Finally, specific pathways for developments, unit testing and integration of technologies per each WP are defined in Section 4.

2. Transforming user requirements to technical requirements, and addressing use case scenarios

The aim of this section is to transform user requirements into technical requirements. Therefore, we provide a categorization, a correlation, a prioritization, and an implementation methodology of the requirements as it was posed by AEC specialists in **D1.1 Requirements definition**, while in parallel taking into consideration the use case scenarios as proposed in **D6.1 Use case scenarios**.

2.1 Summarization and categorization of user requirements

In this section we repeat some of the information given in Deliverable D1.1 - Requirements, and we divide requirements into 3 main types, as described below:

A) **Management tools**: tools that will arrange the coordination among AEC users and teams;
B) **Information tools:** Tools that provide structure to the flow of information.
C) **Design tools:** tools that will allow the disciplines to interact with the 3d objects.
D) **Visualization tools**: tools to customize user viewing capabilities;

Below we give an overview of the tools that belong to each type. The Management tools are and are described in Table 1, the Information tools are shown in Table 2, the Design tools can be found in Table 3, and Visualization tools are found in Table 4.

Table 1: Management Tools

A. Management Tools	
1. Admin tool (Project Settings)	Managing the access privilege, create discipline-based defaults and settings, edit project schedule, and set tasks
2. Contact / Communication tool	Integration of present-day networkability and communication channels into the VR environment
3. Multi-presence on-boarding system	Managing interactions inside a multi-presence immersive space

Table 2: Information tools

B. Information Tools	
1. Dashboard Tool	A one-stop reference for coordination purposes
2. List Maker Tool	Organizational efficiency
3. Speech to text typing tool	Integration of present-day networkability and communication channels into the VR environment

Table 3: Design tools

C. Design Tools

1. Tagging Tool	Flexible data management system for tracking data inside PrismArch
2. Query Tool	Allowing users to isolate relevant assets from the totality of project information
3. Multi-selection Tool	Allows for highlighting, grouping, isolating, showing/hiding objects
4. White Board Tool	Whiteboard inside VR space, to be able to pin reference images for individual use, or to share during meetings
5. Spatial Orientation Tool	To assist the immersed users with wayfinding inside the project
6. Design Support and Evaluation Tool	Assists with spatial planning and evaluation
7. Commenting Mark-Up Tool	This is a helpful way to keep track of comments and quickly exchange ideas inside the virtual environment.

Table 4: Visualization tools

D. Visualization Tools	
1. Toggle Camera Perspective Tool	Allowing user to view the project from several key perspectives repeatedly without having to travel to them each time
2. Toggle-View Mode Tool	Allowing different ways of viewing and reviewing 3D assets, each suitable for a distinct work activity
3. Clipping Plane Tool	Provides ability to see and evaluate the cross-section of a 3D construction

2.2 Technical requirements

After a careful study on the user requirements of Section 2.1 and the use case scenarios of D6.1, we have assembled a technical blueprint as depicted in Figure 1 and explained in the following lines. Overall, the user requirements are describing a VR metaverse for AEC disciplines where they can assemble teams and work in one collaborative environment in real-time. Basic questions, written with blue, formulate the user contextualization of the system. The first basic question is "How to enter the system?". This is achieved through the executable of the application which is an .exe file for Windows, and a respective file for Mac OS systems. Towards this end, we are using the Unreal Engine which is able to compile the developed code into several platforms such as Windows, Mac, Android, iOS, Playstation and others [Unreal Engine CrossPlatform]. The next question from the user perspective is "Where am I?". **Two types of scenes** are defined, namely the "**Lobby scene"** where it is a preparation space for assembling teams and assigning tasks before delving into the 3D model; and the "**Cospace scene"** where users can actually design and collaborate. The next contextualization question is "How to adapt the system to me?". The core of user requirements and use case scenarios is a **Personal Sphere** which contains all the necessary tools for the user. This personal sphere is the back-pack of

the user and can be used both in Lobby and Cospace with proper contextualization for the tools to be shown in each space.

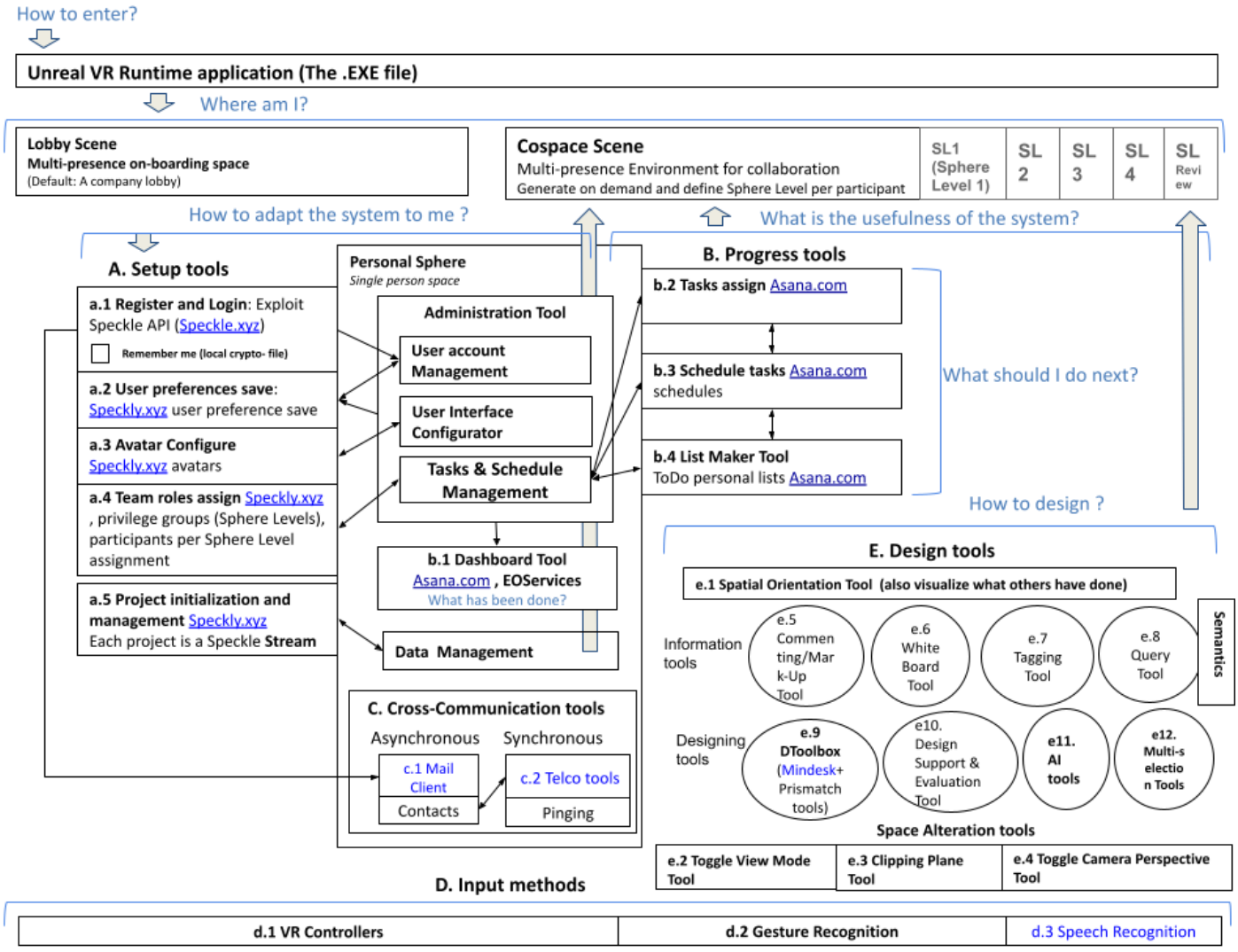


Figure 1: Overall alignment of the user requirements into a technical requirements dependency diagram.

During the initial developments, a mockup for the Personal Sphere was made inside Unreal Engine as shown in Figure 2. It presents the Lobby scene with the UI of the Personal Sphere which allows the user to do setup actions (left sidebar), Manage Projects and Data (central-upper panel), get information about the todo Tasks (central-lower panel), and the Communications panel (right panel).

In the following we describe one by one the technical requirements that were depicted in Figure 1. In this section the naming of the requirement is provided. We provide a numbering that should be used for identifying the requirement, e.g. **a.1 Setup Tools - Registration and Login**. This numbering should be used for reporting issues according to the Issue Reporting Cell found in Section 4 - Integration and testing.

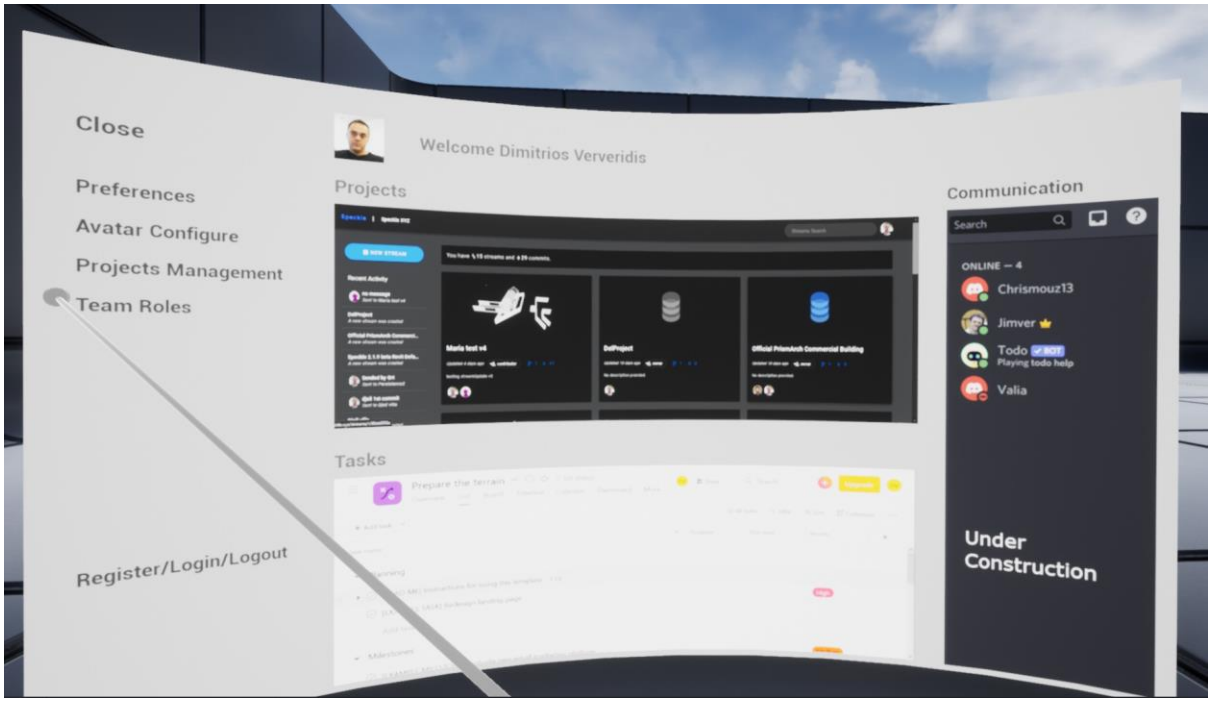


Figure 2: First developments in Unreal Engine 4.

a. Setup tools

The setup actions are the basic actions for any content management system but transferred into the dimensionality of VR. It involves User Account Management actions such as Register and Login; User Interface configuration actions such as User Preferences saving and Avatar Configuration; Team roles assignment and Privileged Groups (Sphere Levels, see D6.1), and participants per Sphere Level. According to Speckle terminology:

a) An Architectural Project is a "**Stream**"
b) Each discipline works in a "**Branch**" of a Stream
c) The **Main Branch** is the Architectural design (otherwise not possible to work fluently)
d) Each insertion of data in the database is a "**Commit**".

These are analysed in the following.

a.1 Registration and Login: The registration allows the existence of the user in the database. It is a required step to assign an id to the user. We will exploit the existing structure of Speckle server [Speckle Server] and Speckle GraphQL API [Speckle GraphQL API] in order to achieve this functionality through VR. Currently required fields are full name, username, mail, and affiliation. We will use the GraphQL mutation functionality to extend it with more fields such as discipline and team name.

a.2 User preferences save: Users can save their preferences regarding the setup of the VR environment, preferred UI tools, active tools and inactive tools. Again, GraphQL Mutation API will be used for storing these preferences in the user entity.

a.3 Avatar configure: Users can configure their appearance in 3D space. In this phase the appearance will be the preferred 3D model, and the colours of the 3D model. Again, GraphQL Mutation API will be used for storing these preferences in the user entity of Speckle.

a.4 Team roles assignment: Here is performed the user account management. The overall entity for an Architectural Project is called a "Stream". The Stream creator is by default Administrator and assigns other Administrators, Collaborators, and Observers of the project. The observers by default will not have editing tools in their Personal Sphere. The aforementioned roles are already defined in Speckle but we will seek to define teams per discipline. The Sphere Level of Accessibility is a request of D6.1. This ontology will define what will be visible by who. The administrator will configure to which Sphere Level each user will belong. The 5 Sphere Levels are delineated in Table 4.

Table 4: Sphere Levels

Sphere Level	Sphere Level 1: **Personal**	Sphere Level 2: **Team**	Sphere Level 3: **Professional**	Sphere Level 4: **Client - Review**	Sphere Level 5: **Public**
Data rights	Only the owner of the change can see the change	Only a team members can see the change	All disciplines but the client can see the changes	The information will be visible by the actual clients.	The information will be visible by anyone.
Editing Tools rights	Discipline specific	Discipline specific	Discipline specific	No editing	No editing

a.5 Project initialization and management: This is where an Architectural Project, namely a "Stream" can be generated and managed.

B. Progress tools

These tools are driving the users on how to contribute to the architectural project, which parts to be responsible for, which conflicts to resolve, and to provide deadlines for each task. Contextually it answers the question "What should I do next?" and also it is part of the question "What is the usefulness of the system?". In order to incorporate the AS-IS scenarios (see D4.1 Interconnection), the Asana management tool has been selected to be incorporated. Thus, the onscreen users can also have accessibility to the system without the need to enter in VR. The tools are related to the previously mentioned tool **a.4 Team roles assign** which is based on the Speckle tool. Ideally, the information about teams should flow bidirectional among Asana [Asana] and Speckle. However, since the integration of Asana API inside Unreal is a task of high effort that was not foreseen in the GA, Asana can be integrated with a Chromium Web Browser Component entity inside Unreal Engine to avoid allocating too much resources on this task. Overall, the technical requirements are described in the following.

b.1 Dashboard tool: It is the briefing of the current project status. It informs the user for the project update, who is now in the system, and about the overall status of the project. It will be based on the Asana tool and the Epic Online Services (EOServices) for finding the users immersed in the system.

b.2 Tasks assign: It is the interface and back-end system for assigning tasks to collaborators. This will be again based on Asana as it provides a full-fledged system for these actions.

b.3 Schedule tasks: It is the time definition for tasks completion. Again, Asana will be used for this action.

b.4 List maker tool: It is a list of notes of what a user has to do next in the system. This will be based on Asana as it offers this feature.

C. Cross-communication tools

In this category belong two tools.

c.1 Asynchronous communication tools: It is a Mail client tool that can be connected to the mail addresses of the user. It is a standard mail client but adjusted for VR environments. We will be based on [Email UE4 plugin] where everyone can connect it with his/her mail server.

c.2 Synchronous communication tools: It is a telecommunication tool for real time chatting, especially useful for communicating to the users that are using onscreen software. Candidate tools are [Vivox] and [Discord]. Vivox is a widely used platform for communication in 3D games. For the integration of Vivox, we will use the [Vivox Core] and [AVRF Vivox Core plugin]. For the integration of Discord, we will use the [Discord SDK] and [Discord-UE4 integration software].

D. Input methods

Input methods define the way that the user interacts with the system. There are three methods:

d.1 VR Controllers: They are the most reliable way that the user can interact with the system. HP Reverb G2 controllers and Oculus Quest 2 controllers are two reliable types of controllers that are constantly used during the developments of PrismArch.

d.2 Gesture Recognition: Gesture recognition by Oculus Quest 2 headset is an alternative way of interacting with the system. Although it is not as robust as the VR controllers, it is mature enough to be used in the project. Unreal Engine already supports the input through Oculus Quest 2 gestures SDK by defining the type of gesture in its “Project Settings - input” item.

d.3 Speech Recognition: Although it is not yet implemented by any vendor, it is also posed as a requirement in D1.1. We will examine the Vivox speech recognition feature, as well as the Web API for speech recognition through Mozilla Voice Open-source technologies [Mozilla Voice]. Mozilla provides pre-trained models as well as Speech data and machine learning software to train new models.

E. Design tools

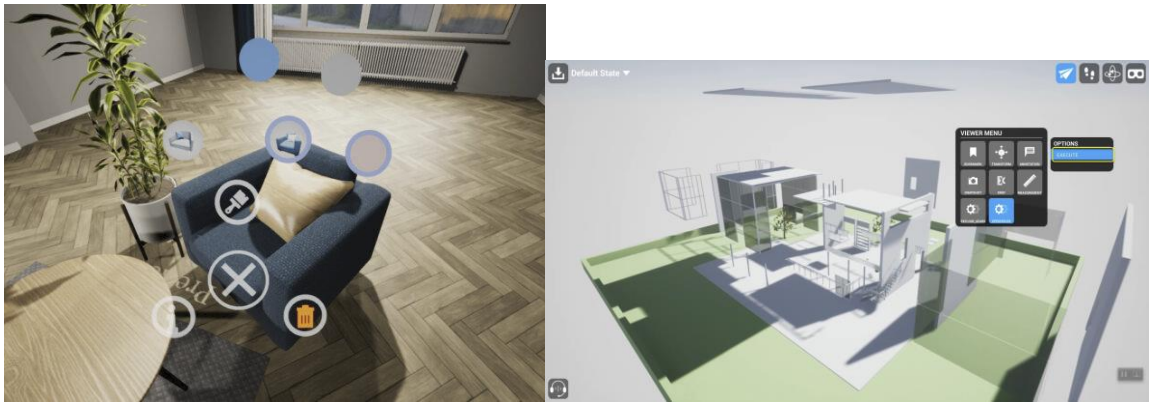
Design tools are divided into four types.

Orientation and Space alteration tools are tools that allow the users to perceive the space and the information they are interested in. Such tools are:

e.1. Spatial Orientation Tool: It allows the user to navigate easily into space, e.g. into certain bookmarked spots, and see what others have done into the certain space. We will be based upon the Advanced VR framework (AVRF) as it provides a way to view the space into a map, to navigate to certain spots and view where the teammates are [AVRF]. Two screenshots of AVRF are shown below. It offers 2D and 3D mapping with users' tele-view and teleportation.

Figure 3: Exploiting AVRF in PrismArch for teleportation, team inspection, and mapping.

e.2 Toggle view mode: Allowing different ways of viewing and reviewing 3D assets, each suitable for a distinct work activity. Examples of viewing modes are white clay mode (Default); Colour coded mode (tagged groups shown in different colours); photo-realistic mode; wireframe mode; xray / ghosted mode; technical mode; simulation mode (point cloud and scan data); Raytracing on/off. Users can save the scene/level contents and load the level contents and materials when they need them. Here we will base our efforts in AVRF and CollabViewer Unreal template by EPIC as they partially support these features, and enhance it where necessary [CollabViewer Unreal template]. Screenshots of AVRF with respect to toggle view mode are shown below.



(a) AVRF material change (b) CollabViewer Explode mode

Figure 4. Changing views for selected objects. CollabViewer also supports xray mode.

e.3 Toggle Camera Perspective Tool: Allowing the user to view the project from several key perspectives repeatedly without having to travel to them each time. This can be achieved with the help of the CollabViewer template as it allows to place bookmarks in certain spots and select these bookmarks via its VR Head Up Display as in Figure 5 “Bookmark”.

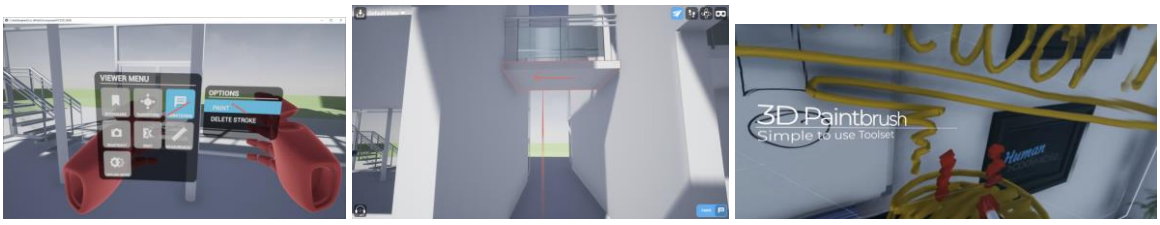
Figure 5: The HUD of Collabviewer. Upper left is the bookmark option to select spots for viewing.

e.4 Clipping Plane Tool: Provides ability to see and evaluate the cross-section of a 3D construction. This allows the user to view a cross-section of a building. It can be achieved with binary operations inside the Unreal Engine.

Information tools

The next subcategory of Design tools is **Information Tools,** namely they are tools that are used for extrapolating or retrieving information about the building. Four tools can be found in this category.

e.5 Commenting / Markup tool: It allows the user to draw annotations on the building. This is a helpful way to keep track of comments and quickly exchange ideas inside the virtual environment. We will base our efforts onto the CollabViewer Paint tool and the 3D Paintbrush of AVRF tools as shown in Figure 6. Also, in PrismArch we develop a note tool for writing letters as annotation.

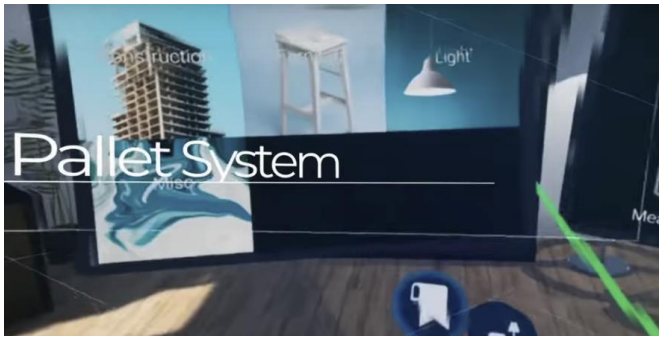


(a) (b) (c)

Figure 6: Paint tool of CollabView (a,b) and 3D Paintbrush tool of AVRF (c) allows to annotate the design.

e.6 WhiteBoard tool: Whiteboard inside VR space will allow the user to pin reference images for individual use, or to share during meetings. We will base our efforts on AVRF as it offers these functionalities as shown in Figure 7.

(a) Screenshot capturing



(b) Pallet for extrapolating images

Figure 7: The Whiteboard tool of AVRF will be used in PrismArch.

e.7 Tagging Tool: Flexible data management system for tracking data inside PrismArch. The tagging tool will be based on Speckle data objects notation which is an extendable way to store 3D information with metadata across major architectural and engineering software. More information about Speckle can be found in D4.1. As regards the interfaces of the tagging tool, we will exploit the interfaces provided by AVRF such as the "Details display" (Figure 8), the "Radial Menu", and the "smartwatch with tablet" interfaces (Figure 9).

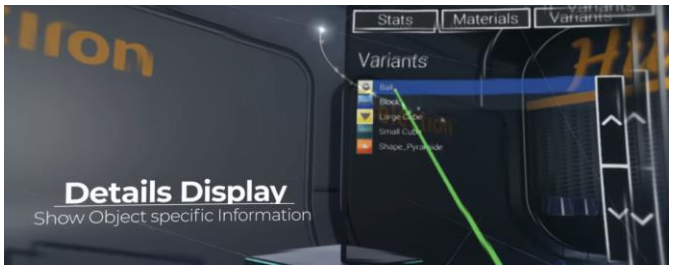


Figure 8: Details display of AVRF.

Figure 9: The radial menu and the smartwatch with table offer enhanced interfaces for the tagging tool.

e.8 Query Tool: The query tool allows to retrieve information from the database where all changes are stored. The database that will be used is the one offered by Speckle as it was described in D4.1. The way data is retrieved is accomplished through PrismArch Semantics

service, namely the Knowledge base which is based on RDF triplets (see D4.1) accessible through a REST service (see Table 5, more details in Section 2 Architecture). In this manner the users in VR can retrieve information about an object across all commits. As regards the interfaces, the AVRF will be used as it offers the radial menu as shown in Figure 10 which can be exploited to interrogate an asset.

Figure 10: Manipulation of information inside VR with AVRF.

Table 5: Examples for calling REST API for Semantic content retrieval from the Knowledge Base.

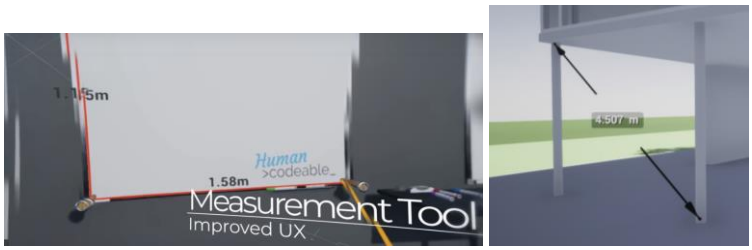
Action	POST COMMAND
Fetch history for a project (commits ids)	curl -d "{}" -v -H "Content-Type: application/json" POST http://160.40.49.211:8080/Prismarch_RetrieveKB/commitHistory
Fetch history for a certain room, e.g. Bath (commits array)	curl -d "{ 'type': 'Bath' }" -v -H "Content-Type: application/json" POST http://160.40.49.211:8080/Prismarch_RetrieveKB/room
Fetch certain parameter for an object based on its id	curl -d "{'objectId': '37902f721d1e6aff15d18274e67ee838' }" -v -H "Content-Type: application/json" POST http://160.40.49.211:8080/Prismarch_RetrieveKB/cost
Fetch designers that edited an object with a certain id	curl -d "{'objectId': '4107cee86039468eafec9596548ba605' }" -v -H "Content-Type: application/json" POST http://160.40.49.211:8080/Prismarch_RetrieveKB/authorHistory

Design tools

e.9 DToolbox: The design toolbox will be based on creating geometries and moving them accordingly in 3D space. Mindesk tools will be used in conjunction with AVRF tools. The interfaces of AVRF tools are shown in Figure 9. The geometries that we are going to support are those offered by Speckle software.

Figure 9: Interfaces for design by the AVRF tools.

e.10 Design Support and Evaluation tool: It is a collection of support tools for design such as: Measuring floor areas and volumes with indication of x,y,z values (Mindesk's annotation-style or tool can be called inside the platform), alternatives for the measurement tools are AVRF (Figure 10a) and Collabviewer (Figure 10b); Bounding box (the same logic to box selection tool in the Multi selection tool) but with the x, y, z values and area/volume annotations; Circulation routing, (e.g. drawing spline route and user object follows the route); Smart staircase modelling; Toggling measurement resolution (mm, cm, m km) to explore measurement resolution; Toggling measurement system - decimal and imperial (feet and inches); When measurement resolution changes, users can see the changes by using the floor grid size or any reference object size change.



(a) (b)

Figure 10: Measurement tool (a) by AVRF; (b) by Collabviewer template.

e.11 AI tools: The idea is that AI tools will give suggestions for design inside VR. More Information about AI functionalities can be found in D2.1. The AVRF tools will be exploited as interfaces. The functionality will be offered by WP2 through a C# modules that will be translated for Unreal with [CLR] extension.

e.12 Multi-Selection tools: Allows for highlighting, grouping, isolating, showing/hiding objects. These tools follow the Rhino3D methodology of selecting objects such as: Box selection method; Users can assign and save tags for single or multiple selected objects for future reviews; Grouping objects; a single object or multiple objects are selectable or highlightable via the Unreal Engine custom-depth/post process, etc.); The highlighting colours are discipline specific; Multiple selected objects can be grouped, ungrouped, inverted, shown and hidden. The selection tools will be based again on AVRF tools (Figure 8) with the proper extension when needed.

2.3 Prioritization

In the following we prioritize the requirements according to what was promised in the GA and according to what it is easy or difficult to achieve.

Table 6: Technical requirements prioritization

Task	Priority	In GA?	Comments
High Priority			
Database interconnection for Asynchronous collaboration (Speckle)	It is highly prioritized because it allows a joint place to save edits	No	Asynchronous collaboration was not foreseen in GA as it was dealing only with synchronous collaboration through the Mindesk plugin. All changes were assumed to be saved within Revit but not to a central server database. However, it is going to be integrated as it was highly recommended by use case partners in order to save changes in a central place.
Design tools	It is highly prioritized as it is the core of PrismArch developments	Yes	-
Multiplaying capability	It is highly prioritized because it will allow team members to enter the system	Yes	-
Low Priority			
Progress tools (Asana)	It is low prioritized because it was not foreseen in GA, nor necessary for VR editing tools. It is a peripheral software.	No	Asana can be incorporated through a Web browser widget inside Unreal but its usability will be limited.
Communication tools (Zoom, Skype)	It is low prioritized as it was not foreseen in the GA and there is no library for integrating Zoom or Skype inside Unreal Engine. Also, it is not directly related to VR tools.	No	There is an existing audio communication already supported by Unreal (Epic Online Services) or Vivox. There is also a library for integrating "Discord Rich Presence" however its communication functionalities are limited and it is an experimental software.
Speech to text input	It is low prioritized as it is a general tool for any VR application.	No	It is a feature that is very experimental for VR and not used currently in the industry. We will examine it through Mozilla open-source technologies.

2.4 External software

PrismArch as described above is a full-fledged VR architectural design system that depends on several external components, software, libraries and tools as shown in Table 7. Many of them have a free version up to a certain quota.

Table 7: External Dependencies

Dependency	Purpose	Type	Cost / License
Speckle https://speckle.systems/	Database	Server	Free for research Apache 2.0 License
Asana asana.com	Task management	UIs and Server	Free up to certain quota (10 members)
Vivox https://developer.vivox.com/	Teleconferencing	Service	Free up to certain quota (5000 players) / 45' free
Epic Online Services https://dev.epicgames.com/	Multiplaying	Service	Free
Easy eMail Client https://www.unrealengine.com/marketplace/en-US/product/easy-email	Mailing	User interface	20 euros
Advanced VR framework https://humancodeable.org/	VR interfaces	User interface	250 euros
Mindesk https://mindeskvr.com/	Live Link communication with CAD software	Plugin	Depends on Mindesk partner policy https://mindeskvr.com/store/
Mozilla Speech to Text technologies https://github.com/mozilla/DeepSpeech	Speech Recognition	Software	Free MPL-2.0

3. System Architectural Design

3.1 Masterplan

The proposed system architectural design concept is assembled according to the notions of *Asynchronous and Synchronous Collaboration* for 3-dimensional data creation, simulation, and annotation, as it was first introduced in D4.1. In Figure 11, we are depicting this concept. The horizontal dimension represents the *Asynchronous Collaboration* that allows any changes on the AEC project to be stored persistently into a database. The vertical dimension stands for *Synchronous Collaboration* across VR users and non-VR users that allows for real-time collaborative design and conflict resolution. More details for each modality are found in the following.

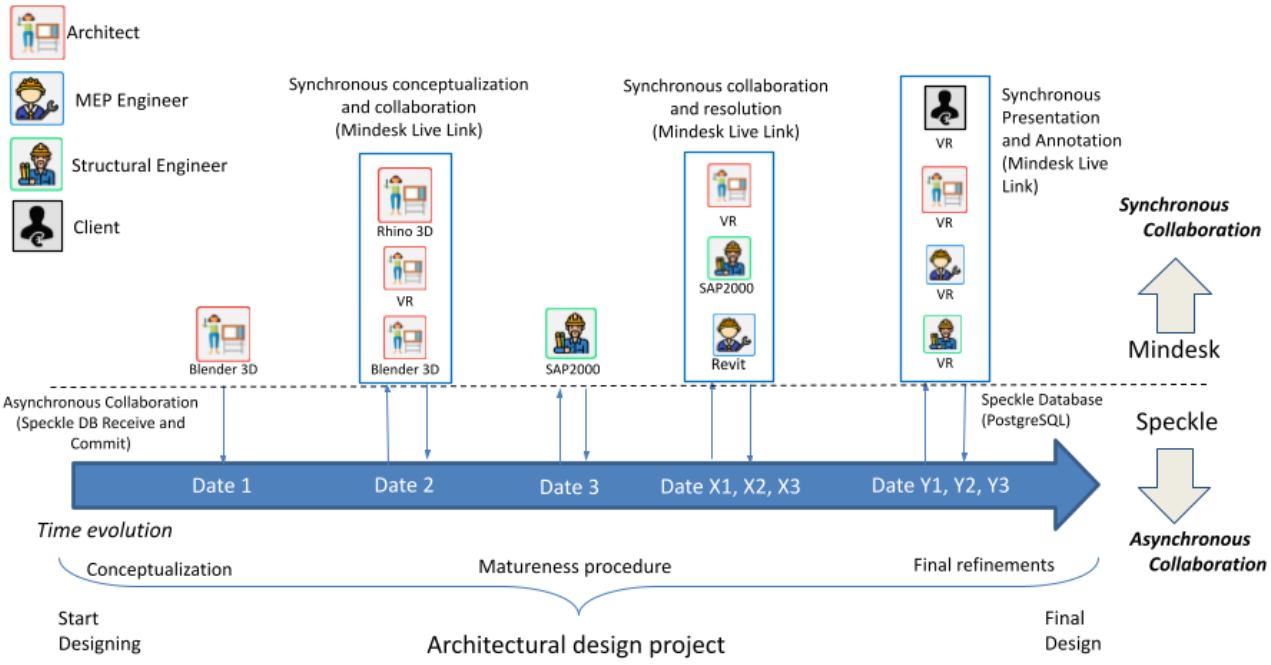


Figure 11: PrismArch Methodology - Merge Asynchronous with Synchronous collaboration.

Asynchronous Collaboration: The Asynchronous Collaboration can be used for storing the changes in the database. In this modality, only one person at a time can push the changes to the database. This is envisaged to happen in the following scenarios:

a) In an early stage (Date 1), when an Architect is formulating the concept of the AEC project in a drafting software like Blender 3D.

b) later in Date 2, when multiple Architects are contributing on the initial draft design using Blender, Rhino, and the PrismArch VR environment in Unreal. They are collaborating in real-time and the principal designer has to commit the changes into the database using the Asynchronous Collaboration plugin.

c) In a later stage of the project (Date, X1, X2, X3), where also other disciplines are involved. A MEP Engineer can commit changes into the database from Revit by working collaboratively with Structural Engineers in SAP2000, and with Architects in VR.

d) Also, in other cases (Date Y1, Y2, Y3), all disciplines can work collaboratively inside VR mainly for resolving conflicts and one of them can push the final design into the database.

In all scenarios, it can be seen that it is possible to commit changes into the database from any type of software for each stage of the project, i.e. Blender 3D for drafting, Rhino 3D for the architectural design, SAP2000 for structural simulations executions, Revit for committing Engineering plans, and VR for collaborative design and conflict resolution. The VR environment is the contribution of the PrismArch project. Asynchronous collaboration should be also integrated with Asynchronous communication tools such as mail clients in order for the users to receive a notification when a commit has been pushed to the system in order to be pulled. However, the main issue is the merging of changes that happen in parallel. We consider that it can be treated only with Synchronous collaboration in VR and with advanced semantics tools that fetch information that PrismArch semantic service proposes. More details about merging of commits can be found in Section 3.3. For the implementation of Asynchronous collaboration, there are currently two systems that could be possibly used, namely Speckle by Speckle Systems and Omniverse by NVidia (see D4.1 for more details about each software) [Omniverse]. We have preferred to use Speckle software as AKTII has experience on writing interconnection software for its Re.AKT components, and due to Speckle's longer history in the field of Architecture.

Synchronous collaboration: The Synchronous Collaboration modality is aiming into real-time collaboration for 3D design, simulation, and annotation among users. This can be experts of the same discipline, experts across disciplines, or between experts and the clients. Synchronous Collaboration helps into coordinating the effort when all designers are present. Each expert is entering the system through his or her software such as Blender or Rhino for Architects, Revit for Engineers, SAP2000 or Sofistik for Structural Engineers, and for designers that are accustomed with VR, they can enter with the VR environment (made by PrismArch) with or without using VR glasses. The Clients can also use the VR environment for inspecting, annotating, and approving the project. Zoom, Skype, or other Synchronous Communication tools can be used by users. Such synchronous communication tools applications can be passed inside VR through virtual desktops that allow windows to be transferred in VR, e.g. [OVR toolkit] and [vrDesktop].

For the implementation of Synchronous Collaboration, there are several commercial tools that can be used as reviewed in D4.1, namely Arkio, Fuzor, Gravity Sketch, Holodeck, IrisVR-The Wild, LumenRT and Mindesk. We have decided to use Mindesk software that allows real-time data transfer across Rhino, Revit, and Unreal, and due to the fact that Mindesk is a partner in the project. The developments in PrismArch will be done for bidirectional communication across software. This is not fully operational for the time being for all 3D objects and directions using Mindesk, e.g. Unreal to Rhino and to Revit is supported only for geometries but not for textures, also Mindesk does not provide interfaces for Unreal but only for a custom VR renderer in Rhino 3D which is not useful for building an enhanced VR environment with great photorealism. These all issues are currently delta developments, i.e. ongoing work inside the PrismArch project. More details about what Mindesk can currently achieve, and on what it works on, in order to achieve the interconnection of software can be found in D4.1.

3.2 Data-flow design

The data flow diagram for the PrismArch overall system is shown in Figure 12. The difference between Figure 11 and Figure 12 is that time context is lost in Figure 2 in order to represent the technical details in a greater depth. One the left-hand side there is the widely used “onscreen” 3D software that constitutes the existing AS-IS scenario, whereas on the right hand-side is the proposed TO-BE scenario that extends the AS-IS scenario with the PrismArch VR environment and services. It is seen that the center of the system is the Database (PostgreSQL by Speckle Server software) where all the design edits can be appended through commits. The communication with the database is achieved with REST HTTP through GraphQL API (black lines).

The real-time data transfer is achieved with Mindesk real-time transfer technologies such as memory access and Live-Link (blue lines). The data transfer across VR collaborators is achieved through VR technologies, e.g. Epic Online Services, Unreal peer-to-peer multiuser communication, and Vivox audio communication (red lines). Experts may use either the left-hand or right-hand software. We hope that with this methodology we will engage users to gradually use the proposed VR environment.

The scenario for the flow of data is as follows. In the left-hand mid-side, there are the Architects 1 and 2 that are using Blender to draft the design concept. Through the Speckle plugin, they collaborate asynchronously and transfer their design in the database. The design is in Blender geometries which are transformed into Speckle Objects on the client side (before submission). Next, at left-hand side bottom, Architect 2, Architect 3, and other disciplines in Rhino can fetch the concept design and collaboratively make edits synchronously through the Mindesk plugin and commit the changes by the Speckle Rhino plugin to the Database. The design is transformed from Rhino geometries into Speckle Objects also on client side (before its submission). On the left-hand upper-side, the Structural Engineers can pull the design from Database and make simulations using SAP2000 or Grasshopper with Karamba plugin. Re.AKT plugin can be used for the communication of SAP2000 with Database, or Speckle plugin can be used for pushing Grasshopper simulation geometries to the Speckle Database. Mindesk plugin can be used for collaboration across Grasshopper users. MEP Engineers at left-hand top side, can use Revit and Speckle Plugin to fetch the design from the database and insert MEP 3D plans. They can also work synchronously through the Mindesk plugin and submit changes to Database through Speckle plugin.

On the right-hand side, the VR technologies can be found. Architect 4 can be immersed into the VR environment and fetch the latest design through the Speckle plugin for Unreal. PrismArch VR UI will be used for the edits. The Mindesk plugin can be used for Synchronous Collaboration between onscreen and VR users, i.e. for interconnecting Unreal with Revit, Rhino, and Grasshopper users. Unreal multiplaying capabilities are used for collaborating synchronously with other VR users. HTTP services offered by the PrismArch Knowledge Base service can be used for interrogating the Database for semantic information across commits, e.g. “fetch all users that modified this asset”. In PrismArch, we have implemented the Knowledge base using GraphDB and RDF triplets [GraphDB, RDF]. A Tomcat server is used for the communication of a java service that transfers the data through REST protocol (POSTs) [Tomcat Server].

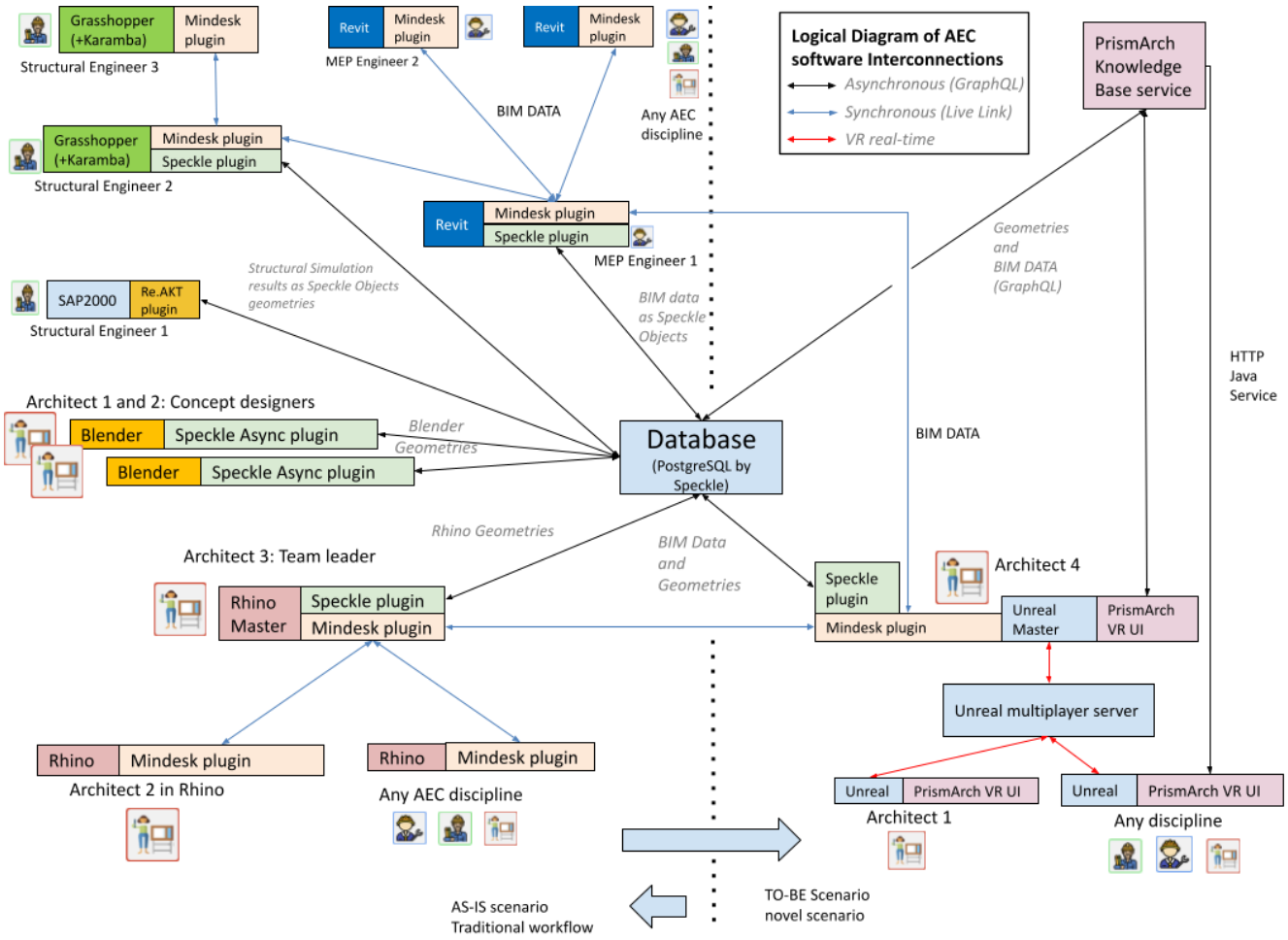


Figure 12: Data-flow path explanation.

3.3 Merging changes done Asynchronously

The Asynchronous Collaboration modality using the Speckle Database may result in changes (commits) done in parallel, i.e. in two different branches, and they have to be merged. In Figure 13, we present this situation. Merging is a challenging issue where a single user has to merge commits done by many different disciplines. For example, in the main branch, an Architect named as Architect 1 may commit in the Database an initial plan through Blender. Another Architect, named as Architect 2 may further evolve the design but in the same time Architect 1 can commit more changes. Another Architect, named as Architect 3 should decide which changes to keep among these two branches. Currently, Speckle supports the merging of changes through Grasshopper Visual Editor which allows to pick which objects to keep from two commits by its visual programming interface. However, this kind of merging is too difficult to accomplish when the disciplines involved are too many, and the history of commits is more complicated, as shown in the branches shown in the right part of the image. The MEP Engineers can not merge the commits done by Architects and Structural Engineers because they do not have the knowledge to do this.

In PrismArch we propose the merging of commits to be done in VR due to the highly collaborative nature of VR and due to the Knowledge Database tool, that allows to pose queries in the database and retrieve crucial information from all commits collectively. This is depicted in Figure 14. For example,

when the users should have to merge their changes, they can arrange a meeting in the VR environment, and perform queries on the 3D design according to the pillars “Who, What, When”. This is envisioned in the mockups of Deliverables D1.1 and D6.1, namely how to interrogate the model with proper interfaces. Several queries are formulated in the Knowledge Database service in order to cope with this requirement. The designers have the ability to view metadata about commits and decide based on the provided information.

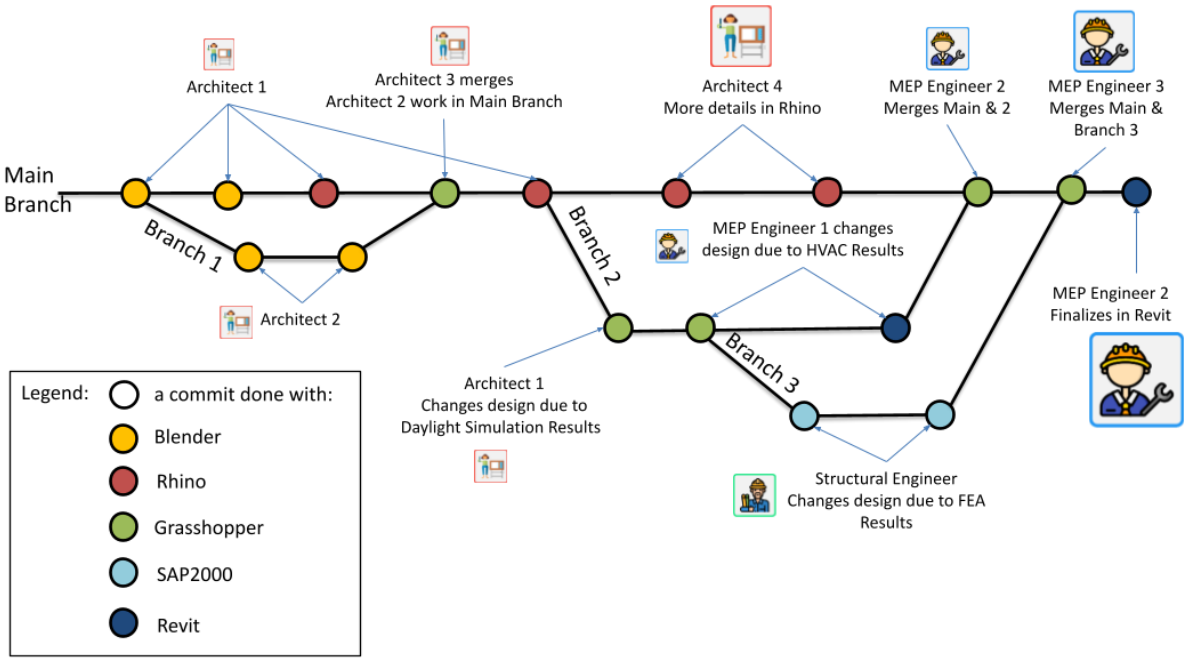


Figure 13: The Asynchronous Collaboration may result in designs that are done in parallel and have to be merged.

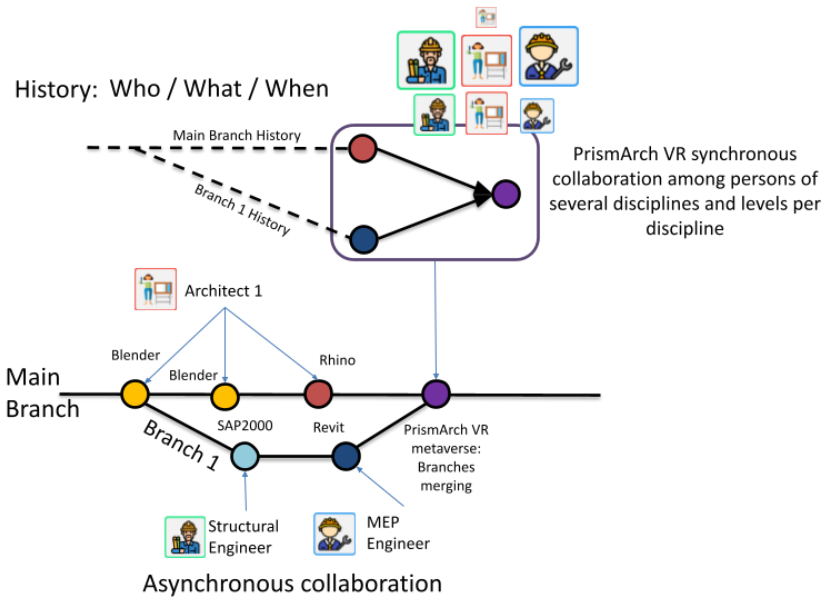


Figure 14: Synchronous collaboration for merging commits that are done in Asynchronous collaboration threads. The VR environment will provide methodologies through the semantic service (Knowledge Database) to retrieve data collectively from past commits, e.g. information about who did the change, what has changed, and when has changed across commits.

3.4 Implementation diagram

In this section we provide more details about the technologies that will be used for each service of the platform. In Figure 15, a masterplan is provided. A timeplan for the developments is presented in the Appendix. The architecture is divided into 5 major parts:

1. **The Data server:** It is the place where information is preserved;
2. **Knowledge server:** It is the place where information is processed semantically;
3. **Supplementary servers:** It is multiple servers that help towards implementing supplementary coordination and communication for VR environment;
4. **Real-time Data server:** It is the server that allows the 3D designs to pass bidirectionally across software.
5. **The "OnScreen" AEC Software:** The currently used software in AEC industry
6. **The VR environment:** The major VR environment proposed in PrismArch.

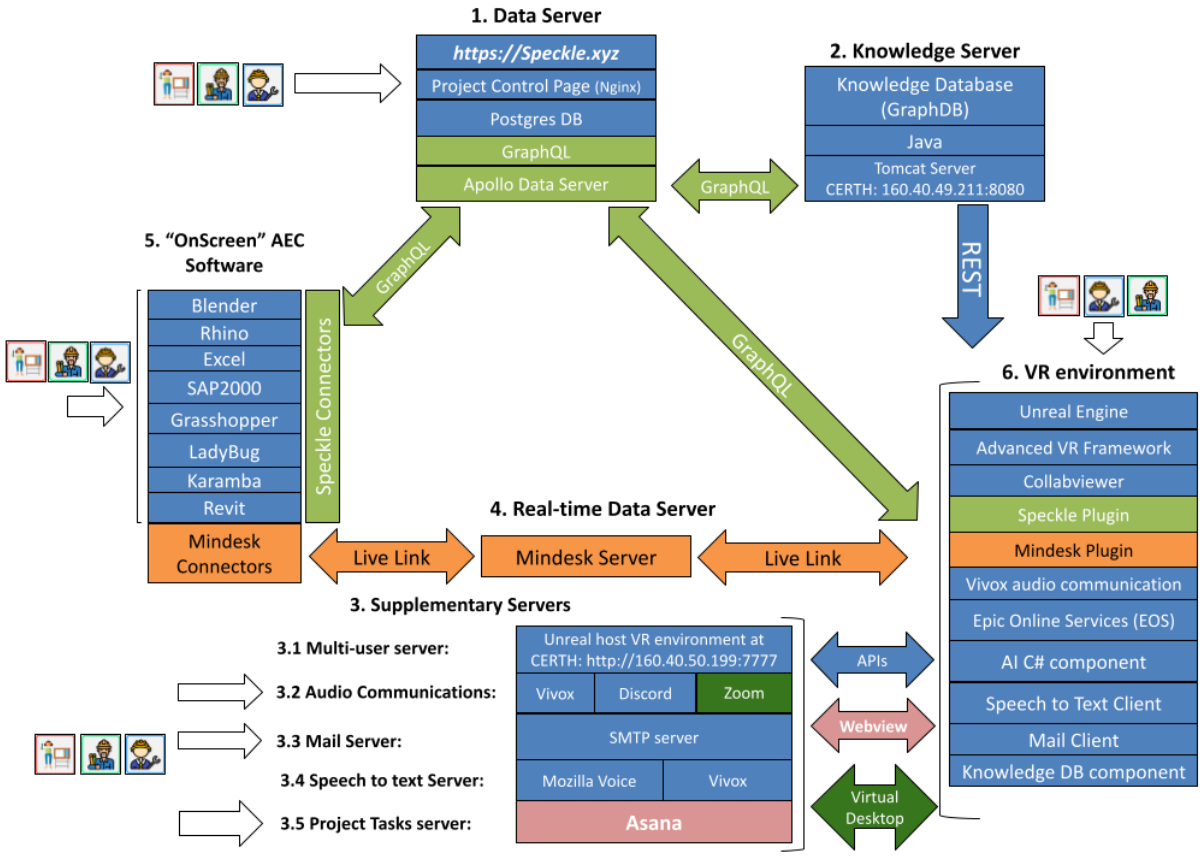


Figure 15: Implementation diagram.

1. **Data Server**: The data server is the one provided by Speckle. It can be found at https://Speckle.xyz address. It is located in the UK and has a privacy webpage (https://speckle.systems/privacy/). As PrismArch does not have a task for the maintenance of a server for preserving data, it is agreed from the partners to submit imaginary AEC data to the Speckle server. In case of real AEC data, i.e. for buildings that already exist, Speckle Systems has offered to make an instance of the Server at CERTH premises in case CERTH can not instal it.

2. **Knowledge Server:** The data is transferred from Data Server to Knowledge Server through the GraphQL API of Speckle. It is stored as RDF triples in a GraphDB server. Then a Java HTTP REST service allows the VR environment to pose POST queries in the Knowledge Database and retrieve JSON information. The Java service is provided through a Tomcat. The server and the database for the Knowledge Server are hosted in CERTH at http://160.40.49.211:8080

3. **Supplementary Servers**

   3.1 Multiplaying Server: The multiplayer server is hosted at CERTH in http://160.40.50.199:7777 . It is an executable that serves as a host in a Peer-to-peer multiplaying schema. In the case that connection speed is not adequate among partners, we will use Epic Online Services technologies, in order to have a more robust solution.

   3.2 Audio communication Server: In the current phase the multiplaying server is also a server for audio communication. However, as the quality of the audio is not good, we aim to use a Vivox server which can host a session with up to 5000 users for free. Vivox is a major player for chatting in multiplaying games. Alternatives are Discord as it offers an SDK that can be integrated in VR environments and Zoom which can be called inside VR with a Virtual Desktop application.

   3.3 Mail server: The users can set up the VR environment so that they can use their company mail server to send and receive mail.

   3.4 Speech to text server: We aim to train a deep learning network to recognize numbers and simple commands. The Mozilla Common Voice provides data and software for training the algorithm [Mozilla Voice]. A server in CERTH will be set up for providing this service. A mitigation plan is to use the Vivox services which aims to release such a component.

   3.5. Project tasks server: It is a service for project management. For the integration, we will use the Asana web tool [Asana], which can be called through a Web browser widget that is offered by Unreal Engine.

4. **Real-time data server:** It is a server that allows the transfer of 3D CAD geometries across CAD runtime and the PrismArch VR environment (Unreal Engine). The server (Mindesk Core) acts as a hub for the CAD data and I/O data (including geometry, metadata, headset position, switch status, etc.) and operates through an internal set of API that connects the Core to each Link (CAD plugin) each time a session is launched. Mindesk can deploy the Core server locally or remotely on a machine determined for operations within the PrismArch project.

5. **The "On-Screen" AEC software**
   - **Blender:** It will be incorporated in the pipeline through a Speckle connector.
   - **Rhino, Revit, Grasshopper, LadyBug, Karamba:** They will be incorporated in the pipeline through Speckle connectors and Mindesk plugin.
   - **Excel**: It will be incorporated in the pipeline through a Speckle connector.
   - **SAP2000:** It will be incorporated in the pipeline through a connector provided by AKT (Deliverable D4.1, Re.AKT component).

6. **The VR environment:** It is the main contribution of PrismArch and it consists of several components.

    a. Unreal Engine: Unreal Engine 4 is the basis for the VR system. It is provided for free for non-profit purposes and it has a certain policy for for-profit companies such as a share of 5% of the profits when the product exceeds 1 million USD (https://www.unrealengine.com/en-US/faq).

    b. Advanced VR framework: It is a collection of tools that allows to make appealing and easy to use UI for VR. It provides a tree shaped visualization for the menu, a VR smartwatch, an EOSLink component for connecting with Epic online services, and a template for Architecture Visualizations as demonstrated in Section 2 [AVRF].

    c. Collabviewer template by Unreal: It is a template offered by Unreal especially designed for the AEC industry. It allows multi-user immersion, multiple wandering modes such as walk or fly, and some other tools such as measurement and xray [CollabViewer Template].

    d. Speckle plugin for Unreal: Speckle plugin for allowing the bidirectional flow of data between Speckle Database and Unreal Engine [Speckle]. The Unreal plugin of Speckle is forked by CERTH in order to further develop the pushing of data as current version supports only pulling of data in Unreal. Also UIs are made for manipulating Speckle in VR as it does not have any.

    e. **Mindesk plugin for Unreal:** It is the plugin that allows real-time data transfer between the VR environment and the Real-time data server. It is being developed by Mindesk in order to cover all types of geometries, textures for bidirectional communication.

    f. Vivox audio communication: It is a high quality audio communication plugin in Unreal with promised Speech-to-text capability in the upcoming months.

    g. Epic Online Services Link: It is a component in Unreal that allows the VR environment to communicate with [Epic Online Services]. The services are divided into

        i. Game Services such as Multiplaying, Lobbies, Game Analytics, Voice, Player Data Storage, Statistics, Achievements and

        ii. Account Services such as single identity for Login, Friends, Presence, and Invites.

    From all these services, the Multiplaying service will be checked as a better alternative from peer-to-peer multiplaying service and the voice service will be checked as an alternative to peer-to-peer audio communication.

    h. Artificial Intelligence C# component: It is the component provided by WP3 that will be translated into C++ with [CLR] and incorporated into the VR environment.

i. **Speech-to-text client:** It is a component that will allow the users to send voice data to the server for voice recognition and perform a command in the UI. A REST API will be called.

j. **Mail client:** It is a component for viewing/composing mails inside VR, connected with the mail server of each company. We will use the “Email Plugin” for Unreal [Email UE4 plugin].

k. **Knowledge DB component:** It is a collection of functions that allow the semantic service through the REST protocol. The **VaRest** plugin for Unreal will be used for this purpose [VaRest].

4. Integration and verification methodology

4.1 Scope, goals, and the verification model

This chapter summarizes the integration and validation plan that is followed by the PrismArch consortium during the development of the VR application as well as the development and integration of web-based services. During the PrismArch project two integration cycles have been envisioned that lead to the first and second prototypes. It is performed in a highly complex distributed environment with multiple companies developing and deploying reusable and integrated services. The described strategy proved to provide the required quality assurance for the development and deployment of the integrated VR solution for all the AEC disciplines. Future developers can reuse this strategy when extending this VR solution.

The scope of the integration methodology is to describe how the VR solution is verified ("Are we building the product right?"). The goals of this methodology are:

- To ensure that the developed VR solutions covers the needs of the three AEC disciplines involved that guided its design and development;
- To ensure that the VR solution work as expected after the tests described have been executed;
- To minimize the efforts in integrating the partners' different components (by eliminating errors in the components in an early stage);
- To align the different partners in the testing process to gain the necessary quality level in the developed software;
- To describe the tests such that after successful testing the software satisfies the needs of all stakeholders.

Verification of the VR solution is performed using the V-model displayed in Figure 16. The V-model is a simple variant of the traditional waterfall model of software development with an emphasis on the verification and validation of the software [V-model]. The V-model identifies different testing activities or phases in which the deliverables of the associated design phases are analysed or tested. The horizontal axis represents time and project completeness, while the vertical axis represents the level of abstraction. Following the V-model, software development starts with describing the use cases and defining the requirements. The requirements lead to a high-level design or architecture. The different system components within this architecture are further elaborated in detailed technical designs. Based on the technical designs the developers start coding, the lowest point on the V-model.

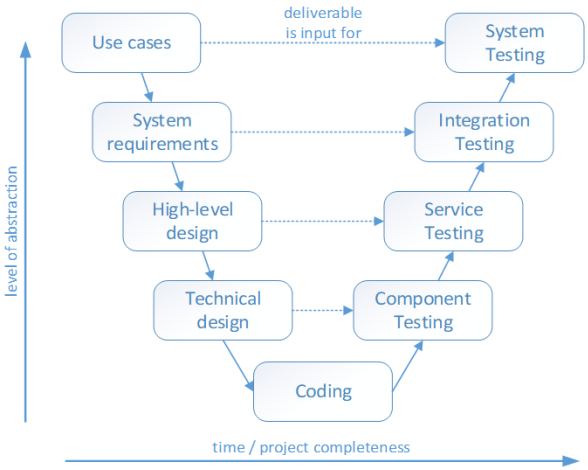


Figure 16: The V-model for integration and verification.

The PrismArch project's result is a prototype consisting of a VR application and several supplementary services offered by partners and by third parties. During the project two prototype testings are performed for two architectural projects, the residential Villa and the commercial Tower (D1.1). These testings will provide feedback on both the usage of the VR solution itself and on the peripheral assisting services. This feedback is used to improve the VR solution, and the services for the second prototype testing phase. In preparation of each phase, the V-model as a whole is executed once, while some parts of the V-model are executed multiple times, e.g. when retesting specific components.

4.2 Test Phases

Table 8 summarizes the two testing-phases, responsible partners and deliverables in a single development cycle.

Table 8: List of test phases

Test phase	Subject	Responsible WP/Partner	Deliverables
Component Level Testing Phase	Individual Components	Developers (WP2, WP3, WP4), CERTH, UoM, Mindesk	Focus on the isolated software components, verifies these components compile and if the basic functions work as expected. All individual components are tested before they are embedded into the VR solution, e.g. UoM verifies the AI component in C# before being translated in C++ for Unreal.

Service Level Testing Phase	Individual Services	Service owners (WP4), CERTH, Mindesk	Testing the services within the PrismArch VR solution and the interaction between the components within each service. Verifies if the basic functions work as expected (functional testing) and that the service meets the performance and security requirements, e.g. the real-time collaboration in VR through Mindesk server is working.
Integration Level Testing Phase	Integrated PrismArch VR solution and services	Service owners (WP5), CERTH, Mindesk	Testing the integration of the different services with a strategy provided by each partner.
System Level Testing Phase	Integrated PrismArch VR solution	Use case partners (WP6), ZH, AKT, SWECO	Testing the VR solution against the functional requirements to validate if the system meets the key business requirements.

Table 9 summarizes which types of testing are advised in a specific test phase.

Table 9: Overview of test types required in the test phases.

Test phase	Functional	Inter-operability	Compliance	Performance	Availability	Security
Component Level Testing Phase	v	-	-	v	-	v
Service Level Testing Phase	v	-	v	v	-	v
Integration Testing Phase	v	v	v	v	-	v
System Testing Phase	v	v	-	v	v	v

4.3 Issue Reporting Cell

In the following, we provide a way to report testing results. The partners that perform tests should use these tables for reporting issues.

Software identification	
Name	[The name of the component according to Section 3]
Version(s)	[Provide a version number for the component including any used libraries versions]

Test period	
Test phase	Use one of the following • Component Level Testing Phase • Service Level Testing Phase • Integration Testing Phase • System Testing Phase
Test Types	[Functional / Interoperability / Compliance / User Acceptance]
Test Status	Test Completed
Planned test start date	XX/XX/202X
Actual test start date	XX/XX/202X
Test completion date	XX/XX/202X
Partners	[Responsible for development]
Tester(s)	[Responsible for testing]

Test environment	
Test environment	Windows, Mac Linux or other (version included, e.g. Windows 11) GPU drivers version

Test devices	Headset (HP Reverb G2, or Oculus Quest 2) Input methodology (VR controllers, gesture, speech)
Test pc's	Computer Specifications, GPU, CPU, RAM

References	
Reference	Any reference document

No.	Requirement(s) [Which requirements are tested?]	Expected behaviour	Results round 1	Results round 2
1	[requirement tested]			
2	[another requirement tested in parallel]			
3	...			

Issue No.	[The unique issue number]
Scenario ID	[Low / Medium / High]
Severity	[Low / Medium / High]
Type	[Bug / Change request]
Summary	[One line summary of the issue]
Description	[Description of the issue, please give enough information to reproduce the issue]
Workaround	[If there is a workaround that mitigates the issue then give it here]
Recommendations	[Recommendation regarding this issue]
Screenshots	[Screenshot relevant for issue]

References

[Asana] Asana, Online Project Management System, URL: https://asana.com/

[AVRF] HumanCodeable, Advanced VR Framework, URL: https://humancodeable.org/

[AVRF Vivox Core plugin] URL: Advanced VR Framework Vivox Core plugin [https://dev.humancodeable.org/our-services-2/advanced-framework-utilities/].

[CLR] C# to C++, Common Language Runtime embedding for Unreal, https://github.com/nxrighthere/UnrealCLR

[CollabViewer Unreal template] Collab Viewer Template, URL: https://docs.unrealengine.com/4.26/en-US/Resources/Templates/CollabViewer/

[Discord] Discord Platform, Voice and chatting for community building, URL: http://discord.com

[Discord SDK] Discord SDK, URL: https://discord.com/developers/docs/game-sdk/sdk-starter-guide

[Discord-UE4 integration software] Discord-UE4 integration software, URL: https://github.com/ryanjon2040/Discord-UE4#how-to

[Email UE4 plugin] Email Plugin, URL: https://www.unrealengine.com/marketplace/en-US/product/88f228c5ad9d4288b6426754c0b3f3d1

[Epic Online Services] EPIC, online services, URL: https://dev.epicgames.com/en-US/home?sessionInvalidated=true

[GraphDB] GraphDB - An enterprise ready Semantic Graph Database, compliant with W3C Standards https://graphdb.ontotext.com/

[Mozilla Voice] Mozilla Speech Open Source Technologies, URL: https://research.mozilla.org/machine-learning/

[Omniverse] NVidia, Omniverse: Real-time simulation and collaboration platform, URL: https://developer.nvidia.com/nvidia-omniverse-platform

[OVR tools] Virtual Desktop: View desktop within VR, URL: https://store.steampowered.com/app/1068820/OVR_Toolkit/

[RDF] Resource Description Framework (RDF), URL: https://www.w3.org/RDF/

[Speckle Server] Speckle Developer Documentation, URL: https://speckle.guide/dev/

[Speckle GraphQL API] Query data coming from a variety of sources with Speckle's API, URL: https://speckle.systems/developers/apis/

[Tomcat Server] Apache Tomcat Server, URL: http://tomcat.apache.org/

[Unreal Engine CrossPlatform] Sharing and Releasing Projects, URL: https://docs.unrealengine.com/4.26/en-US/SharingAndReleasing/

[VaRest] Pushkin Studio, REST API plugin for Unreal Engine 4, URL: https://github.com/ufna/VaRest

[Vivox] Vivox Platform, In game voice and text chat, URL: https://en.wikipedia.org/wiki/Vivox

[Vivox Core Plugin] URL: https://www.unrealengine.com/marketplace/en-US/product/vivoxcore

[vrDesktop] VR desktop - Your PC in VR: URL: https://www.vrdesktop.net/

[V-model] V-Model (software development), https://en.wikipedia.org/wiki/V-Model_(software_development)

Appendix

In the following we present a diagram about the timeline of the developments of each component.

Table A.1: Gantt Chart for the developments

			Year 1						Year 2									
			M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20	M21	M22
Thread	**Developments Task Force**	**Thread Owner**	**May**	**Jun**	**Jul**	**Aug**	**Sep**	**Okt**	**Nov**	**Dec**	**Jan**	**Feb**	**Mar**	**Apr**	**May**	**Jun**	**Jul**	**Aug**
A	Setup tools	CERTH																
B	Progress tools	CERTH																
C	**Cross-Communication tools**																	
C.1	Synchronous (Vivox, Zoom, Discord, EOServices)	CERTH																
C.2	Asynchronous (Mail, Discord)	CERTH																
D	**Input methods (Speech Recognition)**	**CERTH**																
E	**Design Tools**																	
	Orientation and Space alteration tools																	
E.1	Spatial Orientation tool	CERTH																
E.2	Toggle View Mode	CERTH																
E.3	Toggle Camera Perspective	CERTH																
E.4	Clipping Plane tool	CERTH																
	Information tools																	
E.5	Commenting / Markup tool	CERTH																
E.6	WhiteBoard tool:	CERTH																
E.7	Tagging Tool	CERTH																
E.8	Query Tool (Knowledge Base tools (1. Develop queries, 2. Develop REST API)	CERTH																
	Design Core Tools																	
E.9	DToolbox	MINDESK																

E.10	Design Support and Evaluation tool	CERTH															
E.11	AI tools	UoM															
E.12	Multi-Selection tools	CERTH															
F	**Infrastructure Core development**																
F.1	Speckle Unreal plugin	CERTH															
F.2	Mindesk Server Setup	Mindesk															
F.3	Mindesk VR tools	Mindesk															
F.4	Multiplaying services	CERTH															
F.5	UX Mockups (VR)	CERTH - ZH															
F.6	Integration	CERTH															