# MindSpaces

Art-driven adaptive outdoors and indoors design

H2020- 825079

# D6.3

# 1st Prototype and VR Development

| | |
|---|---|
| **Dissemination level:** | Public |
| **Contractual date of delivery:** | Month 19, 30.06.2020 |
| **Actual date of delivery:** | Month 20, 06.07.2020 |
| **Workpackage:** | WP6 |
| **Task:** | T6.4, System Integration |
| **Type:** | Report |
| **Approval Status:** | FInal |
| **Version:** | 1.0 |
| **Number of pages:** | 67 |
| **Filename:** | D6.3 -MindSpaces_v1.0.docx |

**Abstract**

The deliverable presents the prototypes of the various components of MindSpaces platform as well as the UI/UX of the frontend tools. The tool also describes the technical infrastructure of the components and the demonstration of the tools. The Operational Prototype will be the scaffolding on which the platform will be built iteratively, adding functionality and depth on top of the dummy-based setup which marks this first milestone

at its sole risk and liability.

co-funded by the European Union

# History

| Version | Date | Reason | Revised by |
|---|---|---|---|
| V0.1 | 20.05.2020 | ToC | Yash Shekhawat |
| V0.2 | 15.06.2020 | Partner inputs | All partners |
| V1.0 | 30.06.2020 | Final Version | Yash Shekhawat |

# Author list

| Organization | Name | Contact Information |
|---|---|---|
| Nurogames | Yash Shekhawat | **Yash.shekhawat@nurogames.com** |
| Nurogames | Sebastian Freitag | **Sebastian.Freitag@nurogames.com** |
| Nurogames | Robin Kratz | **Robin.kratz@nurogames.com** |
| McNeel | Ayman Moghnieh | **aymanmoghnieh@gmail.com** |
| UPF | Alexander Shvets | **Alexander.shvets@upf.edu** |
| up2metric | Christos Stentoumis | **christos@up2metric.com** |
| up2metric | Ilias Kalisperakis | **ilias@up2metric.com** |
| CERTH | Sotiris Diplaris | **diplaris@iti.gr** |
| CERTH | Stefanos Vrochidis | **stefanos@iti.gr** |
| ZH | Tyson Hosmer | **Tyson.hosmer@zaha-hadid.com** |
| CERTH | Nefeli Georgakopoulou | **nefeli.valeria@iti.gr** |
| CERTH | Petros Alvanitopoulos | **palvanit@iti.gr** |

# Executive Summary

D6.3 presents the current state of the MindSpaces platform and its components. The preliminary implementation is geared to consolidating the integration model of the platform, and establish its processing cycle, as well as to provide a proof-of-concept for each envisioned technology demonstration of the envisioned platform by means of an operational prototype. The prototype shows a rough sketch of the User Interface/User Experience (UI/UX) and dummy implementations of the functionalities of the system. The deliverable provides the overview of the system components along with their current functionalities.

The deliverable discusses eh whole system architecture along with the deployment model used for the system prototypes to be deployed and running the pipelines defined in D6.2. The deliverable expands on each of the component's Data Flow to showing the inputs and outputs of each of the component.

Finally, the deliverable outlines the video demonstrations of the front-end tools of the system, specifically the design tool, the ABPS tool, The VR Tool.

# Abbreviations and Acronyms

| | |
|---|---|
| **VR** | Virtual Reality |
| **ABPS** | Agent Based Parametric Semiology |
| **3D** | Three Dimensional |
| **EEG** | Electroencephalogram |
| **GRPC** | Google Remote Procedure Calls |
| **EA** | Emotional Analysis |
| **GUI** | Graphical User Interface |
| **TR** | Technical REquirement |
| **UR** | User Requirement |
| **API** | Application Programming Interface |
| **CAD** | Computer Aided Design |

Table of Figures

# Table of Contents

## Table of Contents

# 1 INTRODUCTION

MindSpaces project aims to create a new way to urban and architectural design using the physiological psychology and by generating 3D-VR immersive and emotion-adaptive “neuro-environments”. The MindSpaces project helps architects use the knowledge of artists and citizens to help in the design process with physiological inputs for outdoor environments, indoor environments and inspiring workplaces. The project will demonstrate the technology in 3 Pilot Use Cases:

- Design of Outdoor urban environments
- Inspiring workplaces
- Emotionally-sensitive functional interior design

The deliverable documents the operation prototype of the system and its components that were developed based on the requirements gathered and documented in D6.2. In D6.2, we focus on using the requirements to develop the system architecture as well as the basic architecture of the components of the system. The deliverable also mentioned the deployment and the data flow of the entire system.

The purpose of this document is to provide the technological advancements doen in terms of the implementation of the services and components that were envisioned in the first phase. The document also provides the first glance at the system UI/UX for designers/architects as well as the end user in VR.

Section 2 introduces the general architecture of the platform, the updates from D6.2 and the machine based deployment. The section also presents the components and services of the current state of each of the service.

Section 3 contains the liks and details for the demonstration videos of the frontend tools for the platform.

Section 4 presents a brief summary and conclusions.

# 2 ARCHITECTURE OF THE FIRST PROTOTYPE

In the following chapter, we define the general architecture of the entire MindSpaces platform, discussing the concept behind each of the component and the entire platform.

Section 2.1 describes the conceptual design of the platform, then a general definition of the MindSpaces components and architecture is elaborated with the updates from the previous deliverable.

Section 2.2, we discuss MindSpaces components and services along with the Technical requirements and development plans, input and output data samples along with the prototypes/screenshots of the tool

In Section 2.3, we discuss the data storage system of the platform. The platform consists of 2 major storages, A file system, SOLR storage and a MongoDB.

In Section 2.4, We discuss the MindSpaces Design Tool along with the concept behind it, the requirements it addresses, the technological framework and the functionalities implemented.

Section 2.5 discusses the VR Tool that will be used by the platform to experiment with the end users in conjunction with the EEG device.

## 2.1 General Architecture

MindSpaces architecture follows a distributed architecture. Each service is designed and developed to adhere to the communication model and the input output parameters defined in the architecture based on the role of the component. Each service in the platform communicates with other services/components using a GRPC client inside the service containing the information about other services to be able to reach them and send data using the predefined proto files.

Each component is developed independently so that it can operate with a specific set of inputs received by it either by other components or 3$^{rd}$ party services and it can provide a specific set of outputs for other tools in the system.

The final architecture for the prototype includes the integration of services into pipelines that can be deployed actively and passively to run the entire system. The architecture diagram, show in Figure X shows the connections between the different services of the platform.

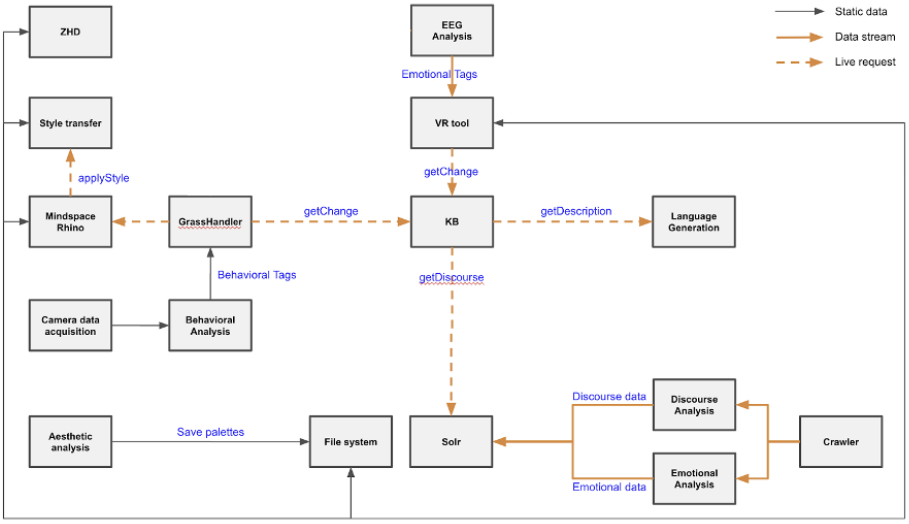



Figure 1 Platform Architecture

The architecture diagram also shows the different messages that are bring transmitted as inputs and outputs between different components of the platform.

As defined in previous deliverable, a machine based architecture is done for deployment, consisting of 4 machines:

1. Design Machine: The design machine groups the design tools and components available for the artist or architect. It is mainly composed of three components: the Mindspaces Rhino plugin, the GrassHandler (GH) service, and the ABPS Generative Design and Behavioural Simulation Tool.
2. VR-EEG Machine: This machine is dedicated to supporting EEG experimentation in virtual reality (VR). Two hardware components will be simultaneously worn by subjects: one EEG device that will capture brain signals, and one VR device that will show a virtual scene to be used in the experiment along with the configuration changes.
3. Cloud Machine: The cloud machine consists of multiple machines on the cloud that work in conjunction with each other and managed mostly by the Knowledge Base and supported with the data storage.
4. Behavioural Analysis Machine: Behavioural Analysis Machine works to analyse the video and work on creating the output of the behavioural data.

Figure 2 shows the Machine-to-Machine communication

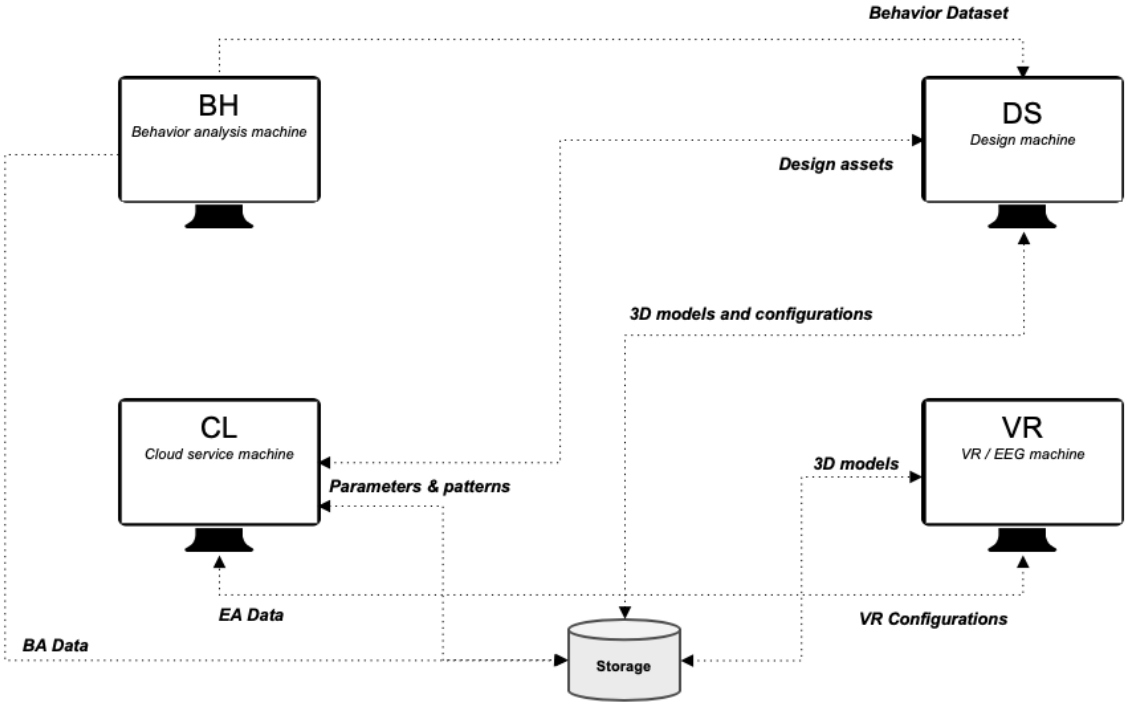



Figure 2 Machine based architecture

The architecture has not changed much from the planned architecture in D6.2. Although the design of the components have changes along with the design of the messages that each of the component will send and receive. The next sub-section elaborates on the various components and services in details.

## 2.2 MindSpaces components and services

### 2.2.1 Emotional Analysis

| **Name of the component: Emotional Analysis** |
|---|
| **Description of the component** |
| Emotional Analysis (EA) component is a service responsible for emotional state detection and recognition from physiological signals. This service will receive physiological signals as input, e.g., electroencephalogram (EEG), from lightweight recording devices (e.g., Enobio) and it will predict an emotional state as output. More specifically, EA will acquire physiological signals in real-time, analyze them with signal processing and machine learning techniques and predict emotional states on the two-dimensional valence arousal space. |
| **Technical Requirements addressed** |
| **TR7. Create an emotional state recognition service in order to recognize the emotional states of users in real time.** |
| **Input Data with sample** |
| **Input data on this service are physiological signal live streams through Lab Streaming Layer (LSL) library (**https://github.com/sccn/labstreaminglayer**).** |
| **Output Data with sample** |

Output data on this service is a json object, generated every 5 seconds (only the first json object is generated 10 seconds since the beginning of the recording). This object consists of four integer values: emotional state tag, valence tag, arousal tag, timestamp (in seconds)

Output sample: { "valence": 1.0, "arousal": 1.0, "emotional_state": 1, "time": 30 }

**Screenshots**

EA service will not operate individually but in combination with VR tool. The output of EA service, the json object, will be sent to VR tool through GRPC communication. Then, VR tool will communicate with Knowledge Base (KB) and send the emotional state information along with the virtual location of the users in a new single json object. In Figure 1 you can see the output of the EA service, operating individually, as produced in a command line window. The printed messages are indicating the flow of the service, e.g., the connection to the EEG stream etc. and the results that are saved in a json file, and then addressed to the VR tool through GRPC communication. Finally, last message confirms that GRPC communication was successful.

```
Prediction time is every 5 seconds.
The number of channels is: 8
Looking for an EEG stream...
Start acquiring data
Press Ctrl-C in the console to break the while loop and stop the service.
Baseline period of 10 seconds is over
Predict the emotional state for the first 10 seconds
Predicted valence is 0
Predicted arousal is 1
Emotional state is: 2
Store results in dictionary object
Save results in a .json file
Send results to VR tool through GRPC communication
Emotion recognition client received: 1
Now predictions will be made every 5 seconds
Predict the emotional state
Predicted valence is 1
Predicted arousal is 0
Emotional state is: 4
Store results in dictionary object
Save results in a .json file
Send results to VR tool through GRPC communication
Emotion recognition client received: 1
Predict the emotional state
Predicted valence is 0
Predicted arousal is 1
Emotional state is: 2
Store results in dictionary object
Save results in a .json file
Send results to VR tool through GRPC communication
Emotion recognition client received: 1
Predict the emotional state
Predicted valence is 1
Predicted arousal is 1
Emotional state is: 1
Store results in dictionary object
Save results in a .json file
Send results to VR tool through GRPC communication
Emotion recognition client received: 1
Predict the emotional state
Predicted valence is 0
Predicted arousal is 1
Emotional state is: 2
Store results in dictionary object
Save results in a .json file
Send results to VR tool through GRPC communication
Emotion recognition client received: 1
Predict the emotional state
Predicted valence is 0
Predicted arousal is 1
Emotional state is: 2
Store results in dictionary object
Save results in a .json file
Send results to VR tool through GRPC communication
Emotion recognition client received: 1
Closing!
```

Figure 3: Output of the EA service

In Figure 2, there is a screenshot of the communication between EA service, VR tool and KB. The first message (green square box) displays the information VR tool received from EA service. The second message (yellow square box) indicates the data VR tool sent to KB and finally, the third message (blue square box) presents the design configuration, defined by KB

that will be applied in a specific hotspot. Hotspot is a specific virtual position from which the user will be experiencing MindSpaces installations inside the virtual environment.

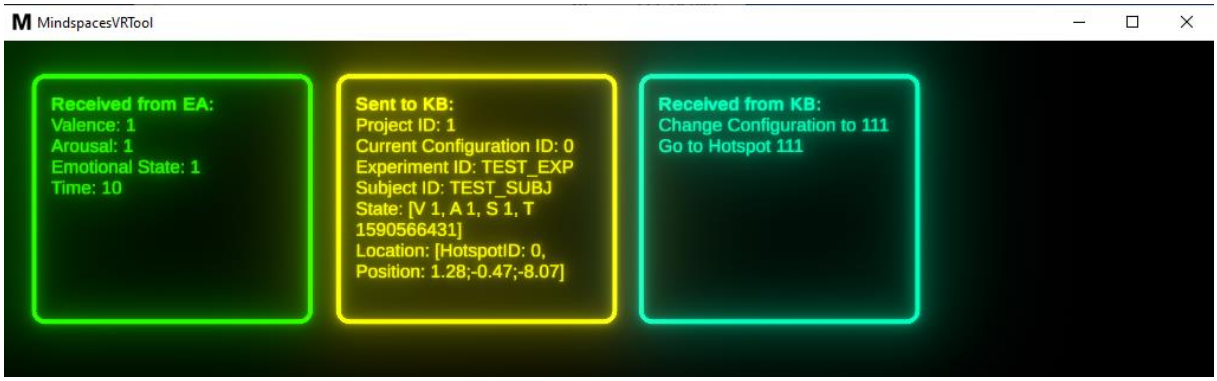



Figure 4: Communication between EA service, VR tool and KB.

**Prototypes and timelines**

The first prototype of EA service will be delivered with V1 of the MindSpaces platform, in M19 of the MindSpaces project. This first prototype is developed using data from preliminary data collection experiments and data from benchmark datasets. The second prototype of EA service will be developed after delivering V1 of MindSpaces platform and it will be using more advanced feature extraction techniques and the machine learning models will be trained with more data gathered during extensive data collection experiments within MindSpaces project.

### 2.2.2 Behavioural Analysis

| **Name of the component** |
|---|
| **Description of the component** |
| **Analyze videos to detect behavior of people in open spaces.** |
| **Technical Requirements addressed** |
| **GPU accelerated server to process the videos** |
| **Input Data with sample** |
| **Video sequence of consecutive frames.** |
| **Output Data with sample** |
| **JSON with relative locations of people and tags describing what they are doing over time.** |

**Screenshots**

There is no GUI yet to demonstrate screenshots.

### 2.2.3 **ABPS Behavioural Simulation Tool**

| **Name of the component** |
| --- |
| **Description of the component** |
| The ABPS (Agent Based Parametric Semiology) Behavioural Simulation tool is used to 1) simulate human behaviour and social interaction within and in relation to 3d virtual workplace environments and their design features, 2) record spatially located data points over time for individual and collective behaviour in relation to design features in the 3d environment, 3) visualize crowd behavioural data through spatial heatmaps and collective numeric data tables.<br><br>The tool is a cognitive agent-based behavioural model used for the simulation of people interacting within 3D virtual environments. People are modelled as agents, each with a utility-based autonomous decision-making framework triggering changes in each agent's actions such as focused working, social behaviours, pre-scheduled meetings, unscheduled spontaneous collaborations, and other workplace activities. Decisions are made for actions at any given moment in the simulation by agents weighing their internal motivational state parameters, their proximity and social relation to each other, and their proximity and relation to spatial features in the workplace environment.<br><br>Simulation output data is used to identify the social / behavioural performance of the overall model and identify zones and design features with high or low influence on the performance. The tool is integrated in the Mindspaces Platform by receiving 3d virtual models to simulate/analyse and exporting simulation analysis data. 3d virtual models are exported from 1) the Design Tool or 2) Rhino/Grasshopper or 3) generated by the ABPS Generative Design Tool. Simulation data is visualized within the ABPS Behavioural Simulation Tool and exported 1) to Json format and importable back to Grasshopper or the Design Tool. For the purpose of this integration across Rhino / Design Tool and Unity formats used by the VR Tool and the ABPS Behavioural Simulation Tool we developed the following additional component:<br><br>**Mindspaces Model Integration Component:**<br>The component is used to integrate models developed in the Design Tool or in Rhino/Grasshopper with Unity for importing them into the VR tool and the ABPS Behavioural Simulation Tool. The component has 2 parts:<br>**1) Grasshopper2Unity plugin:**<br>The Grasshopper2Unity plugin utilizes a shared Mindspaces Asset Library containing 3dm Block files with associated textures. The plugin allows users to load blocks through an interface, deploy and manipulate object instances in Grasshopper, and export them to Json format to be imported into a Unity scene. This Json format is also exported directly from the Design Tool, enabling its integration with the VR Tool and ABPS Behavioural Simulation Tool using the UnityAssetImporter.<br><br>**2) UnityAssetImporter plugin**<br>The UnityAssetImporter plugin also utilizes a parallel shared Mindspaces Asset Library containing Unity Prefab asset files with associated textures and materials. The plugin allows users to load Json files exported from the Grasshopper2Unity plugin as well as from the Design Tool and automatically deploy them as Unity Prefab assets in a Unity |

scene which can be loaded in the VR Tool or ABPS Behavioural Simulation Tool. Updates in the Json file are reflected in the Unity scene through a button press.

**Technical Requirements addressed**

ABPS Behavioral Simulation Tool addresses the following TR(s):

- TR 14. Simulating human behavior which is related to the following User Requirements: UR-8,UR_11,UR_12,UR_14,UR_15,UR_16,UR_17,UR_29,UR_30,UR_31, UR_32,UR_33,UR_34,UR_36,UR_37,UR_38, UR_61, UR_63,UR_64, UR_69,UR_79
- TR 15. Visualizing behavioral simulation data which is related to the following User Requirements:UR-8,UR_11,UR_12,UR_14,UR_15,UR_16,UR_17,UR_29,UR_30, UR_31,UR_32,UR_33,UR_34,UR_36,UR_37,UR_38,UR_61, UR_63,UR_64,UR_69,UR_79

**Input Data with sample**

Simulation Model Format: Unity scene file

The scene file can be 1) Built in Grasshopper and Exported Using Grasshopper2Unity Exporter, 2) Built and exported by the Design Tool in Json format, or 3) Built in the ABPS Generative Design Tool and exported in Json format; Json files are imported to build Unity Scene files using the UnityAssetImporter plugin. Rhino/Grasshopper export and Json file example shown below:

**Grasshopper2Unity Exporter:**
Input data: Grasshopper definition / deployed rhino 3d blocks
Sample:
(Figure 5)

**Unity UnityAssetImporter:**
Input data: Json
Sample:


```
{"meshList":[],"xformList":[{"guid":"00000000-0000-0000-0000-000000000000_1_0",
"objName":"Coffee_Table_00_4p","position":{"X":8.0568621266898059,"Y":0,"Z":3.5881048606653252},
"xDirection":{"X":1,"Y":0,"Z":0},"yDirection":{"X":0,"Y":0,"Z":1},"zDirection":{"X":0,"Y":1,"Z":0}},
{"meshList":[],"xformList":[{"guid":"00000000-0000-0000-0000-000000000000_1_0",
"objName":"Coffee_Table_00_4p","position":{"X":8.0568621266898059,"Y":0,"Z":3.5881048606653252},
"xDirection":{"X":1,"Y":0,"Z":0},"yDirection":{"X":0,"Y":0,"Z":1},"zDirection":{"X":0,"Y":1,"Z":0}}

•••
```


**Output Data with sample**

Simulation data is spatially located within the 3d model as 3d datapoints as well as tabulated as cumulative data. The primary data that is collected and visualized over the course of the simulation is described below:

**Simulation Sample Frame / Diagrams:** (Figure 6, Figure 7)

Agents are deployed based on user inputs for quantity / type of agents to autonomously interact with each other and design features of their environment.

**Actions Datapoints / HistoryMaps:** (Figure 8, Figure 9)
Each agent's current action is recorded and located as a 3d datapoint every few frames in the simulation. Specific action data and all action data can be visualized as datapoints, heatmaps, and in table format showing the % of collective time spent performing each action.

**Encounters Datapoints / EncounterMap**: (Figure 10)
Moments when two agents come within close visually perceived proximity. They influence decision making and are recorded and visualized as datapoints and heatmaps.

**Unscheduled Events - Collaborations Datapoints / CollaborationMaps:** (Figure 11)
Unscheduled conversation and collaboration events take place based on a series of utility based considerations including proximity and social relation between agents, internal state parameters of the agents, and their location in the environment relative to places for collaboration or social interaction. Collaborations and social events are recorded and located as a 3d datapoints and visualized as datapoints, heatmaps, and tabulated data.

**VisionMap**: (Figure 12)
Each agent is constantly tracking a 3d field of vision. The cumulative VisionMap shows a heatmap of 3d areas seen most to least collectively by the crowd of agents.

**SoundMap:** (Figure 13, Figure 14)
Each agent produces sound projected as a decaying wave bouncing off partition walls and other design features in the 3d model. The action type an agent performs when sound is produced dictates the loudness of the projected sound (ex. Focused work is soft, walking is medium, conversations and meetings are loud). Current step and cumulative SoundMaps can be visualized as a heatmap identifying the most noisy and quiet areas.

**Destination Data / Asset Utilisation:**
Each interactable workplace destination (desks, meeting tables, kitchens, collaboration booths, etc.) stores utilisation data identifying how often and to what extent it was utilised throughout a simulation.

**Map Export:** Figure 15)
DataMaps are exported to Json Format readable in Grasshopper and the Design Tool

**Screenshots**

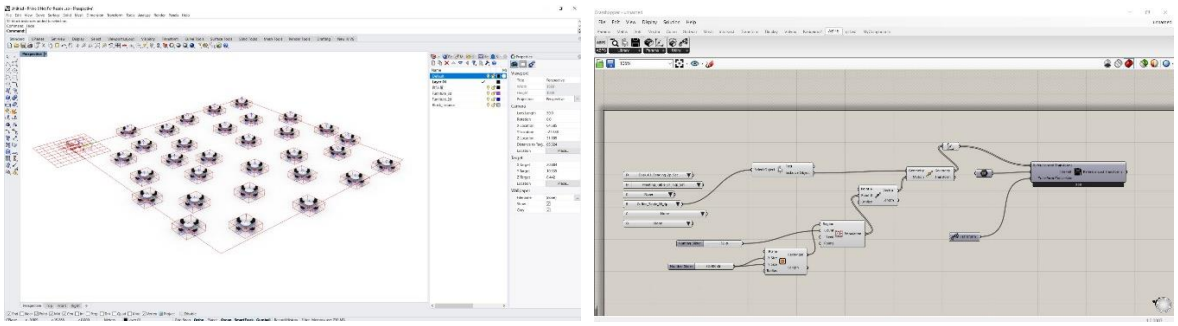


Figure 5 This figure shows the deployment and export of Rhino block instances of tables and chairs from the Mindspaces Asset Library (Left: Rhino window, Right: Grasshopper canvas window)

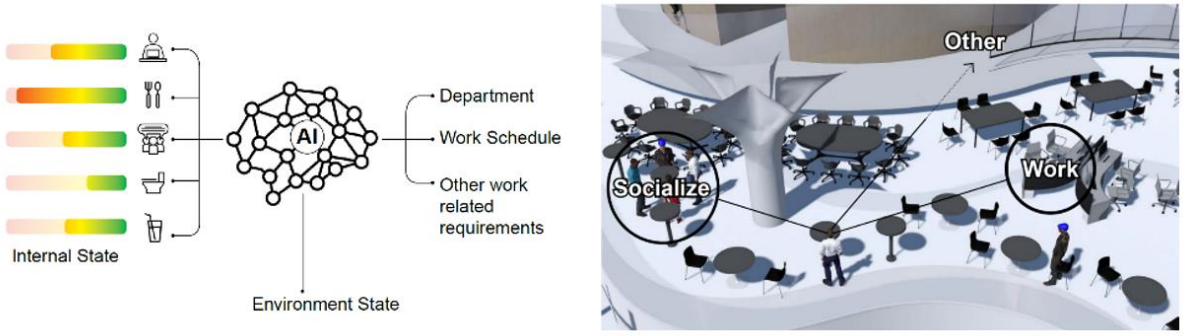



Figure 6 This figure shows agent AI diagram and agent in its workplace environment diagram

Figure 7 An image of one frame in the ABPS Behavioural Simulation Tool

Figure 8 This figure shows the agents' cumulative Datapoints spatially recording each agent's actions taken during the simulation

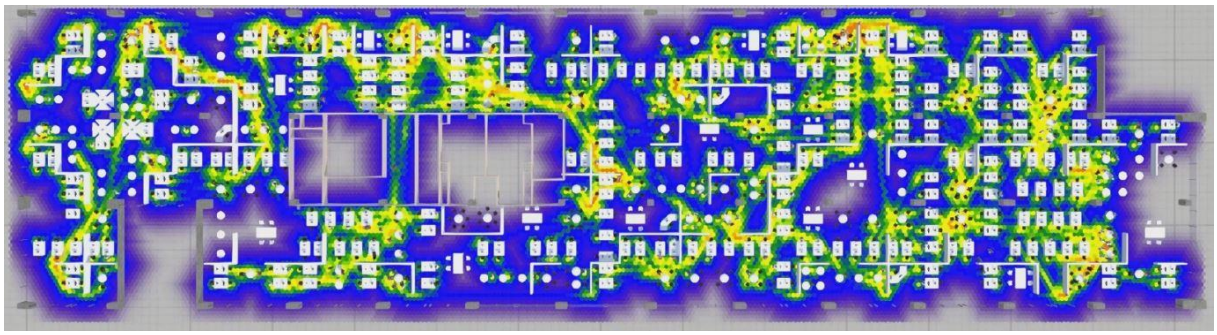


Figure 9 This figure shows the cumulative HistoryMap of all agents actions taken over a simulation

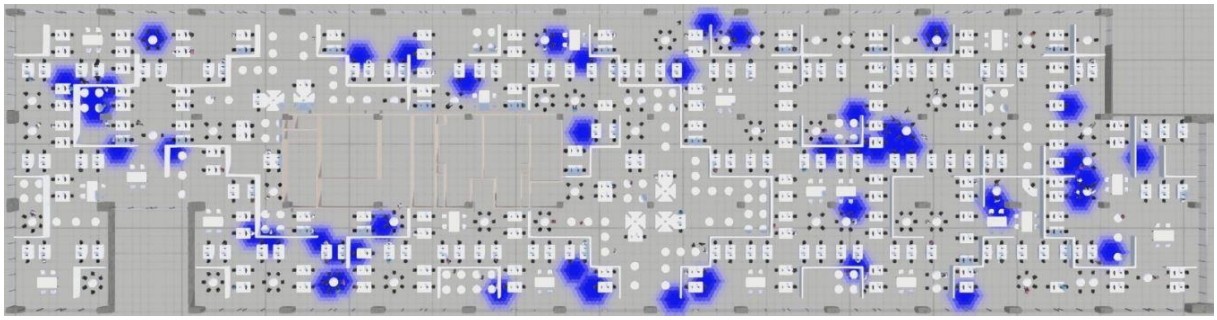


Figure 10 This figure shows the cumulative EncounterMap of all agents' conversations and collaborations

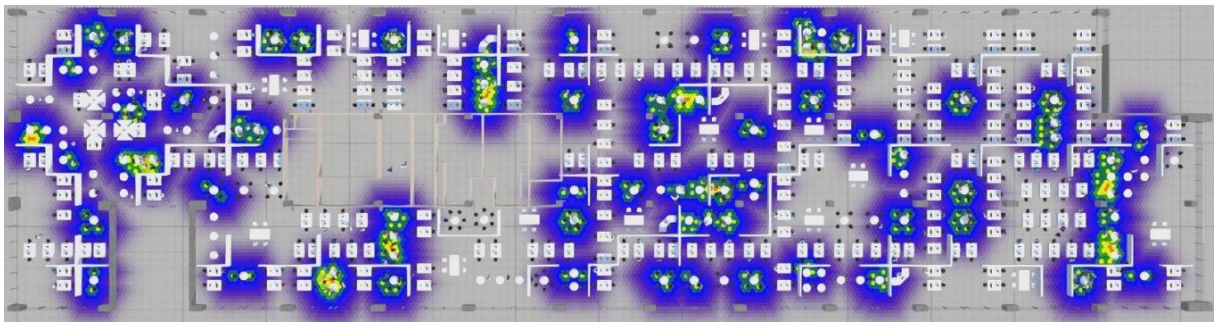


Figure 11 This figure shows the cumulative CollaborationMap of all agents' conversations and collaborations

Figure 12 This figure shows the cumulative VisionMap of all agents' collective 3d vision over the simulation

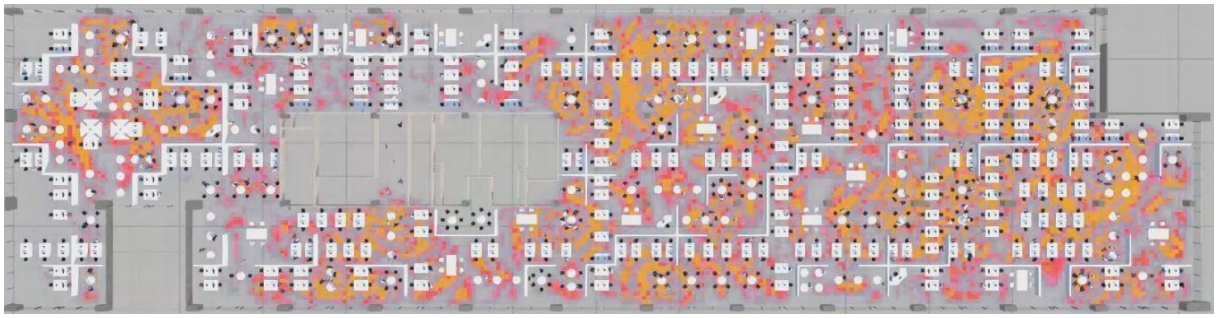


Figure 13 This figure shows a live step in the SoundMap of all agents' sound dispersion

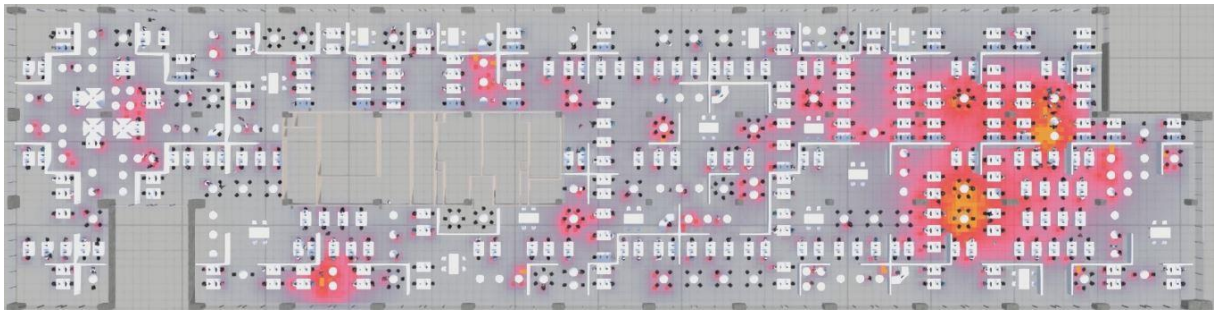
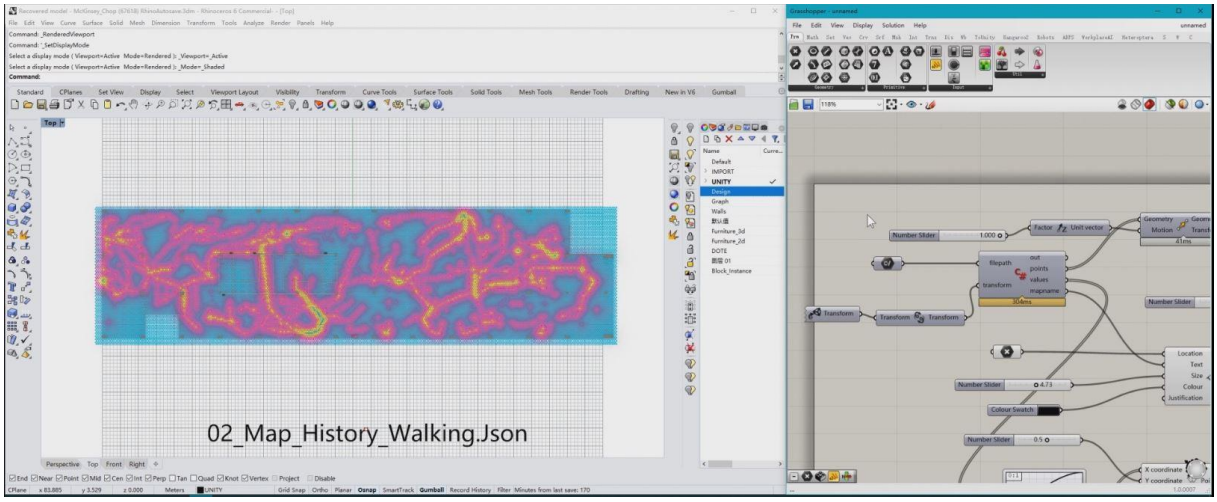


Figure 14 This figure shows a cumulative SoundMap of all agents' sound dispersion over a simulation



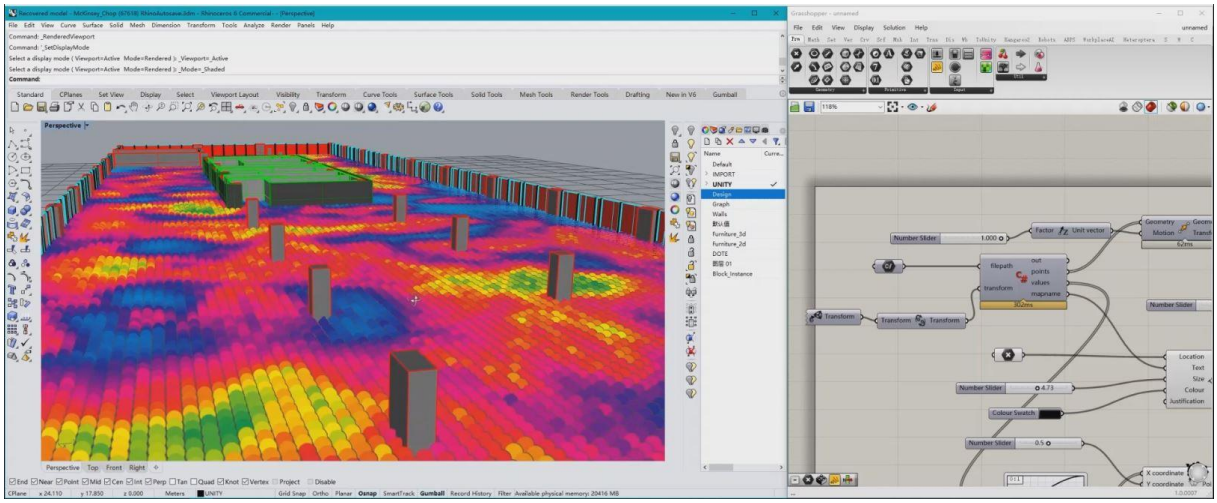


Figure 15 These figures show the importing of DataMaps in Grasshopper from the Json files exported by the behavioural simulation.

**Prototypes and timelines**

The first prototype of ABPS Behavioral Simulation Tool will be delivered with V1 of the MindSpaces platform, in M19 of the MindSpaces project. This first prototype is developed to run crowd behavioral simulations on Unity models imported from Grasshopper or the Design Tool using our UnityAssetImporter component as well as from models generated in the ABPS Generative Design Tool. Behavioral data is visualized both inside the tool and can be exported back to Grasshopper and the Design Tool. V1 operates through a simple user interface to control simulation input variables, collect, view, and export simulation data. The V2 prototype of ABPS Generative Design Tool will be developed after delivering V1 of MindSpaces platform and it will include improved agent decision making, more advanced user control, and improved integration with the Design Tool and ABPS Generative Design Tool. One goal in future versions is to integrate where possible any relevant insights obtained from the empirical Behavioral Analysis component data collection and analysis to calibrate parameters in the agent-based decision making.

### 2.2.4 **Aesthetic Analysis**

The aesthetic analysis comprises two components. The style transfer component applies the style transfer technique to raw images from materials aiming to create a gallery of new materials, available for users for the design of spaces and artworks. The second component is the colour palette generator which produces colour palettes from archival material aiming to inspire MindSpaces users for the design of spaces.

**Style Transfer**

| **Name of the component: Style Transfer** |
| --- |
| **Description of the component** |

Style transfer is a technique that modifies the style of an image, while keeping its content. One content image and one style image are fed into the style transfer network, in order to create the output stylized image.
The style transfer component applies a new style to a content image that is saved in the file storage and creates a new stylized image.
The aim of this component is to create new stylized materials using material images as content images and apply the 53 supported style images of paintings. A gallery of new stylized materials will be created through the style transfer component that could be used by architects and designers to create innovative and more artistic 3D environments. An example of applying a style image to a material image is shown in Figure 16 below.


**Figure 16 Example of applying a style image to a material image. The initial material is a type of gravel. The style transfer component applies the style image to the content image and the result is a new stylized material**

The component is implemented using gRPC framework in Python 3.7 and consists of the server and the client applications.
The client sends its messages for the creation of a new stylized material to the server. Client's message (STrequest) contains the raw image's id and the style image's id. The raw image is saved to the file storage. The 53 supported style images are saved locally to the server and their id can be a number from 1 to 53. After sending its request, client waits for the server's response.
The server receives client's request, logs in the file storage and uses the raw image's id to download the image. Then, the selected style image is applied to the downloaded raw image, using a deployed style transfer network and the new stylized material is created and saved locally. Next, the server uploads the stylized material to the file storage and gets a corresponding id. Server sends its message (STresponse) to the client. STresponse contains the upload status of the stylized image to file storage, the id of the stylized image and the url, in order to download the stylized image from the file storage.
Finally, the client application receives server's response and, once the uploading of the stylized image is successful ("upload status: 200"), it logs in the file storage and uses the id to download the stylized material.
Figure 17 shows a diagram of the communication between the client and the server applications, as well as their connection to the file storage.

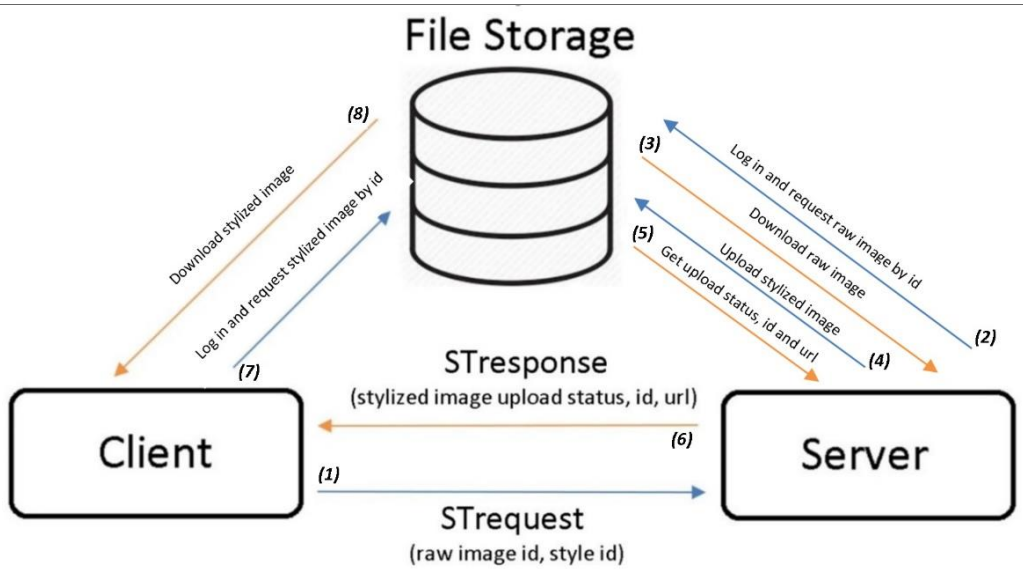



**Figure 17 Communication between client and server and their connection to the file storage. The numbers indicate the order of the actions.**

**Technical Requirements addressed**

ST component addresses the following TR(s):

- TR 2. Create a style transfer technique in order to change the style of 3D objects , which is related to the following User Requirements: UR_3, UR_7, UR_70, UR_71

**Input Data with sample**

The gRPC framework used a proto file to define the format of the messages sent between the client and the server. This file defines the format of STrequest and STresponse in our case.
As input data we consider the STrequest that the client application sends to the server, which is defined as followed in the proto file of the ST component:

```
//Define message by the client :
message STrequest{
    string rawimage_id = 1;
    string style_id = 2;
}
```

This means that the STrequest contains the raw image's and the style image's ids, so that the server gets all the information it needs to download the raw image from the file storage and create the new stylized image.

**Output Data with sample**

The same proto file of the ST component also defines the message that the server application sends to the client, which is called STresponse and has the following format:

```
//Define a message to hold the predicted price :
message STresponse{
    string upload_status = 1;
    string styled_image_url = 2;
    string styled_image_id = 3;
}
```

The STresponse includes the status of the new stylized image's uploading, so that the client knows if it is successfully saved in the file storage and download it using the corresponding stylized image id.

***Screenshots***

In the link below, you can find a demo video for the usage of style transfer service.

https://github.com/palvanit80/MindSpaces

The process captured in the demo video consists of the steps below:

1. Server application runs on MindSpaces server and is ready to receive client's request.
2. Client sends its request (STrequest) to the server.
3. Server connects to the file storage and downloads the raw image using the given id (see Content Image in Figure 18).
4. The style images are saved locally to the server. The style image that corresponds to the style id sent by the client is applied to the content image and a new stylized image is created (see Style and Stylized Images in Figure 18).
5. The stylized image is initially saved locally and then uploaded to the file storage. When the upload is successfully finished, the stylized image gets an id, which is sent to the server.
6. Server sends its response (STresponse) to the client, giving all the information needed.
7. Finally, client connects to the file storage and downloads the stylized image, using the corresponding id.

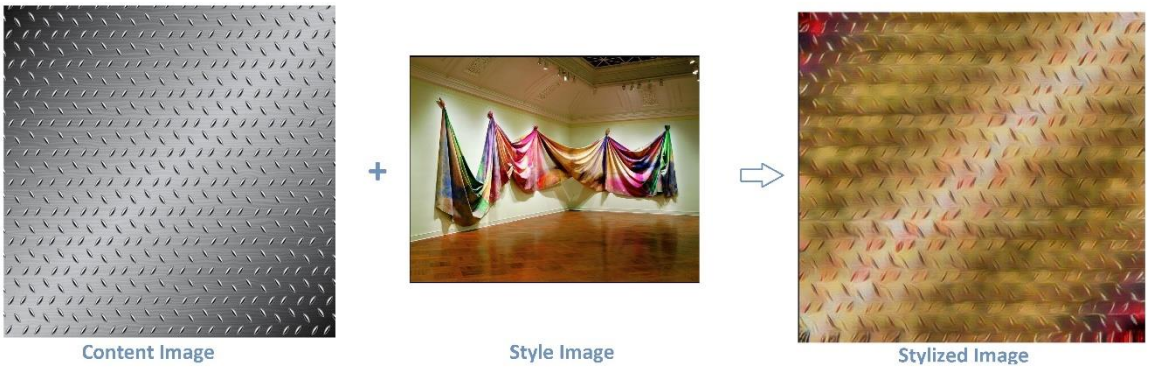



Figure 18 This figure shows the result of the combination of the depicted content and style images. The initial material is a type of metal and the style transfer component creates a new stylized material.

***Prototypes and timelines***

For the style transfer component we have the following prototypes, as referred in D6.1:

- 1st Prototype (V1) – M17

The 1st version of the style transfer component, which is described at the current deliverable (D6.3), is included in the 3rd milestone (MS3) at M20 of MindSpaces project. The current deliverable (D6.3) contributes to MS3.

- 2nd Prototype (V2) – M26

The 2nd version of the style transfer component will be deployed by M26 and the idea is to support style transfer using multiple different styles, instead of only one style image. This prototype is included in the 4th milestone (MS4) at M28.

- Final Version – M33

The final version (FV) of the style transfer component will be deployed by M33. The 5th milestone (MS5) includes this final version, which ends the last month of the project (M36).

Figure19 shows the timeline for the development phases of the style transfer component.

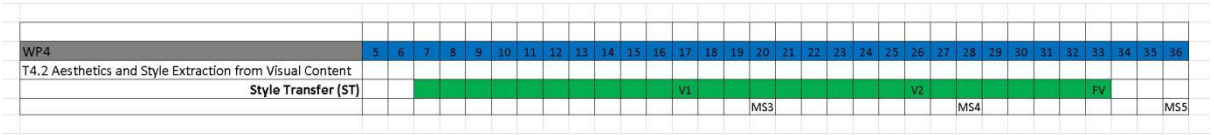



Figure 19 Style transfer component prototypes and timeline

**Colour Palette generator**

| **Name of the component: Colour Palette Generator** |
|---|
| **Description of the component** |
| Colour Palette service is an ASP.NET core web application. It combines a web api and an MVC application. This component aims to extract aesthetics (colour palettes) from visual content. It is a service that targets artists and creatives. They could use this tool as a baseline to build novel 3D-artworks or spaces.<br>The colour palette information can be extracted either from archival material (cultural heritage artefacts in paintings, images of artwork or videos).<br><br>The current version of the service provides colour palettes extracted from a list of paintings from WikiArt. More specifically:<br>- Wikiart Emotions dataset[1] is used for our experiments. WikiArt Emotions is a dataset of 4,105 pieces of art WikiArt Emotions includes annotations for emotions evoked to the observer. A set of paintings is examined for the extraction of colours.<br>- A set of 597 abstract paintings are selected. The related emotions are: anger, anticipation, disgust, fear, happiness, optimism, other, pessimism, sadness, surprise.<br>- The styles of paintings include: Modern Art and Contemporary Art.<br>The colour palette service is similar to the colormind.io. The goal is to use a deep learning architecture (Pix2Pix GAN Architecture) in order to learn colour styles from paintings and other pieces of artwork and produce colour palettes. |

[1] http://saifmohammad.com/WebPages/wikiartemotions.html

- A sql server is created to store the data from wikiart emotions dataset.
- The web API reads the data from the SQL server and exports a json file with all the attributes.
- The Colour Palette web API is a backend component for the integration of the colour palettes into the design tool.
- It includes features such as the name of the creator, the style of the painting, the genre, the image width, image height, link of the image and the 5 dominant colours.
- Cross-domain requests are supported.

In the next version of the colour palette generator service some considerations for the new features include CRUD operations and a more user driven approach with deep learning models trained with python ML libraries, from other artwork/design sources, defined by users.

**Technical Requirements addressed**

Colour Palettes component addresses the following TR(s):

- TR 1. Create a colour palette generator service. UR_3, UR_7, UR_12, UR_71

**Input Data with sample**

**1)** Web api call: 160.40.52.102:14140/api/colorpalettes
**2)** MVC application: 160.40.52.102:14140/list

**Output Data with sample**

A screen-shot of the MVC application can be seen in the following Figure.

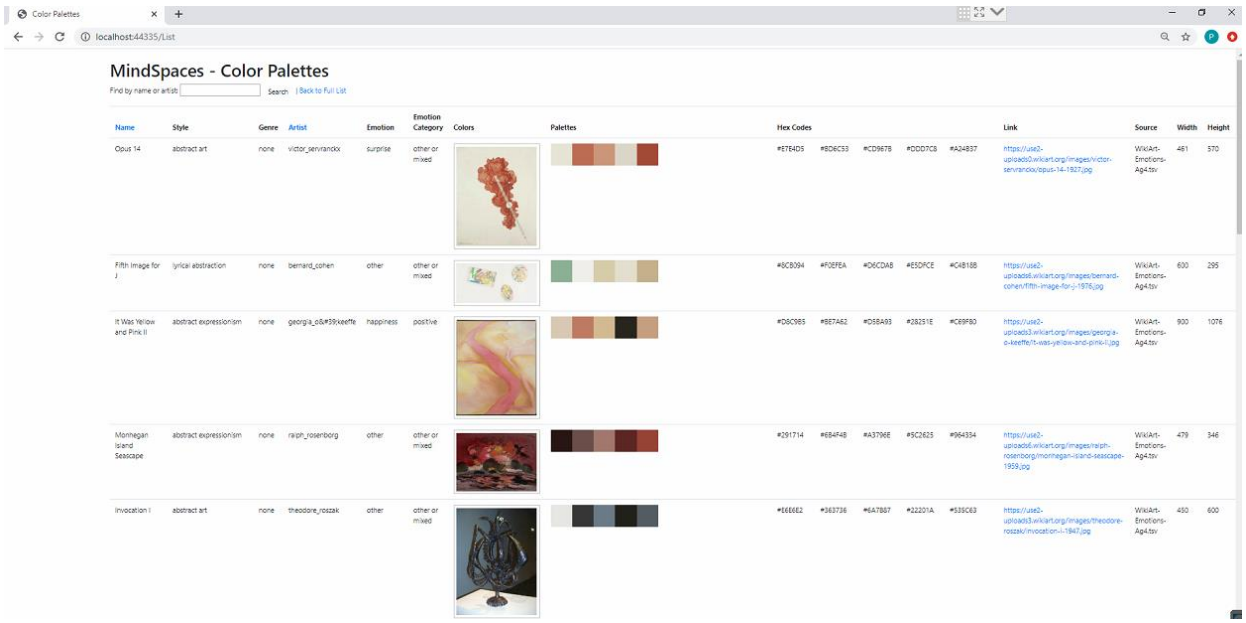



**Figure 20 A screen-shot from the MVC application of Colour Palettes service.**

Figure 20 shows the response from the web api which returns the json file with all the attributes saved in the SQL database.

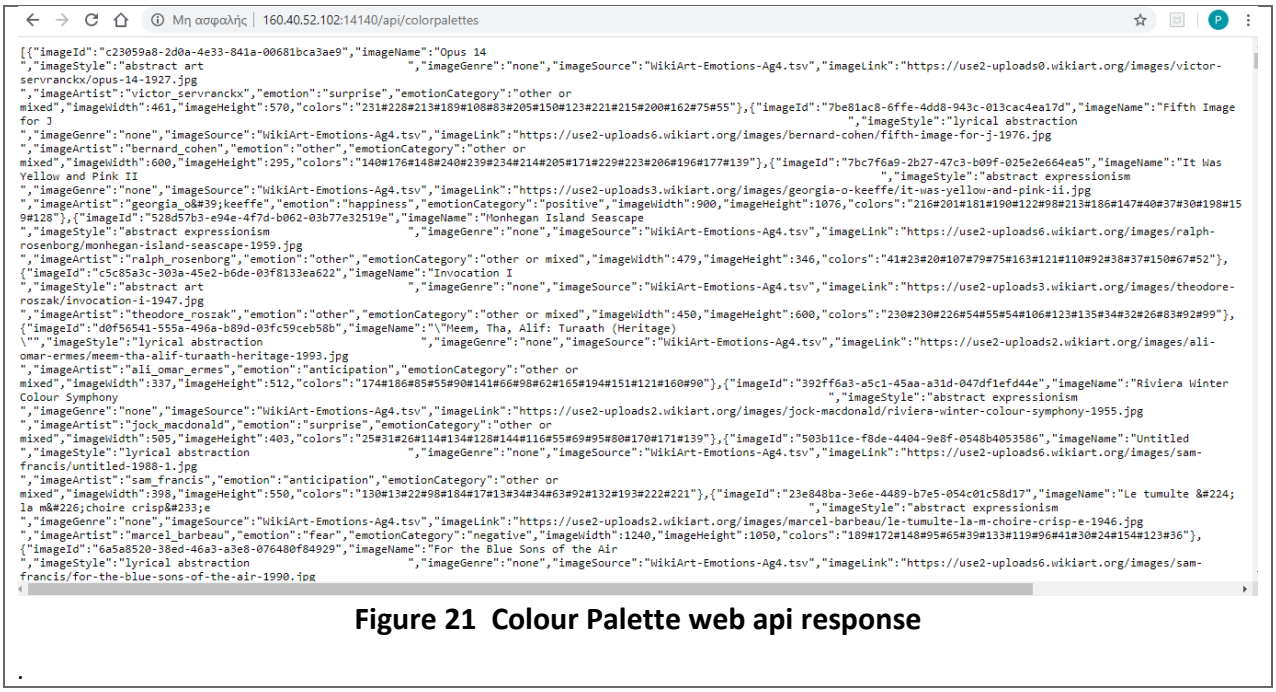



**Figure 21 Colour Palette web api response**

***Screenshots***

An available demo for the Colour Palette service is available in github. The link is the following:

https://github.com/palvanit80/MindSpaces

***Prototypes and timelines***

For the Colour Palette component we have the following prototypes, as referred in D6.1:

- Basic Version (BV) – M12

  This version is the skeleton service. It includes the definition of inputs, outputs and the functionalities foreseen for the next version of the service. It supports the integration of the service through the communication with the rest connected components of the MindSpaces platform.

- 1st Prototype (V1) – M15

  The 1st version of the colour palette component, which is described at the current deliverable (D6.3), is created at M15 of MindSpaces project. It includes the web api for the connection to the design tool and the MVC application for the creation of the UI for users.

- 2nd Prototype (V2) – M22

  The 2nd version of the colour palette component will be deployed by M22. It will support more functionalities and will be an intelligent tool which will support the generation of colour palettes from GANs trained based on data from additional sources of artwork/design.

- Final Version – M33

The final version (FV) of the colour palette component will be deployed by M33. The 5th milestone (MS5) includes this final version with all the updates based on the requirements from users.

Figure 22 shows the timeline for the development phases of the colour palette component.

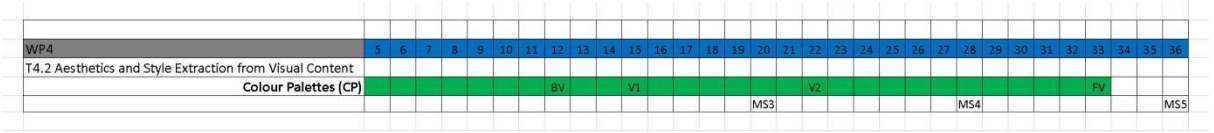



Figure 22 Colour Palette component prototypes and timeline

### 2.2.5 **Knowledge Base**

| **Name of the component** |
|---|
| **Description of the component** |
| The Knowledge Base component encapsulates the population service and the reasoning service. The KB population service aims at mapping the results created by the various MindSpaces services as analysis results to the RDF-based representation structures relying on the ontologies that are iteratively being developed (vocabulary development, semantic relations and properties). The reasoning service is offering custom adhoc rules based on knowledge fusion and performs partially localized emotion-based logic procedures towards Virtual Reality adaptations. Furthermore, it is responsible for the query formulation, the translation of the rule authoring tool's user interactions into one or more rules to the backend of the reasoning tool which will be triggered as appropriate. |
| **Technical Requirements addressed** |

| TR | Title | Description | Functions | Function performed | Related URs |
|---|---|---|---|---|---|
| TR 8. | Create a knowledge base that will fuse multimodal data and unify them into one ontological model | The KB will serve as a semantic information database, holding data and controlling the flow about geolocation, and correlating emotional states with objects present at scene. | Behavioural Analysis Offline Bulk Bundle Results Insertion,<br>Aesthetics Analysis Offline Bulk Bundle Results Insertion,<br>Textual Analysis Offline/Online Bulk Bundle Results Insertion,<br>EEG Analysis Results Online Insertion | Mapping multiple JSON data objects to RDF data | UR_5, UR_10, UR_11, UR_12, UR_13, UR_14, UR_15, UR_16, UR_51, UR_61, UR_62 |
| TR 9. | Create a Reasoning Service providing the changes that are wanted to occur | The Reasoning Service will provide a set of changes to occur inside the VR based on logic on what the user (artist/architect) want to achieve as the emotional state of the subject | StartSearch, StartReasoning | Query formulation / enrichment | UR_24, UR_41, UR_42, UR_44, UR_50, UR_57, UR_58, UR_64 |

**Input Data with sample**

| Input source | Data Type (live stream, packaged, or static) | Description of the data |
|---|---|---|
| JSON objects | live stream | JSON objects are surrounded by curly braces {}. JSON objects are written in key/value pairs. Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null). Keys and values are separated by a colon. Each key/value pair is separated by a comma. |
| SPARQL Queries | Static | A semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format |

**JSON object Input Data Sample**

```
CONFIGURATIONS: [
{  ID: "556631",
   PROJECT-ID: "986",
   ORDER: 1,
   PARAMETER: "greenspace",
   BASE: "default",
   TRANSFORMATIONS: [
          { ID: "556631_1",
            MODEL: "Default_Model_URL",
            CHILD_INDEX: [],
            TRANSFORMATION: "Appear",
            TRANS_PARAMS: {
            objs: [
                        {  obj_id: 986_OBJ02, location:[100,20,41], scale:[0.8, 0.8, 0.8]  },
                        {  obj_id: 986_OBJ02, location:[205,30,41], scale:[0.8, 0.8, 0.8]  },
                        {  obj_id: 986_OBJ02, location:[300,40,41], scale:[0.8, 0.8, 0.8]  },
                  ] },
            ANIMATION_PARAMS: {smooth=true, duration:1000 }  }
          },
          { ID: "556631_2",
            MODEL: "Default_Model_URL",
            CHILD_INDEX: [],
            TRANSFORMATION: "Appear",
            TRANS_PARAMS: {
            objs: [
                        {  obj_id: 986_OBJ03, location:[100,20,41], scale:[0.8, 0.8, 0.8]  },
                        {  obj_id: 986_OBJ03, location:[205,30,41], scale:[0.8, 0.8, 0.8]  }
                  ] },
            ANIMATION_PARAMS: {smooth=true, duration:1000 }  }
          }]
]}
```

**SPARQL Query Input Data Sample**

```
'PREFIX : <https://mindspaces.eu/ontologies>\
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\
    \
    SELECT ?BATagF ?EEGTagF ?ParamF ?3DObjectF ?typeF WHERE {\
      {\
```

```
    ?EEGSubject :hasEEGTag ?EEGTag       .\
    ?EEGTag :isLatestEEGTag true\
    BIND(STRAFTER(str(?EEGTag), "#") AS ?EEGTagF)\
    } UNION {\
  ?BASubject :hasX ?BAX.\
  {select ?BAX WHERE {\
                ?EEGSubject :hasEEGTag ?EEGTag.   \
                ?EEGTag :isLatestEEGTag true. \
                ?EEGSubject :hasX ?BAX}\
  }\
  ?BASubject :hasY ?BAY.\
        {\
             SELECT ?BAY WHERE {\
                ?EEGSubject :hasEEGTag ?EEGTag.   \
                ?EEGTag :isLatestEEGTag true.\
                ?EEGSubject :hasY ?BAY\
                }\
           }\
   ?BASubject :hasZ ?BAZ.\
        {\
             SELECT ?BAZ WHERE {\
                ?EEGSubject :hasEEGTag ?EEGTag.\
                ?EEGTag :isLatestEEGTag true .\
                ?EEGSubject :hasZ ?BAZ\
                }\
           }\
     \
  ?BASubject :hasBATag ?BATag\
  BIND(STRAFTER(str(?BATag), "#") AS ?BATagF)\
  }UNION{\
    ?EEGSubject :InsideofEEGPersonView ?3DObject.\
    ?3DObject :includesParam ?Param.\
    ?Param rdf:type ?type\
    filter (?type not IN (:Parameter))\
    filter (?type not IN (owl:NamedIndividual))\
    filter (?type not IN (:Visuals))\
    BIND(STRAFTER(str(?Param), "#") AS ?ParamF)\
    BIND(STRAFTER(str(?3DObject), "#") AS ?3DObjectF) \
    BIND(STRAFTER(str(?type), "#") AS ?typeF)\
    }\
  }'
```

**Output Data with sample**

| Output data | Data Type (live stream, packaged, or static) | Description of the data |
|---|---|---|

| | | |
|---|---|---|
| RDF Triplets | static | a RDF triple is a set of three entities that codifies a statement about semantic data in the form of subject–predicate–object expressions |
| JSON Object | Live stream | JSON objects are surrounded by curly braces {}. JSON objects are written in key/value pairs. Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null). Keys and values are separated by a colon. Each key/value pair is separated by a comma. |

**RDF Triplet Output Data Sample**

```
rdf:type                rdf:typerdf:Property
rdfs:subPropertyOf      rdf:typerdf:Property
rdfs:subPropertyOf      rdf:typeowl:TransitiveProperty
rdfs:subClassOf         rdf:typerdf:Property
rdfs:subClassOf         rdf:typeowl:TransitiveProperty
rdfs:domain             rdf:typerdf:Property
rdfs:range              rdf:typerdf:Property
owl:equivalentPropertyrdf:typeowl:SymmetricProperty
owl:equivalentPropertyrdf:typeowl:TransitiveProperty
```

**JSON Object Output Data Sample**

```
{current_configuration_id : 12321,
configuration_id : 12341,
state : {timestamp: 1592227429, eeg_tag: 2,valence: 1,arousal: 2},
location : {hotspot_id : 5, pos_x : 102, pos_y : 243, pos_z : 15},
triggered_rule: triggered_rule}
```

**Screenshots**

Unavailable due to pure backend application

**Prototypes and timelines**

[M20] 1st prototype: Full support for Version 1 regarding mapping of incoming information, ontological structures and the reasoning tool.

[M28] Further development and allocation of an updated version regarding the mapping procedure (e.g. from offline modules to online), the ontological model which follows an iterative evolution approach and the reasoning mechanism.

[M34] Final version of the updated services inside the component and finalization of the framework.

### 2.2.6 **Language Generation**

| **Name of the component** |
|---|
| **Description of the component** |
| The text generation service takes structured input from the knowledge base, which originates from non-textual sources such as outputs from the processing of EEG or visual signals (i.e., aggregated *valence* and *arousal* values, EEG tag and timestamp, subject's location), and identifiers of VR configurations (current and suggested) along with textual descriptions of categorical variables, and verbalizes the information as human-readable text in one of the supported languages. The graph-transduction grammars and lexical resources are used to perform all the main stages in the generation (text planning, linguistic generation, determination of the structure of the sentence, introduction of grammatical words, resolving the morphological agreements between the words are resolved, ordering the words, introduction of punctuation signs) to provide a projection of complex ontological configurations onto lexicalized semantic structures and their subsequent realization as natural language sentences. |
| **Technical Requirements addressed** |

| TR | Title | Description | Function | Function performed | Related URs |
|---|---|---|---|---|---|
| TR 20. | Create a technique of the projection of ontology constructs to respective lexicalized semantic structures to support knowledge-driven content selection | A methodology and algorithm for the mapping of ontological representations to conceptual structures appropriate for linguistic generation purposes will be developed | Ontological data transformation | Ontological data is mapped to semantic structures | UR_7, UR_9, UR_10, UR_11, UR_12, UR_13, UR_14, UR_15, UR_16, UR_17, UR_19, UR_21 |
| TR 21. | Create a linguistic generation service for the realization of acquired knowledge as natural language sentences | The linguistic generation service will allow for presenting in human-readable form the information derived from ontological representations. In order to support multilingual text generation, the compilation of the linguistic resources will be conducted and statistical modules will be trained | Generation of natural language sentences | Generates texts in three languages (English, Spanish, Catalan) | UR_7, UR_9, UR_10, UR_11, UR_12, UR_13, UR_14, UR_15, UR_16, UR_17, UR_19, UR_21 |

| **Input Data with sample** |
|---|

Input format: JSON

Sample:

```
{
  current_configuration_id: 1
  configuration_id: 2
  state {
    timestamp: 2
    eeg_tag: 2
    valence: 4
    arousal: 5
  }
  location {
    hotspot_id: 1
    pos_x: 2.0
    pos_y: 3.0
    pos_z: 4.0
  }
  triggered_rule: "switch (call.request[\"state\"][\"eeg_tag\"])@@@ case 3"
}
```

## Output Data with sample

Output format: plain text (generated human-readable text in one of the supported languages)

Sample (English): "The walls were green , the ceiling low , the room round and it was lit from the top . The subject was probably feeling joy so the wall 's colour was changed to blue ."

Sample (Catalan): " el color de les parets era groc , l' alçada del sostre  baixa , la forma de l' habitació  quadràtica i s' il·luminava l' habitació des de baix . l' individu podia estar sentint enuig i per això el color de la paret es va canviar a "blau""

### Screenshots

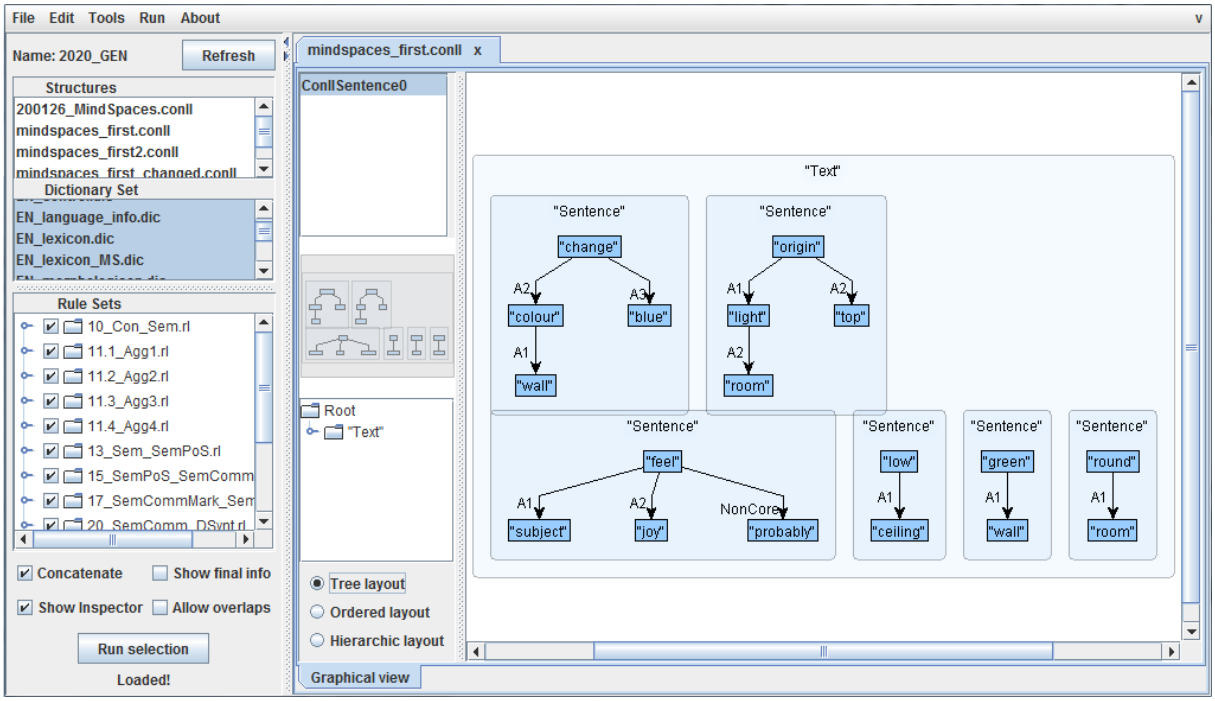



Figure 23 Screenshot of the Text Generation demo – initial input structures

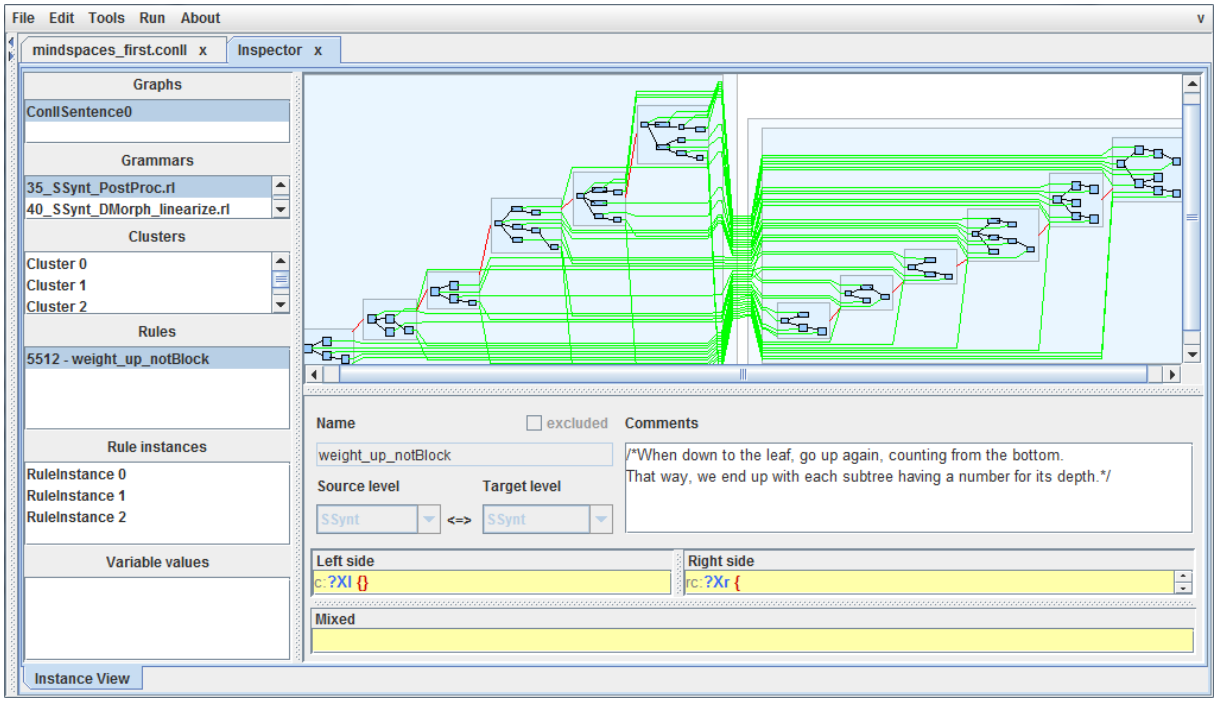



Figure 24 Screenshot of the Text Generation demo – intermediate representation as a result of several subsequent graph transformations

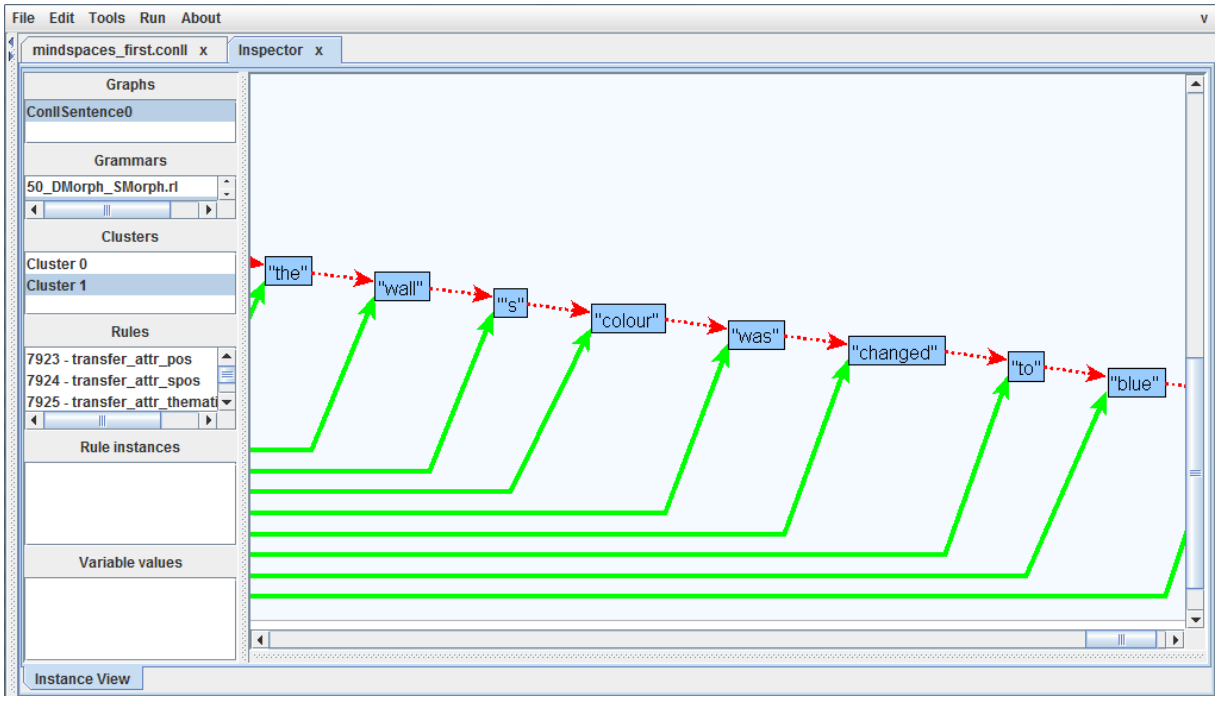



Figure 25 Screenshot of the Text Generation demo – final output as a linear text

**Prototypes and timelines**

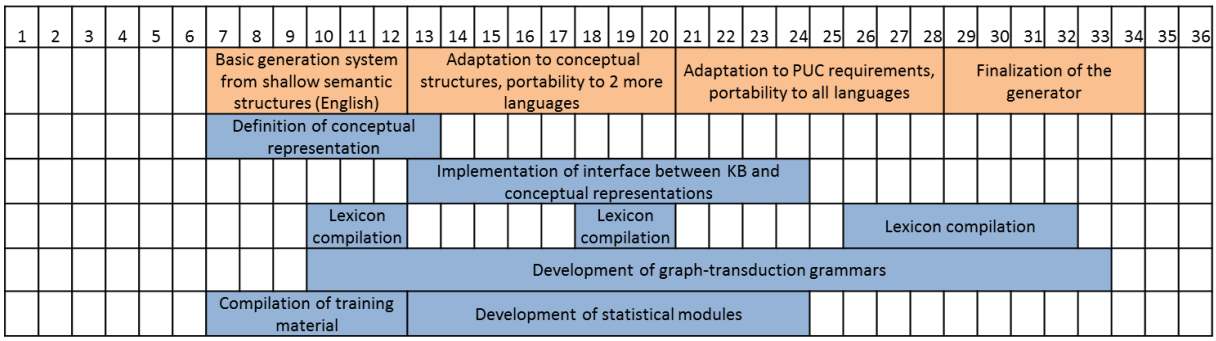



Figure 26 Development timeline with prototypes

### 2.2.7 Discourse Analysis + Sentiment Analysis

| **Name of the component** |
|---|
| **Description of the component** |
| The textual analysis component receives unstructured text data (in any of the supported languages) and extracts (i) discourse information, including but not limited to concepts appearing in the text, when possible with links to external knowledge resources such as DBpedia or WordNet, and relations between them; (ii) sentiments in citizens' contributions to social media, in transcripts of interviews, and opinions in online content on websites.<br>The information extracted from textual collections is fed into a knowledge base where it is used along with information acquired from data sources of another type to allow for an understanding of end-users' feelings and needs and further semantic reasoning. |
| **Technical Requirements addressed** |

| TR | Title | Description | Function | Function performed | Related URs |
|---|---|---|---|---|---|
| TR 16. | Create semantic parsing technique to capture the underlying semantics of the textual material | An algorithm of deep-parsing of the textual inputs will be developed in order to obtain linguistic predicate-argument structures that capture the underlying semantics | Semantic parsing | Creates predicate-argument structures from textual content in three languages (English, Spanish, Catalan) | UR_6, UR_8, UR_20, UR_65, UR_72, UR_82 |
| TR 17. | Create concept extraction technique to support the contextual interpretation within the reasoning | Concept extraction will serve for generic entity detection, named entity recognition, and entity linking. The multilingual nature of the considered inputs and outputs will be taken into account | Concept extraction | Extracts entities from texts in three languages (English, Spanish, Catalan) | UR_6, UR_8, UR_20, UR_65, UR_66, UR_72, UR_74, UR_82 |

| | | | | | |
|---|---|---|---|---|---|
| TR 18. | Create techniques for the projection of linguistic structures onto formal abstract representations that will be factored into knowledge graphs | The algorithm for the projection of linguistic structures onto formal abstract representations will be developed in order to allow for the population of the ontologies with information that supports understanding of sentiments, feelings, historical inputs and needs | Creation of abstract linguistic representations | Projects linguistic structures onto formal representations in three languages (English, Spanish, Catalan) | UR_6, UR_8, UR_20, UR_65, UR_66, UR_72, UR_74, UR_82 |
| TR 19. | Adapt existing sentiment analysis tools to analyze emotions in citizens' contributions to social media and opinions in online content on websites | Off-the-shelf state-of-the-art technologies in sentiment analysis will be adapted to the languages of the project to provide detection of emotions and opinions | Aspect-oriented Sentiment analysis | Detects emotions and aspects in texts in three languages (English, Spanish, Catalan) | UR_6, UR_8, UR_20, UR_65, UR_72, UR_82 |

## Input Data with sample

Input format: plain text

Sample (English): “We are the city with the highest density of population in Europe and the second with the least green zone in Catalonia!”

Sample (Spanish): “De hecho, el km2 con más densidad de población de toda Europa está en Hospitalet de Llobregat.”

Sample (Catalan): “L'Hospitalet de Llobregat es la ciudad más densa de Europa y la siguiente es Badalona.”

## Output Data with sample

Output format: JSON

Sample:


```
{
  "id":"287c5202-ba7d-4bc6-8c0b-6a72631d4908",
  "text":"We are the city with the highest density of population in Europe and the second with the least green zone in Catalonia!",
  "language":"en",
  "result":{
    "ner":{
      "directed":true, "type":"ner", "label":"ner",
      "nodes":[...], "edges":[], "metadata":{}
    },
    "surface_parsing":{
      "directed":true,"type":"surface","label":"surface dependencies",
      "nodes":[...],"edges":[...],"metadata":{}
    },
    "concept_extraction":{
      "directed":true,"type":"concept","label":"concept extraction",
      "nodes":[...],"edges":[],"metadata":{}
    },
```

```
    "emotion_recognition":{
      "directed":true,"type":"emotion","label":"emotion recognition",
      "nodes":[...],"edges":[],"metadata":{}
    },
    "dbpedia_linking":{
      "directed":true,"type":"dbpedia","label":"dbpedia linking",
      "nodes":[...],"edges":[],"metadata":{}
    },
    "deep_parsing":{
      "directed":true,"type":"deep","label":"deep dependencies",
      "nodes":[...],"edges":[...],"metadata":{}
    },
    "predicate_arguments_parsing":{
      "directed":true,"type":"predargs","label":"predargs dependencies",
      "nodes":[...],"edges":[...],"metadata":{}
    }
  },
  "metadata":{"date":"2020/06/11 16:34:42"}
}
```


**Screenshots**

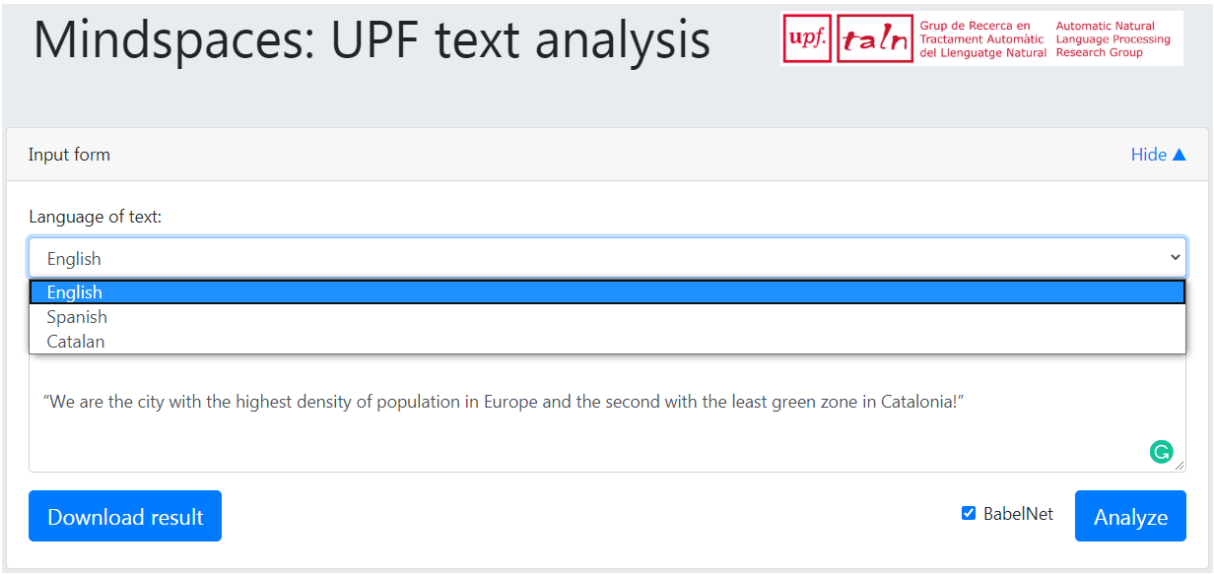



Figure 27 Screenshot of the Textual analysis demo – input form with language selection

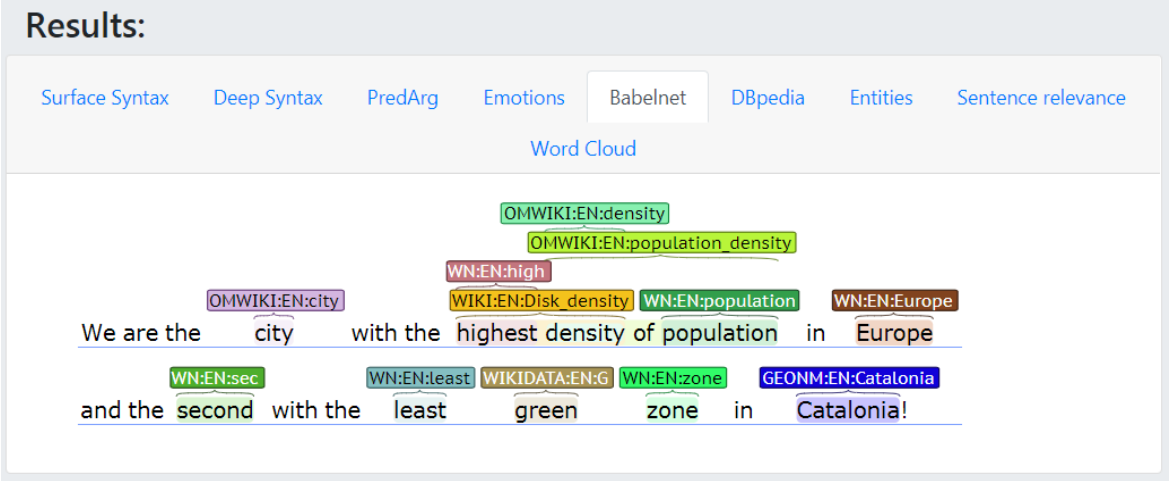



Figure 28 Screenshot of the Textual analysis demo – results of entity linking for English

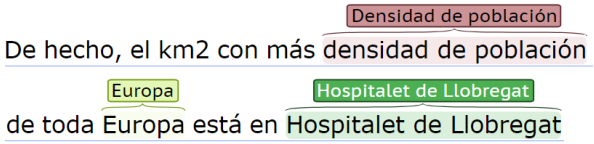



Figure 29 Fragment of screenshot of the Textual analysis demo – results of entity linking with DBpedia Spotlight for Spanish

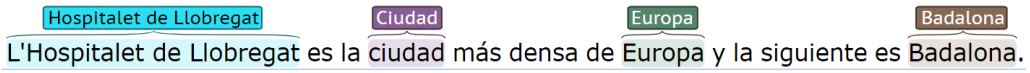



Figure 30 Fragment of screenshot of the Textual analysis demo – results of entity linking with DBpedia Spotlight for Catalan

**Prototypes and timelines**

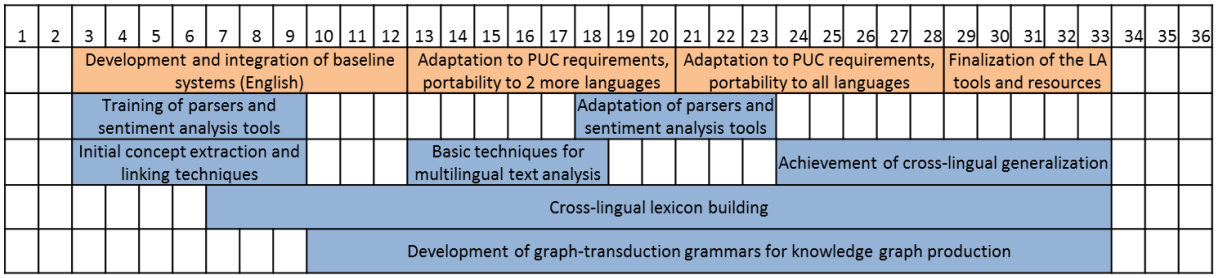



Figure 31 Development timeline with prototypes -Discourse Analysis

### 2.2.8 **Crawler Service**

| **Name of the component** |
|---|
| **Description of the component** |
| The crawling services satisfy and integrate every crawling and scrapping functionality as conceived for the needs of MindSpaces. In particular, its main purpose at this stage is to extract sufficiently, freely and publicly available textual content from open web resources, such as the social media platform of Twitter (user posts along with metadata) and pre-selected |

domain-related websites (author posts and user comments wher applicable), and feed it to a NoSQL document server inside the file storage.

**Technical Requirements addressed**

| TR | Title | Description | Function | Function performed | Related URs |
|---|---|---|---|---|---|
| TR 4. | Create a crawling service to gather online content from social media | Frameworks will be developed that will utilize available APIs from relevant social media platforms to fetch online public user data | Twitter Crawling | Extracts content from social media | UR_6, UR_9, UR_20, UR_65, UR_72 |
| TR 5. | Create a scrapping service to gather online content from websites | A framework will be developed that will fetch online content from websites | Scrapping From Selected Websites | Extracts content from web pages | UR_6, UR_9, UR_20, UR_65, UR_72 |

**Input Data with sample**

| Input source | Data Type (live stream, packaged, or static) | Description of the data |
|---|---|---|
| Strings provided by the user (artist/architect) | Live stream | The user (artist/architect) will provide the content of the query in a string format. |

**Twitter Crawler Sample**

{"keyword_list_0": {"terms": ["\"Accessibilitat\"", "\"arquitectura\"", "\"ciutat\"", "\"entorn urb\u00e0\"", "\"espai urb\u00e0\"", "\"est\u00e8tica\"", "\"industrial\"", "\"medi ambient\"", "\"mobilitat\"", "\"sostenibilitat\"", "\"Tecla Sala\"", "\"urbanisme\"", "\"modernitat\"", "\"territori\""], "querystring": "(\"accessibilitat\") OR (\"arquitectura\") OR (\"ciutat\") OR (\"entorn urb\u00e0\") OR (\"espai urb\u00e0\") OR (\"est\u00e8tica\") OR (\"industrial\") OR (\"medi ambient\") OR (\"mobilitat\") OR (\"sostenibilitat\") OR (\"tecla sala\") OR (\"urbanisme\") OR (\"modernitat\") OR (\"territori\")", "since_id": 1255102674556633088, "geocode": null}}

**Website Crawler Sample**

{"url":{"base_url":"https://www.danieldavis.com"},"group_of_urls":["/page/1/","/page/2/","/page/3/","/page/4/","/page/5/","/page/6/","/page/7/","/page/8/","/page/9/","/page/10/","/page/11/","/page/12/"],"source_name":"danieldavis","entity_name":"danieldavis","table_selector":"div.post","entity_info":[{"label":"article_title","value":{"selector":"h1","type":"text"}}],"metrics":[{"label":"crawl_to","value":{"selector":"a","type":"link"}}],"crawl":{"metrics":[{"label":"text_article","value":{"selector":"div.entry-content","type":"text"}},{"label":"comment","value":{"selector":"div.comment-body

p","type":"list"}}]},"store":{"format":"json","hd_path":"C:/Users/vasgat/Desktop/danieldavis_with_comments_result.json"},"dynamic_page":false}

**Output Data with sample**

| Output data | Data Type (live stream, packaged, or static) | Description of the data |
|---|---|---|
| JSON structures | Live Stream | The output contains json structures about each web entity crawled with the respective textual (and/or visual) data and metadata. |

**Twitter Crawler Output Sample**

{"created_at":"Wed Jul 03 13:13:19 +0000 2019","id":1146406567203528700,"id_str":"1146406567203528704","full_text":"Starwood sella la compra de cinco edificios de oficinas, en Madrid y Barcelona, a Oaktree por 153 millones https://t.co/KlUOB1kg2J @Ejeprime #oficinad","truncated":false,"display_text_range":[],"entities":{},"metadata":{},"source":"<a href=\"http://twitter.com/download/android\" rel=\"nofollow\">Twitter for Android</a>","in_reply_to_status_id":null,"in_reply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id_str":null,"in_reply_to_screen_name":null,"user":{},"geo":null,"coordinates":null,"place":{},"contributors":null,"is_quote_status":false,"retweet_count":0,"favorite_count":0,"favorited":false,"retweeted":false,"possibly_sensitive":false,"lang":"es"}

**Website Crawler Output Sample**

{"danieldavis":{"article_title":"Genetic Staircase by Caliper Studio"},"metrics":[{"name":"text_article","citeyear":2020,"type":"textual","value":"C aliper Studio’s Genetic Staircase translates obscure computer code into an accessible metaphor. The literal correlation between a genome and the metal woven between the coiled stairs, produces an easy to understand diagram of a genetic algorithm. It is not the first time a genetic algorithm has been used to design (see my past post on Archikludge) but for many people this will be the first time they understand what a genetic algorithm does. As a communication device the Genetic Staircase is exceptional but as a staircase I still wonder if the genetic algorithm has been successful. The program for the Genetic Staircase was a Rhino script. The script generates a population of individual staircases, each one with the diagonal members in different places. These individuals are then evaluated in an external structural analysis program (CADRE lite). The most successful designs produce offspring by splicing their diagonal members together using a single crossover point (Video of the crossover process). As an aside I think this is a technically limited method since Rhino script is slow and the structural analysis program had no scripting interface – both of these factors greatly limit the potential of the program to rapidly iterate between options. The actual stairs are manually defined using a parametric model, as are the main runners and the handrails. It is only the criss-cross of diagonal members under the stairs that have been created by the genetic algorithm.The problem is so well defined that the solution found by the genetic algorithm could have been anticipated beforehand; it was always going to be random diagonal members running under the stairs. The genetic algorithm therefore is not operating as a discovery tool but more as an optimisation tool. But what is it optimising? The staircase is not cheap, lightweight or easy to construct. Presumably the algorithm has some

variables that balance the cost of the staircase with the visual weight and structural stability. Correlating such variables in my experience is problematic since they are all in different units – creating a formula to balance structural stability against cost gives a false sense of authority to the process. It is here that the Genetic Staircase is unsuccessful: it is great at explaining how a genetic algorithm works, but it does not demonstrate why the genetic algorithm could be better than the normative design methods. Then again I could have this all wrong and the genetic algorithm may just be there as an advertising device. If it is, Caliper Studio have done a wonderful job capturing the attention of non-geeks. A bold and exciting first step by Caliper Studio. See also: Caliper Studio's website; Pictures of the genetic staircase on flickr. Cover photo via flickr, taken by Ty Cole.","source":"https://www.danieldavis.com/genetic-staircase-by-caliper-studi/"},{"name":"comment","citeyear":2020,"type":"other","value":["Interesting commentry. I'm interested in further discussion/debate on strategies such as Genetic Algorithms used in design, as are a few other colleagues at Expedition. If you're after another application of this, you might be interested in this project we worked on, the Santa Maria Del Pianto Station. http://www.expedition.uk.com/index.php?pid=141","This project usually evokes opinions and discussions when raised.","Cheers,","Jon","Thanks for the link Jon, I have not come across that project before. It is interesting to see the Expedition project explicitly state the objectives of the Genetic Algorithm (especially compared to Caliper Studio's staircase). I find that a much more successful approach, even though it is perhaps not as intuitive as the staircase.","Also your blog http://geometrygym.blogspot.com/ has some really useful stuff on it.","Daniel"],"source":"https://www.danieldavis.com/genetic-staircase-by-caliper-studi/"}]}

**Screenshots**

Unavailable due to pure backend application

**Prototypes and timelines**

The implementation plan from this moment on for the crawler services are elaborated below:

[M20] 1st prototype: Development and integration of Resource Filtering, after the indication from UPF regarding the desired/unwanted content, and gRPC (google Remote Procedure Call) communication establishment with the Data Storage Instance.

[M28]: Further development and allocation of an updated version regarding the website and the social media crawling and scrapping functionalities.

[M34]: Final version of the updated modules inside the component and finalization of the framework.

## 2.3 Data Storage

### 2.3.1 File System

The MindSpaces File storage is the main storage that is used by all the components to save platform specific content. The data storage will save, all the projects, users and the 3D

models along with the metadata of the project. The data storage saves all the information on the experiments and the configuration on the 3D models.

The file storage is developed using PHP and includes an SQL relational database. Figure 32, describes the Data Model for the file storage

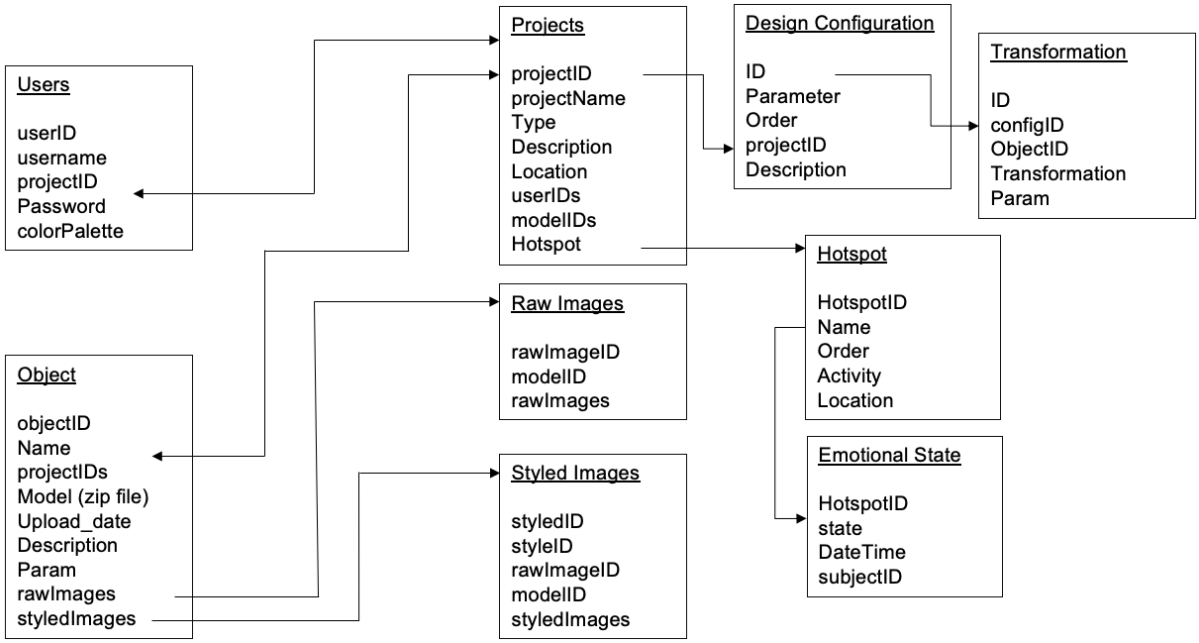



Figure 32 Data Model for the File Storage

The File system is connected to the rest of the system's components by powerful RESTful APIs that can be accessed by the tools and the users. The tools can connect using HTTPS calls to the File Storage. The Swagger GUI for the file storage can be accessed here: **https://mindspaces-integration.nurogate.com/Server/Swagger/**

Figure 33 represents the screenshot of the Swagger GUI.


Figure 33 GUI of Swagger

The File system allows the following functionalities:

- USER related functionalities:
  - Create new user
  - Login to a user account
  - Get info on existing user

- Update user info
- PROJECT related functionalities:
  - Create new project
  - Get information on an existing project
  - Delete an existing project
  - Assign the specified users to the project
  - Remove the specified users from the project
- MODEL related functionalities:
  - Add a new model to a project
  - Get info about a model
  - Update a model
  - Delete an existing model
  - Add a raw image to the model
  - Get info on the raw image
  - Update info on the raw image
  - Delete a raw image
  - Add a new styled image
  - Get info on the styled image
  - Update info of the styled image
  - Delete styled image
- DESIGN CONFIGURATION related functionalities:
  - Add new design configuration
  - Get info about the design configuration
  - Update a design configuration
  - Delete a design configuration
  - Add a new transformation to the design configuration
  - Get info on existing transformation
  - Update the info of existing transformation
  - Delete a transformation
- HOTSPOT related functionalities:
  - Add a new hotspot
  - Get info on existing hotspot
  - Update info of hotspot
  - Deletes an existing hotspot
  - Add new emotional state
  - Get info on existing emotional state
  - Updates on info on existing emotional state
  - Delete an emotional state

The File Storage tool uses secure HTTP requests to handle the transfer of information from the end user tools to the system, therefore the system is secured by TLS (Transport Layer Security). Following that, the File Storage tool uses an OAuth[2] token system to create a session ID that is needed to request any information from the server or send any information to the server.

[2] https://en.wikipedia.org/wiki/OAuth

### 2.3.2 SOLR and MongoDB

In order to complement the Knowledge Base (with its extensive capabilities for ontologically defined data and reasoning) with support for less structured or document-oriented data, a Solr database working in conjunction with MongoDB has been implemented in the MindSpaces platform. These databases are currently used mainly for the outputs of the crawler as well as Text Analysis but can potentially include additional data sources in an integrated fashion.

Solr and MongoDB work together in a complementary fashion, with MongoDB providing support for storing documents in a structured way (with a JSON-based structure), allowing for complex hierarchical structures when required, without the need for a previously defined rigid data schema. This makes it possible to easily store crawled data, including all associated metadata from the content source in its original format, along with the output from Text Analysis for the given document, again in a structured form including nested subsections, etc. It also supports adding any other related data to the document, following whatever structure may be required to represent that information. In addition to the flexibility with regard to the document structure, MongoDB provides highly efficient access for writing, reading and updating the documents.

Solr on the other hand is used as an index on top of that data, providing useful functionalities for retrieval and visualization tasks, including highly efficient statistics on occurrences and co-occurrences of terms in the index (facets) that make it easy to build e.g. tag clouds or relation graphs on top of large document collections. Contrary to MongoDB, Solr relies on a manually defined data schema (with some support for dynamic fields for added flexibility) and does not support complex nested structures.

As a result, MongoDB provides the means to store any kind of complex data with minimal setup, whereas Solr provides highly efficient indices that are specifically designed and optimized for a specific application or access pattern.

The data in Solr is automatically synchronized from MongoDB using mongo-connector[3], meaning that components such as the Crawler or Text Analysis only need to write to MongoDB, with the appropriate information reflected with minimal delay through Solr and available to visualizations and other user interfaces. The data is MongoDB is structured with a section designed specifically for indexing with Solr, following a simplified “flat” structure that corresponds to the definition of the Solr data schema defined for the MindSpaces project.

Both databases are deployed as Docker containers through Docker Swarm on the central project server, with data directories mounted directly from the host file system for persistence. This application stack, which contains Solr, MongoDB, mongo-connector, as well as GRPC endpoints needed for communication with and between the related components (the Crawler and Text Analysis), is replicated three times: the “production” stack which represents the full stack with all analysis pipelines for the final project use, a “development” stack which is a replica of the former but intended to be used for tests (with possibly incomplete or unfinished pipelines), and finally a “data collection” stack which is a simplified version that does not include any analysis and only stores the data collected by the Crawler.

[3] https://github.com/yougov/mongo-connector

The “data collection” stack was introduced to simplify the collection of large amounts of data independently of the status of the analysis pipelines (or their computational cost). Data can then be replayed from the “data collection” stack to one of the other ones in order to undergo full analysis while maintaining the exact same communication interfaces, making it easier to iterate and improve the analysis pipeline throughout the project.

## 2.4 MindSpaces Design Tool

### 2.4.1 Concept

The Mindspaces Design Tool was conceptualized in order to meet the requirements of different stakeholders involved in the design process that underlies the transformation of spaces through artistic and architectural interventions, which intends to maximize or minimize the impact of specific concerns on the emotional state end behaviour of people that dwell in these spaces.

From a conceptual standpoint, the Mindspaces Design Tool enables the designer or architect to connect his or her design prototypes, suggestions and solutions to a data-gathering and analysis environment, in order to test and validate these prototypes and to refine them. Furthermore, it allows the user to identify which specific solutions work better to maximize or minimize the impact of selected concerns, such as privacy, wellbeing, fostering collaboration, comfort, relaxation, etc.. Therefore it helps to optimize the artistic and architectural interventions intended in the space.

The user profile that the Mindspaces Design Tool targets is someone from a design background active in conceptualizing and developing solutions befitting the design spaces defined by the Mindspaces project. Therefore, architects, interior designers, artists, urban designers, and other similar profiles all are potentially targeted under this definition of the user profile. The user is knowledgeable about the design processes involved in such an endeavour, and uses a series of digital tools to model, design, and execute these processes, and produce design prototypes. The Mindspaces Design Tool does not intend to replace the existing toolsets for supporting Design, but rather complement these tools with additional functionalities that facilitate the testing, validation, and optimization of design solutions that are developed using other classic design frameworks (e.g. CAD software).

In order to generate, test, validate, and optimize design Solutions, designers are generally expected to have some background in computational geometry and computational Design, given the state of the art of current computationally driven Design tools. This is true in many cases, however a portion of the user base that is potentially targeted by the Mindspaces project, and involved in the process of transforming spaces through designing and implementing interventions of artistic and architectural in them, may not have consolidated know-how in this area, or may not be able to access such area of expertise effectively. The Mindspaces Design Tool pretends to reduce the knowledge requirements for such profiles, and facilitate knowledge acquisition about computational geometry and design. The tool streamlines the computations and processes involved in generating, testing, validating, and optimizing design Solutions according to the expected behavioral and emotional impact that they would have on the people dwelling in the targeted space.

The Mindspaces Design Tool plays a central role in the Mindspaces´ ecosystem of technologies, services, and tools. It connects to and integrates different services, such as the semantic analysis services, the Aesthetics and texturing, the knowledge base, the simulation tool, among others. It also connects with the GrassHandler machine, and provides interfacing capacity for this machine. The GrassHandler machine acts as a back-end support system for the tool, housing computationally intensive algorithms that can execute seamlessly the cloud rather than otherwise on the user machine. This allows the tool to concentrate on supporting the graphical and Interactive functionalities required to support the user experience and processes intended. This dual design allows for separating computationally intensive aspects from interactive aspects, and has several advantages beyond allowing the tool to centre on user-related and interaction-related concerns. For instance, it allows the computational resources required for supporting the design process (in particular the generation, testing, validation, and optimization of design Solutions) to scale beyond the user machine. This concept is illustrated in the following diagram.

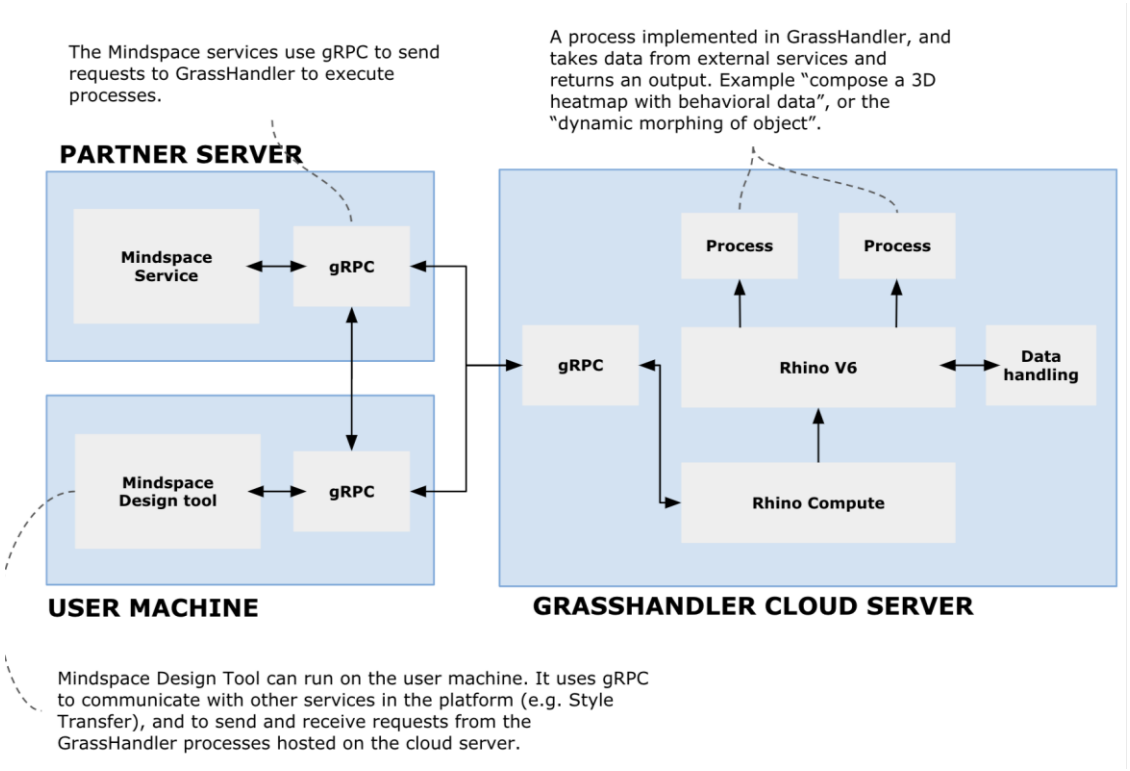



Figure 34 The Design Tool in sync with GrassHandler and other services.

### 2.4.2 **Gathering and addressing requirements**

The interface and interaction designs of the Mindspaces Design Tool have been formulated in order to satisfy the requirements behind the conceptualization and development of this tool. Before developing the wire frames and Technical architecture of the tool, the requirements were identified and analyzed. In order to identify these requirements, the user and technical requirements defined and listed in the context of the deliverable D.7.1 have been taken as a basic framework to understand the scope and area of application of the tool. However, more specific requirements are needed to define the processes that the tool

is expected to implement, especially with relation to the contemplated use of the tool in resolving design problems, such as those defined and delimited by the Project's use cases. For this purpose, interviews, discussions, and conceptualization exercises have been conducted with a total of 9 representatives of the stakeholder group targeted as a user base for the Mindspaces Design Tool. This included five members of the project Consortium, in addition to four artists that have won the project's artistic residency call. In addition, the definition of these requirements draws on our knowledge and know-how related to the expectations of users involved in the design domain.

The first of these requirements centers on visualizing complex and computationally oriented models that represent, digitize, or virtualize the space in which the designer intends to apply artistic or architectural interventions. The models are parametrized to facilitate the process of defining possible optimizations as well as generative components for the design solution, and identify inherent stress points with ease.

The second of these requirements centers on supporting the virtualization of relevant data streams, and data sets, related to the discourse, the behavior, and the emotional states, of dwellers in the targeted space, over or on the top of the virtualized environment that represent the targeted space data layers are visualized to indicate correlation between the concerns represented by the data, and captured through sensing or scanning mechanisms, and the geometry of the space. The user can utilize the visualized data to infer such correlations, and identify hotspots, stress points, and areas of Interest in the space targeted by the intended design intervention.

The third of these requirements centers on helping the user to define and generate different design configurations for the intended design intervention. These design configurations explore possible variations, alternations, and alternatives along specific design parameters, such as mobility in the space, user comfort, illumination, privacy, and decoration. The design configurations are utilized by other tools and services to orient and focus data acquisition activities and analytical activities pertaining to the testing, validation, and optimization of design Solutions along defined parameters.

The fourth of these requirements centers on supporting the engagement of users with monitoring and data collection mechanisms developed in the context of the project in order to acquire information on the design context (in other words, the design space of a project), as well as to test different design configurations in real time. In this sense, and based on the capacity of the tool to process and visualize real-time data using the grass Handler machine as a back-end service, The tool would support real-time monitoring of behavioral and emotional analysis and experimentations as they happen, and with that allow the user to intervene or interact with such dynamics by setting, configuring, or generating design configurations during data collection and analysis.

In addition to these high level requirements, other more clear-cut requirements defining some of the functionalities expected to be implemented in the tool have also been identified. These include functions related to data management and visualization, transformations expected to be supported by the tool, and paradigms related to the management and manipulation of 3D assets inside the tool.

These requirements are being taken into consideration in the conceptualization and development of the tool. In addition, a continuous engagement with the representatives of the user group is undertaken under a co-design approach in which users have access to early prototypes, and are constantly providing feedback, comments, and suggestions for guiding the development of the tool.

### 2.4.3 Main underlying technological framework

The tool's technology is based on Rhino, a leading 3D computer graphics and computer-aided design (CAD) application software developed by Robert McNeel & Associates. Rhino uses the NURBS mathematical model for its geometry, to produce mathematically precise representation of curves and freeform surfaces in computer graphics (as opposed to polygon mesh-based applications). Rhino is a popular tool for computer-aided design (CAD), computer-aided manufacturing (CAM), rapid prototyping, 3D printing and reverse engineering in industries including architecture, industrial design, and virtually all of the Design disciplines.

The tool makes use of the Rhino3DM library (https://github.com/mcneel/rhino3dm), which is a powerful framework for building interactions with all types of geometries supported in Rhino. This includes points, point clouds, NURBS curves and surfaces, polysurfaces (B-Reps), meshes, annotations, extrusions, SubDs, among other geometric components. The library also supports the manipulation of Rhino objects of non-geometry nature, like layers, object attributes, transforms, and viewports. Also, the library supports parsing and saving .3dm file format. Finally, the library implements and streamlines communication between the tool and the GrassHandler machine.

### 2.4.4 Supported data model

The data model that governs the tool's technical framework has been designed and represented in the following figure. It shows how the tool interrelates different objects of information, in particular how it connects non-geometric or non-visual data such as behavioral data, discourse data, emotional states, with hotspots, design configurations, and 3D objects. Also, as evident in the model, the project created inside the design tool is able to integrate and bring together the different elements of knowledge and information created in the project.

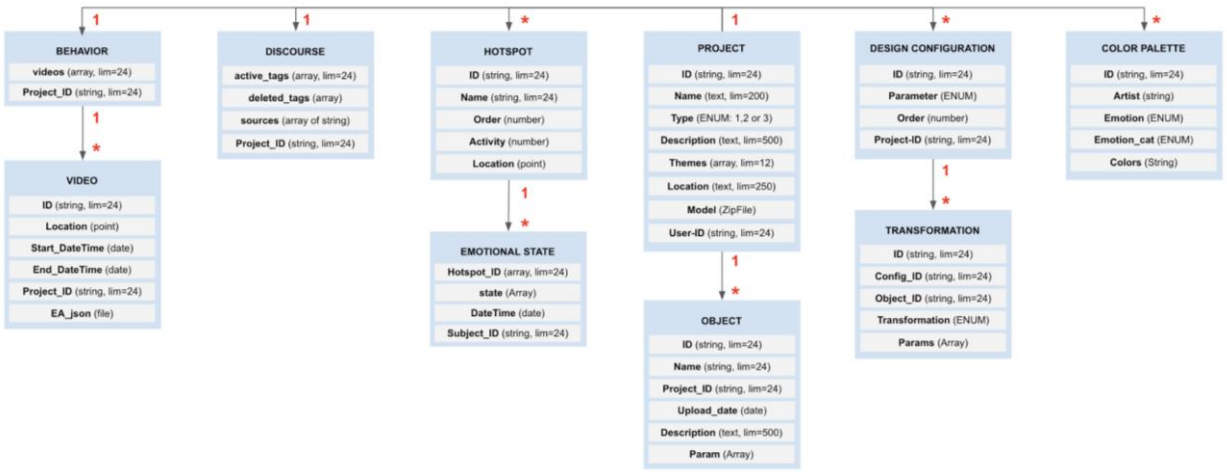



Figure 35 Relational Data Model for Design Tool

### 2.4.5 **Implemented functionalities and components**

The first version of the tool establishes the implementation framework required for its development, and creates a large part of the different components envisioned under this version, a first-version tool shell has been developed and is able to communicate with other services and components to acquire and send relevant data.

In particular, the tool interfacing mechanisms with the Rhino platform have been created. A complete parser able to transform complex 3DM models into structured 3D objects that can be visualized in the tool's renderer. The parser supports the transformation of meshes, arcs, polylines, layers, groups, instances, textures, Nurbs, and complex geometric transformations, enabling the tool to visualize and manipulate elements of the loaded design solution in a minute and detailed fashion. In essence, the parser allows the Rhino platform to interface effectively with other technologies and platforms for 3D visualization. The interfacing mechanism implemented to connect the tool with the Rhino platform is the integration with the GrassHandler machine, under which the tool can send geometric data to the machine for real time analysis and transformation.

Accordingly, complex 3dm files can be loaded and parsed by the Mindspaces Design Tool into parameterised models, Breaking down the constituents of the 3DM model into objects that can be manipulated easily and wholesomely by the designer through the design tool. This process involves transforming different components of computational nature such as NURBS curves into geometric representations that can be easily manipulated and the tool's GUI. In addition, different layers existing in the 3dm definition are handled accordingly and fused into a single base layer where all objects are treated identically. Finally, instantiations and transformations defined in the 3dm definition are also applied, and accordingly each instance of an object is defined as an independent but identical copy of the original object class from which it has been instantiated.

A first-version graphic user interface has also been created and developed to support the processes implemented so far. The tool's graphic user interface is composed of different sections or contexts, each specialized in the specific process or subprocess related to the functionalities that the tool aims to support.

The first context is the environment context where the base model of the space related to the user project is visualized for inspection. From this context, the user can upload the Project's base model and verify that it is well compatible with the Mindspaces Design Tool. The base model is imported in 3dm file format or obj file format, both supported by the Mindspaces Design Tool. The environment context includes a powerful renderer able to visualize complex 3D models in a light way. It supports free and smooth navigation in the 3D space, allowing the user to position the camera in any perspective and in any position desired. This facilitates the space exploration and reasoning about possible design configurations that can help to optimize the space design. On top of the visualized model, the user can superpose a 3D heatmap generated on the basis of the behavioral analysis data acquired from other services. The visualization of the behavioral analysis data enables the user to identify hotspots in the environment, and to define these hotspots for further analysis, and for other services, namely for the VR EEG machine that uses these hot spots to position subjects in the virtual environment while acquiring EEG data.

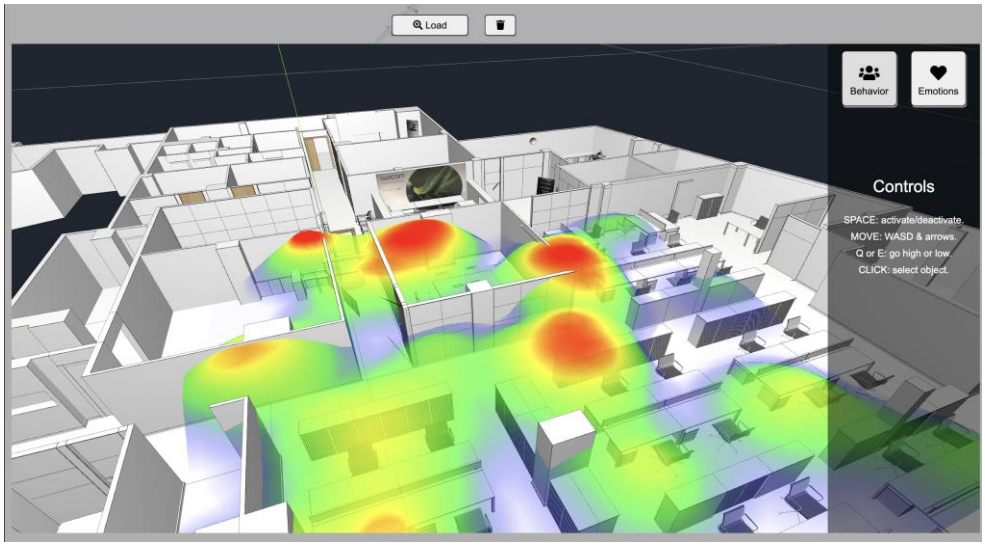



Figure 36 behavioral analysis data visualized

The interface also includes a library of 3D objects that the user can customize in terms of colors, textures, and other aspects. Then the user can include these objects in the design configurations, creating new configurations from existing ones with ease. The user can import object models designed in other tools into the object library, and can also extract objects from the 3D scene of the virtualized design space. The object Library provides management capacities for assets related to the creation of the design configurations and the design tool. The assets can be classified by parameter, and under each parameter a set of assets is created and used when designing new configurations.

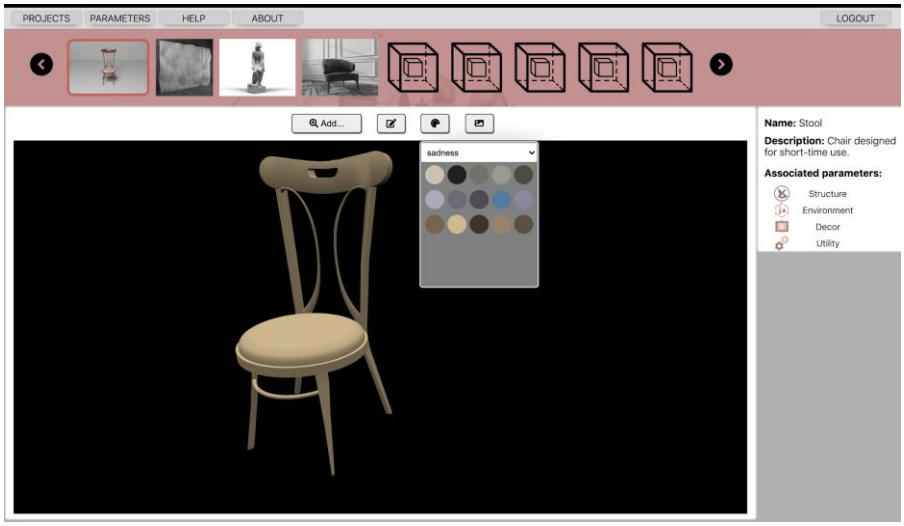



Figure 37 Layout of the object library

Using the tool, the user can create new projects, and for each project a design space is initiated. When a new project is created, the user uploads the base model of the space in which the design configurations will be created. The base model may be composed of different 3dm files, each capturing a conceptual layer in the design, such as structure, furniture, and decorations or accessories. The tool loads all these files into a single scene to showcase the virtualized environment. The user then navigates the space and explores it in order to plan different design solutions. Under each project, a set of parameters is activated. Currently, the tool supports eight predefined parameters, but in the future it will support more custom-defined parameters. When a parameter is activated, the tool allows the user to access the design configuration editor where the user can create new configurations. when the user accesses the editor, the set of objects or assets related to the specific parameter selected are displayed, facilitating the access to relevant components.

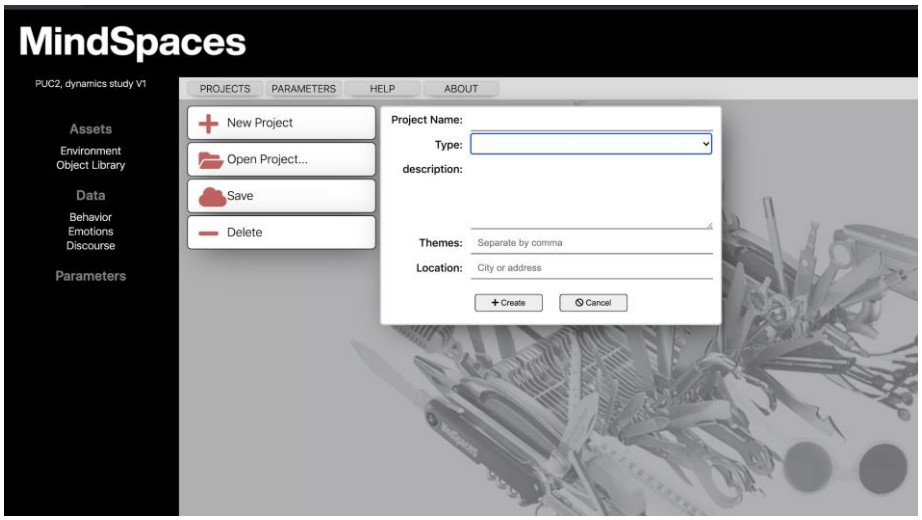



Figure 38 Project creation and management

The design configuration editor allows the user to select objects or to introduce new objects in the scene, and then apply Transformations on these objects. These transformations are executed in virtual reality in order to change from one configuration to the other.

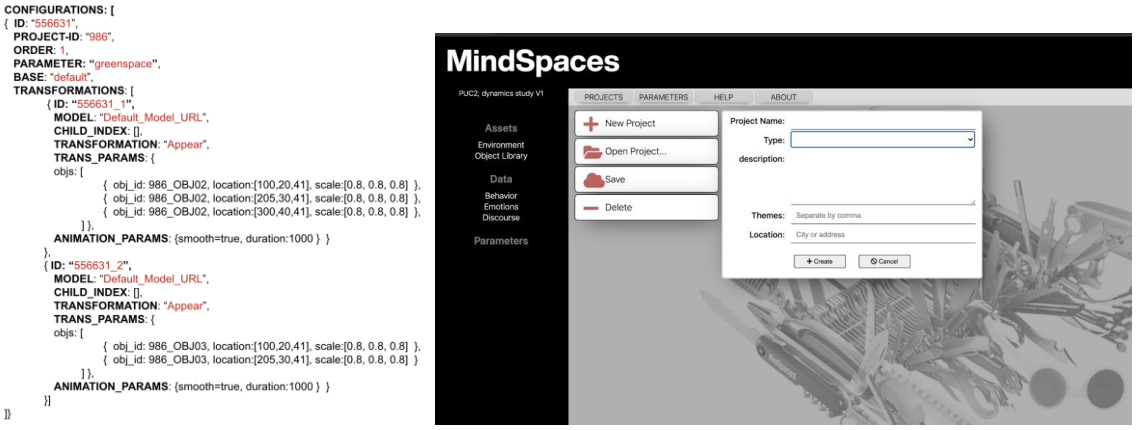



Figure 39 Design configuration editor and resulting JSON

A preliminary integration of the discourse data has been completed. The discourse data is visualized as a graph of interrelated concepts that represent the online discourse around a set of concerns. It is intended to showcase the discourse data at this stage in the development of the Mindspaces platform, and will be elaborated further in upcoming versions.

In addition, version-one tool includes communication components that allows it to exchange data from the data warehouse, to save and retrieve project information, including objects, configurations, hotspots, and others. The communication components also connect with other services in the platform, such as the Palettes and the Styling service, among others.

### 2.4.6 The GrassHandler machine

**Concept**

The GrassHandler is a cloud machine that hosts the Rhino V6 platform and allows external tools and software to access the functionalities of this platform remotely, as a service. In other words, the GrassHandler machine wraps the Rhino platform with an Application Program Interface, allowing other external components to remotely execute Rhino processes with their own data, and in real-time. In addition, users can host different Grasshopper programs tailored to support specific generations, transformations, and computations developed to meet the requirements and necessities of artistic and Architectural interventions contemplated in the context of the project. In addition, the GrassHander machine supports Nurbs-related computations, such as Closest point calculations, Intersection calculations, Surface tessellation (meshing), area calculation, among others.

The GrassHandler Cloud machine has a dual role in the project from a technical stance. First, it supports the design tool with advanced and powerful computations, and offers a unique set of functionalities from the Rhino suite. Second, it allows Advanced users to deploy their algorithms and plugins designed to support their artistic or architectural interventions on the cloud, and access these algorithms from the design tool as well as from any other tool or component developed for the purpose of supporting such interventions.

**Main underlying technological framework**

The GrassHandler Cloud machine is constructed around the Rhino Compute solution, which allows accessing Rhino and Grasshopper through a stateless REST API, in order to manipulate two and three-dimensional curves, surfaces, and solids. In this context, and by using the Compute solution, 2400+ geometric operations on custom objects supported from within Rhino become supported to manipulate 3D models effectively, efficiently, and remotely.

**Implemented functionalities and components**

Currently, the machine architecture of GrassHandler has been implemented. This includes the deployment of the operating system on the cloud machine, and the installation of Rhino and Grasshopper V6. in addition, the Compute solution was deployed and connected both to the Rhino instance and to the web, listening on a port for requests from other tools and services. Also, A module that supports grpc communication has been developed and deployed, In addition to a set of simple Grasshopper transformative programs able to execute the Transformations supported by the design tool on the objects has been implemented and deployed as well.

The GrassHandler machine has been functionally tested through a battery of tests conducted to evaluate its performance both in terms of processing and communication. The best included simulations of complex cases involving large models, parallel requests, large request queues, and other paradigms. The results of the tests allowed for adjusting the configuration of the GrassHandler machine in order to optimize its performance on both ends (processing and communication). The tests also show that the machine is now stable and can be used easily by other tools to execute complex computations in real time and in the cloud. As a cloud machine, the resources of grass Handler are scalable and can be increased in the future to support complex use cases and situations according to the needs of the user projects.

The following figure shows a screen capture of the remote desktop of GrassHandler cloud machine, and the queue of requests and responses for a mesh transformation service deployed on the grasshopper instance within the machine. The service is accessed remotely from another tool running on another machine.

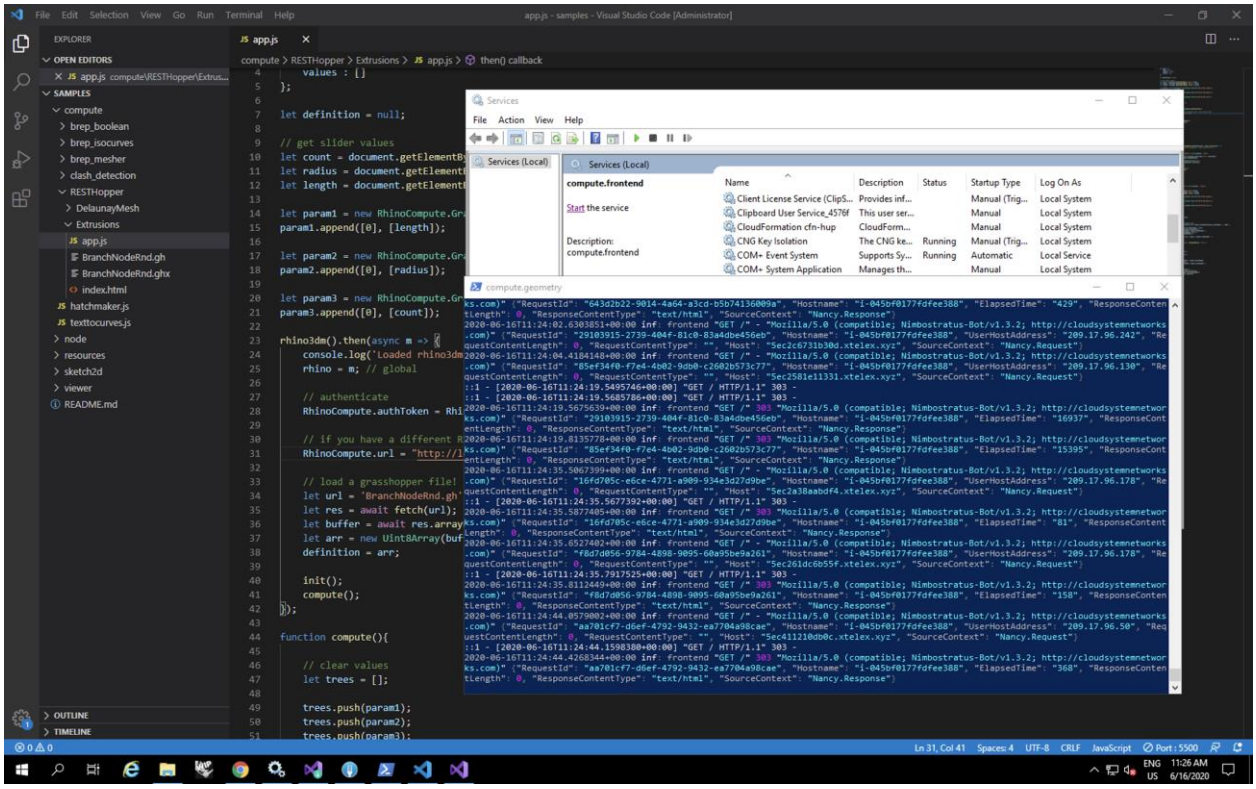


Figure 40 Screenshot of GrassHandler's desktop

The power of the GrassHandler Cloud machine can be better discerned once it is connected to live data streams from sensors and data collection mechanisms that hardness data relevant to the design artistic and Architectural interventions in the spaces targeted in the project. This connection is contemplated in the upcoming phase of the project, once these services and mechanisms have matured and are working in real time. GrassHandler allows to process this data in real-time and dynamically use it to alter and change aspects in the design configurations of the targeted space. For instance, it would be possible to support the dynamic behaviour of specific design components, such as the distribution of furniture in the space, or the intensity of the lighting in specific areas, or any other similar dynamic behaviour.

## 2.5 MindSpaces VR Tool

| **Name of the component: VR Tool** |
| --- |
| **Description of the component** |
| The VR tool acts as user-facing application that lets users experience the MindSpaces environments in Virtual Reality. Furthermore, it supports user experiments where different design configurations of the environments are shown to experiment participants in VR who are also subject to an EEG analysis at the same time, in order to gain insight into the perception of the different configurations by different people.<br><br>Therefore, the VR tool executes the following responsibilities:<br>1) Provide a user interface to start an experiment<br>2) Download all necessary model data from the file storage |

3) Download all necessary configuration data from the file storage, i.e., design configurations and hotspots (locations in the virtual environment that are expected to be of interest and where experiment participants are placed)
4) Run the experiment. While it is running,
    a. Accept connections from the Emotional Analysis component
    b. Forward data sent by the Emotional Analysis component to the Knowledge Base, along with state and user information (the user's position in the virtual environment, the current design configuration, experiment information)
    c. React to a response by the Knowledge Base, by
        i. Possibly changing the design configuration, i.e., changing the
        ii. Possibly transporting the user to a different location in the environment
    d. Report data sent by the Emotional Analysis component, enriched with state and user information, to the File Storage for later analysis

When started, the VR tool prompts the user to login (using their file storage credentials), whereupon a list of this user's projects are displayed. After selecting a project, all necessary data is downloaded and loaded, additional experiment data (a experiment and subject ID to differentiate results in later analysis) can be specified, and the experiment can be started.

While the experiment is running, the VR tool runs a gRPC server to accept local connections from the Emotional Analysis tool (see below). When data arrives using this connection, the data, enriched with state and user information, is sent to the Knowledge Base via a separate gRPC connection, as well as to the file storage via its REST API (see below). The response from the Knowledge Base is interpreted as command, to change the design configuration and/or move the user.

Design configurations describe different arrangements of the virtual scene the user is experiencing, e.g., different furniture or wall placements or different color setups. Since in many cases, only parts of the scene changes between similar configurations (e.g., there might be four different layouts for desks, while everything else remains unchanged), design configurations are organized hierarchically. In this hierarchy, design configurations can load a base model, and extend their "parent" configuration by a set of *transformations*. In the first prototype, the following transformations are implemented:

- Appear: an object is added to the base model or parent configuration
- Remove: objects are removed from the base model or parent configuration
- Move: objects are moved, rotated, or scaled
- Change Color: the primary color of an object or set of objects is changed (textured objects are tinted)

Despite the hierarchical structure, the VR tool can change from any design configuration to any other configuration. To do so without introducing lags which might degrade the experience, the smallest set of necessary changes between the old and new design configurations is computed and executed.
Furthermore, in order to avoid startling the user, the changes can be animated. In the first prototype, only simple animations are implemented, i.e., for the Appear and Remove transformations, the objects' opacity is modulated linearly, the Move transformation

animation interpolates the objects' position and scale linearly (and the rotation spherically linearly), while the Change Color transformation linearly interpolates between the colors in RGB space. However, the set of possible animations can and will be extended in future iterations.
In the screenshots (below), an example of an Appear and a Remove transformation is shown.

If a command to move the user is received from the Knowledge Base, the user is teleported to the new position to avoid any issues with cybersickness, shortly fading to black while doing so to avoid startling them. The target of a movement is always a hotspot, such that emotional analysis data can always be related to a certain, semantically meaningful position (e.g., in the kitchen, in the hallway).
In addition, the user can always move by moving or rotating their head, to ensure an accurate experience of the environment respecting typical motion cues. While it would also be possible to physically walk to a different place in the virtual scene, this possibility is not currently foreseen in the experimental setup to avoid affecting the EEG sensors by excessive noise.

**Technical Requirements addressed**

| | | |
|---|---|---|
| UR_5 | HLUR_1 | As an Architect I want to be able to geolocate the biometric and behavioural data in virtual space |
| UR_11 | HLUR_2 | As an architect I want to correlate the amount / quality of light with the behaviour and/or emotional state of users in designed workplace environments |
| UR_13 | HLUR_2 | As an architect I want to correlate the ceiling / spatial height with the emotional state and/or behaviour of users in designed workplace environments |
| UR_14 | HLUR_2 | As an architect I want to correlate the size/shape of a personal working desk/space with the emotional state and/or behaviour of users in designed workplace environments |
| UR_15 | HLUR_2 | As an architect I want to correlate amenities with the emotional state and/or behaviour of users in designed workplace environments |
| UR_19 | HLUR_2 | As an artist I want to be able to use VR as a simulation environment to test new designs |
| UR_68 | HLUR_10 | As an architect I want an easier way to develop 3D environments than traditional tools |

**Input Data with sample**

**Emotional Analysis**
The VR tool provides the following gRPC service to accept data from the Emotional Analysis:

```
service EmotionRecognition {
    rpc SendEmotionalTag(EmotionalTag) returns (ACK) {}
}
message EmotionalTag{
  int32 emotional_state = 1;
  int32 valence = 2;
  int32 arousal = 3;
  int32 time = 4;
}
```

```
message ACK {
  int32 ack_tag = 1;
}
```

**Knowledge Base**

It uses the following gRPC service of the Knowledge Base to send Emotional Analysis and user state information and receive a response:

```
service KnowledgeBase {
      rpc GetLatestChange (GetChangeRequest) returns (GetChangeResponse){}
}

message GetChangeResponse {
      bool change_configuration = 1;
      int32 configuration_id = 2;
      int32 hotspot_id = 3;
}

message GetChangeRequest{
      int32 project_id = 1;
      int32 current_configuration_id = 2;
      string subject_id = 3;
      string experiment_id = 4;
      emotional_state state = 5;
      location_info location = 6;
}

message emotional_state{
      int64 timestamp = 1;
      int32 eeg_tag = 2;
      int32 valence = 3;
      int32 arousal = 4;
}

message location_info {
      int32 hotspot_id = 1;
      float pos_x = 2;
      float pos_y = 3;
      float pos_z = 4;
}
```

In the request, the project_id corresponds to the project loaded by the VR tool, the current_configuration_id refers to the currently active design configuration. subject_id and experiment_id are strings chosen by the user of the VR tool to identify the experiment and experimental subject for later analysis. In location, the user's current location in the virtual environment, as well as the corresponding hotspot id are transmitted.

**File Storage**

The VR tool downloads project information from the File Storage using its /Project/{ProjectId} API:

```
{
  "ProjectId": int32,
  "ProjectName": string,
  "Type": int32,
```

```
  "Description": string,
  "Themes": [ string ],
  "Location": string,
  "UserIds": [ int32 ],
  "ModelIds": [ int32 ],
  "DesignConfigurationIds": [ int32 ],
  "HotspotIds": [ int32 ]
}
```

An example for a test project is the following:

```
{
  "ProjectId": 16,
  "ProjectName": "McNeel Office Barcelona",
  "Type": 0,
  "Description": "Model of McNeel's office in Barcelona.",
  "Themes": [],
  "Location": "Barcelona",
  "UserIds": [ 3 ],
  "ModelIds": [ 19, 20 ],
  "DesignConfigurationIds": [ 6, 7, 8 ],
  "HotspotIds": [ 7, 8, 9, 10, 11, 12 ]
}
```

For each model id in ModelIds, the model information is downloaded using the /Model/{ModelId} API:

```
{
  "ModelId": int32,
  "ProjectId": int32,
  "ModelName": string,
  "Description": string,
  "Params": string,
  "UploadDate": "2020-02-20T01:02:03+00:00",
  "DownloadUrl": string,
  "RawImageIds": [ int32 ],
  "StyledImageIds": [ int32 ]
}
```

The model data itself is downloaded using the provided DownloadUrl. It is a zip file containing one or several FBX model files, or alternatively, one or several Unity AssetBundles.

For each design configuration id, the design configuration information is downloaded using the /DesignConfiguration/{DesignConfigurationId} API:

```
{
  "DesignConfigurationId": int32,
  "ProjectId": int32,
  "BaseModelId": int32,
  "BaseConfigurationId": int32,
  "Parameter": string,
  "Order": int32,
  "TransformationIds": [ int32 ]
}
```

where BaseModelId refers to the model that is loaded as a base in this configuration (e.g., one version of an office floor), and BaseConfigurationId specifies the parent of this design configuration in the hierarchy.

For each transformation id, the transformation information is downloaded using the /Transformation/{TransformationId} API:

```
{
  "TransformationId": int32,
  "DesignConfigurationId": int32,
  "ModelId": int32,
  "TransformationType": string,
  "Params": { (JSON substructure) }
}
```

Here is an example of a transformation that removes a number of objects from the scene represented by model id 19 in a fade-out animation that takes 2000 milliseconds:

```
{
  "TransformationId": 6,
  "DesignConfigurationId": 7,
  "ModelId": 19,
  "TransformationType": "Remove",
  "Params": {
    "subobjects": [
      "物件_3436",
      "物件_3443",
      "物件_3445",
      "物件_3442",
      "物件_3444"
    ],
    "animation": {
      "animated": true,
      "duration": 2000
    }
  }
}
```

For each hotspot id, the hotspot information is downloaded using the /Hotspot/{HotspotId} API:

```
{
  "HotspotId": int32,
  "ProjectId": int32,
  "Name": string,
  "Parameters": [ string ],
  "Activity": string,
  "Location": [ float, float, float ],
  "EmotionalStateIds": [ int32 ]
}
```

The VR tool also conducts a login, sending the following structure to the /UserLogin API:

```
{
  "UserName": string,
  "Password": string
}
```

and receiving a SessionId:

```
{
  "Result": "Success",
  "SessionId": string
```

```
}
```

**Output Data with sample**

For data sent to the Knowledge Base, see Input (since this is a two-way communication).

Whenever an emotional state is received from Emotional Analysis, it is transmitted to the File Storage for later analysis using the /Hotspot/{HotspotId}/EmotionalState API (where {HotspotId} is the id of the hotspot where the user is located in the virtual environment):

```
{
  "State": int32,
  "DateTime": int64,
  "SubjectId": string,
  "DesignConfigurationId": int32
}
```

where State indicates the current emotional state, DateTime is a Unix timestamp, SubjectId identifies the subject (based on the information provided by the user), and DesignConfigurationId refers to the currently active design configuration.

**Screenshots**

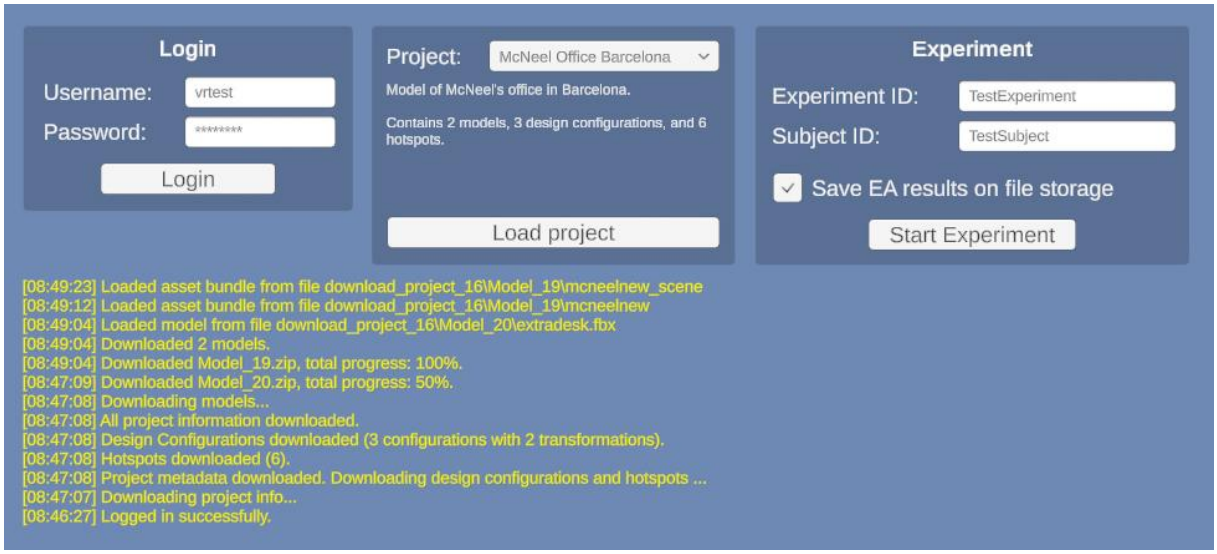



Figure 41 The prototypical login interface of the VR tool, allowing to load a project from the file storage and start an experiment.

Figure 42 In the process of switching to a different design configuration, a structural element of the environment is removed. Left: before, Center: during animation, Right: after

Figure 43 In the process of changing design configurations, a desk is added to the room. Left: before, Center: during animation, Right: after

# 3 DEMONSTRATION URLS AND INFORMATION

## 3.1 MindSpaces design tool demonstration

The demonstration video can be found here: **https://drive.google.com/file/d/1Wmv8GzKLBBdDcIIUay1P_EgFbMzOS_U8/view?usp=sharing**

The demonstration shows the following aspects fo the tool

- Login
- Create project
- Design parameters
- Load models from the user's library
- Work with 3D Environments
- View various analysis data
- Create hotspots
- Select objects in an environment and modify them
- Add new objects to the environment
- Edit objects individually
- Create configurations

## 3.2 ABPS Generative Design and Behavioural Simulation Tools Demo Description

### 3.2.1 Simulation Data

The demonstration video can be found here: **https://drive.google.com/file/d/1SMBOxi8_QuJhpeeCrmOD_Gr94ohHP1tb/view?usp=sharing**

**ABPS_DemoVid1_Design_Simulation_Data**

- *Design in Rhino/Grasshopper (0-15 seconds)*

First, the demo shows a generic corporate office 3d model is loaded in Rhino/Grasshopper. In Grasshopper we show the process of using our the Grasshopper2Unity plugin with the shared Mindspaces Asset Library to select 3d assets as Rhino blocks to deploy (work desks, collaborative sitting and standing tables, meeting spaces, etc.) use Grasshopper to populate the model with instances of assets. This is a parametric design process in Grasshopper. Note, using the Design Tool enables this deployment of objects from the Mindspaces Asset Library which can also replace this step.

- *Model Integration Component: Export/Import, Reconfigure / Update (15-50 seconds)*

Next, using the Grasshopper2Unity plugin, we export the model from the Rhino/Grasshopper/Design Tool environment into Json format which is read by the UnityAssetImporter plugin to bring the design into the Unity environment for behavioural simulation and VR experience. The design is reconfigured in the Grasshopper environment and updated in the Unity environment with a button press.

- *ABPS Behavioural Simulation (50 seconds-2 minutes, 16 seconds)*

Next, the ABPS Behavioural Simulation Tool runs a simulation over the imported design. The user uses the user interface to select options for deploying the agents (type, quantity, etc.) and presses build to run the simulation. A second part of the interface is used to demonstrate turning on the various spatial maps as the simulation is run including Live OccupancyMap, HistoryMaps (All, Collaborations, Walking, Conversations, Focused Working, Meetings), EncountersMap, Live SoundMap, Cumulative SoundMap, and DaylightMap.

- *Export Data to Grasshopper (2 minutes, 16 seconds-3 minutes, 8 seconds)*

Finally, the demo shows DataMaps are exported in Json format and imported into Grasshopper (Note - also importable here by the Design Tool) for the designer to use in an iterative design process.

### 3.2.2 **Simulation Experience**

The demonstration Video can be found here: **https://drive.google.com/file/d/1v8T-4KEMJcXmo2vjuL1VhVnZ3c4UYqCV/view?usp=sharing**
**ABPS_DemoVid2_Generate_Simulate_Experience**

- *ABPS Generative Design – Space Planning Example 1 (0-32 seconds)*

First, the demo shows a generic corporate office 3d model is loaded in the ABPS Generative Design Tool. On the left are the user's chosen workplace assets to be deployed by the tool. The first example shows a space planning standard example that includes 120 degree desks in the spatial organization planning. The tool recognizes the core elements and builds around them. The tool automates the search for optimal solutions, exporting a series of design layout options.

- *ABPS Generative Design – Space Planning Example 2 (32 seconds-1 minute, 1 second)*

Next, the demo shows a second selection of user's chosen workplace assets to be deployed by the tool. The second example shows a space planning standard example that is more rectilinear and tightly packed in the spatial organization planning. The tool recognizes the core elements and builds around them. The tool automates the search for a second set of optimal solutions, exporting a series of design layout options.

- *ABPS Behavioural Simulation (1 minute, 1 second-2 minutes, 2 seconds)*

Next, the demo shows the running of the ABPS Behavoural Simulation Tool on the generated design model. It shows the turning on of the various spatial maps as the simulation is run including Live OccupancyMap, HistoryMaps (All, Collaborations, Walking, Conversations, Focused Working, Meetings), EncountersMap, Live SoundMap, Cumulative SoundMap, and DaylightMap.

- *First Person Experience (2 minutes, 2 seconds – 2 minutes, 34 seconds)*

Finally, we demo that each model can be seamlessly viewed as a navigable first-person experience.

## 3.3 VR Tool

The demonstration video of the tool can be found here: **https://drive.google.com/file/d/1EmHFW5fmzLSsk4E7SLtZ7vwX-g3YTaHm/view?usp=sharing**

The following Timestamps show the following functionalities:

- 00:00 - The user logs in, and all their project information is downloaded from the Mindspaces data storage.
- 00:07 - The tool downloads all necessary information for the selected project. Since 3D models can be very large, previous downloads can be re-used.
- 00:17 - Experiment results are saved on the Mindspaces data storage, associated with the identifiers provided here.
- 00:21 - In the experiment, users can look around in the virtual environment.
- 00:29 - Based on Emotional Analysis data, the Knowledge Base can request to move the user to a different location.
- 00:42 - Similarly, the Knowledge Base can trigger different Design Configurations. In this example, a different configuration is activated that does not contain these partitions.
- 01:01 - Different configurations can also add or move elements, or vary their color.

# 4 SUMMARY AND CONCLUSION

In this deliverable we have presented the current state of the MindSpaces platform and its components. The preliminary implementation is geared to consolidating the integration model of the platform, and establish its processing cycle, as well as to provide a proof-of-concept for each envisioned technology.

Section 2 has given the overview of the system components and the architecture of the first version of the system along with the communication model for each of the services inside the platform. All the services were introduced individually along with a description, the input and output data to each of the service and the requirements they fulfil. Furthermore, where ever possible, screenshots and prototype information is provided for the service. Section 2 furthermore elaborates the backbone of the entire system with is the Data Storage, which is divided into the main File storage and a MongoDB and SOLR instance running on the same Data Storage. Lastly, the section describes the 2 main frontend tools of the system; The design tool and the VR Tool which act as a User entry point as a Designer or Architect and as an end user on whom the designs can be tested upon inside VR.

Most of the operational prototypes show basic functionalities that meet the most basic of its associated requirements. However, each prototype has validated the capacity of its related component to connect to the platform, receive data and process it, and post the results back to other components that come next in the pipeline.

The user tools illustrate how the envisioned user profiles can access and manipulate the data created by the MindSpaces platform. They represent a placeholder for the envisioned user-experience and will be refined accordingly in the next development cycle.

Section 3 documents the demonstration videos of the user tools and their functionalities for the reviewers to see.

Finally, with each of the platform modules, including services, middleware, and tools, prototyped and integrated, the work can now focus on building the pipeline that would help in the development of an end-to-end system which is addressing all functional requirements of the users to be able to conduct experiments and verify the results. In the next version, the users would be involved in improving the system and its components.