Human Brain Project

Co-funded by the European Commission

Project Number:	284941	Project Title:	Human Brain Project

Document Title:	HBP Neurorobotics Platform Specification
Document Filename[(1)]:	HBP_SP10_EPFL_140425_D10.4.1_Final.docx
Deliverable Number:	D 10.4.1
Deliverable Type:	Platform Specification
Work Package(s):	WP10.1, WP10.4
Planned Delivery Date:	M 6 / 31 March 2014
Actual Delivery Date:	M 7 / 25 April 2014

Authors:	Daniel PEPPICELLI (EPFL) Stefan DESER (TUM) Amal BENZINA (TUM) Manuel HUBER (TUM) Axel VON ARNIM (Fortiss) Alois KNOLL (TUM) Marc-Oliver GEWALTIG (EPFL) Florian RÖHRBEIN (TUM)
Editors:	NRP software engineering team (EPFL – TUM - Fortiss)
Reviewers	Celia LUTERBACHER (EPFL), Guy WILLIS (EPFL)

Abstract:	This document describes the requirements for the Human Brain Project's Neurorobotics Platform (NRP). The NRP will allow researchers to perform simulated experiments with neurobots, i.e. robotic devices with control systems based on principles of the nervous system, to gain insights in the links between low level brain circuitry and high-level cognitive and behavioural functions. It will also facilitate the development of a new type of robotic controllers derived from those insights.
Keywords:	Neurorobotics platform, Neurobot, Robotics

Document Status

Version	Date	Status	Comments
0.1	01.03.2014	Draft	Initial draft
0.2	07.03.2014	Draft	- Added CAVE definition - Modified map of Europe (compliant with template now) - Reviewed figures of Brain interfaces & body integrator - Changed RD Arch. Overview schema + minor corrections (axel) - Changes in Overall Goals of the NRP (easy transfer from sim.robot to real robot), extended roadmap (Stefan) - Added world snapshot in Exp.Viewer and comments for this in Exp.Designer (axel) - Changed minor points in overall goals use cases (removed Bill's help) (axel) - In Exp.Designer: added information about snapshot feature, introduced problem of links/references vs. big data in files; extended requirement about snapshots (Stefan) - Rephrased the "Overall Goals" of the Env.Designer (Stefan) - Added figure table. Updated figures and content tables. - Replaced user-in-the-loop by online and post-mortem analysis by offline - Inserted a "Executive Summary" (Stefan) - Update to most of the figures with the correct icons
0.3	11.03.2014	Draft	- Modified figures (added mechanic wheel to picture engines) - Updated the BIBI overview schema. - Modified figures (ESV, Exp. set-up) - WSE Overall Goals review/rephrasing - New overview figure of the BIBI - Renumbering of the chapter so that they appear in the same order as in the user workflow overview figure. - Refactored BIBI main figure to make it simple and consistent - Added VInero SP pisa location to the map in CLE chapter -Added CLE "Architectural Overview" text/figure
0.1	01.03.2014	Draft	Initial draft
0.2	07.03.2014	Draft	- Added CAVE definition - Modified map of Europe (compliant with template now) - Reviewed figures of Brain interfaces & body integrator - Changed RD Arch. Overview schema + minor corrections (axel) - Changes in Overall Goals of the NRP (easy transfer from sim.robot to real robot), extended roadmap (Stefan) - Added world snapshot in Exp.Viewer and comments for this in Exp.Designer (axel) - Changed minor points in overall goals use cases (removed Bill's help) (axel) - In Exp.Designer: added information about snapshot feature, introduced problem of links/references vs. big data in files; extended requirement about snapshots (Stefan) - Rephrased the "Overall Goals" of the Env.Designer (Stefan) - Added figure table. Updated figures and content tables. - Replaced user-in-the-loop by online and post-mortem analysis by offline - Inserted a "Executive Summary" (Stefan) - Update to most of the figures with the correct icons

0.3	11.03.2014	Draft	- Modified figures (added mechanic wheel to picture engines) - Updated the BIBI overview schema. - Modified figures (ESV, Exp. set-up) - WSE Overall Goals review/rephrasing - New overview figure of the BIBI - Renumbering of the chapter so that they appear in the same order as in the user workflow overview figure. - Refactored BIBI main figure to make it simple and consistent - Added VInero SP pisa location to the map in CLE chapter -Added CLE "Architectural Overview" text/figure
0.4	04.04.2014	Draft	Close edit/spellcheck
0.5-0.7	08.04.2014	Draft	Final review, editorial office
0.8	10.04.2014	Draft	Added a paragraph in 10.3.3 about the building of the backlog during the first months. Added a paragraph about SP9 NM-PM system in 2.8.2 Added sub-project directors to the author list.
0.9	11.04.2014	Draft	Review last addition, final edit by Guy Willis
1.0	17.04.2014	Draft	Use Case identifiers added by Martina Schmalholz

Table of Contents

1. Executive Summary 9
2. The Neurorobotics Platform 10
2.1 Overall Goals 10
2.1.1 Roadmap 10
2.1.2 Workflow 11
2.1.3 Users 12
2.1.4 Neurorobotics platform development team 12
2.2 Terminology 13
2.2.1 Definitions 13
2.2.2 Roles 14
2.3 Use Cases 14
2.3.1 Test a retina model on a brain model (offline simulation) (SP10NRP-UC-001). 14
2.3.2 Test a retina model on a brain model while interacting with the environment (SP10NRP-UC-002). 15
2.3.3 Test bio-inspired learning algorithms (offline simulation) (SP10NRP-UC-003). 15
2.3.4 Test a spinal cord model while interacting with the brain model (SP10NRP-UC-004). 16
2.4 Functional Requirements 17
2.5 Non-Functional Requirements 18
2.5.1 Usability and user experience 18
2.5.2 Open source 18
2.5.3 Community building 18
2.5.4 Interoperability 18
2.5.5 Best practices / quality 18
2.6 Architectural Overview 19
2.7 Physical Architecture 20
2.8 Relations to other Platforms 20
2.8.1 Unifying Portal (developed by SP6) 20
2.8.2 Brain Simulation (SP6) and neuromorphic hardware (SP9) 21
2.8.3 Brain models (SP3/SP4) 22
2.8.4 Brain Visualisation (SP6/SP7) 22
3. Robot Designer 24
3.1 Overall Goals 24
3.2 Use Cases 24
3.2.1 Main use case: Assemble a complete virtual robot (SP10NRP-UC-005) 25
3.2.2 Assemble a virtual robot from robot parts (SP10NRP-UC-006) 25
3.2.3 Load a virtual robot from robot library (SP10NRP-UC-007) 25
3.2.4 Import a virtual robot from file (SP10NRP-UC-008) 26
3.2.5 Customise robot's graphical properties (SP10NRP-UC-009) 26
3.2.6 Attach sensors and actuators (SP10NRP-UC-010) 26
3.2.7 Define a kinematic chain (SP10NRP-UC-011) 26
3.2.8 Customise robot's physical properties (SP10NRP-UC-012) 27
3.2.9 Add noise to sensors and actuators (SP10NRP-UC-013) 27
3.2.10 Save robot (SP10NRP-UC-014) 27
3.2.11 Change visualisation settings (SP10NRP-UC-015) 28
3.2.12 Load and customise a robot from script (SP10NRP-UC-016) 28
3.3 Functional Requirements 28
3.3.1 Robot assembly 29

3.3.2 Robot edition (expert) (SP10NRP-FR-004) 30
3.3.3 Storage 30
3.3.4 Change visualisation settings (user, expert) (SP10NRP-FR-007) 30
3.4 Non-Functional Requirements 31
3.4.1 Predefined models 31
3.5 Architectural Overview 31
3.6 Relations to other Platforms 32
4. Environment Designer 32
4.1 Overall Goals 32
4.2 Use Cases 33
4.2.1 Assemble and model virtual environment (SP10NRP-UC-017) 33
4.2.2 Define and save a building block (SP10NRP-UC-018) 33
4.2.3 Load/import/export (SP10NRP-UC-019) 33
4.2.4 Automatically generate a virtual environment from a script (SP10NRP-UC-020) 34
4.2.5 Edit virtual environment (SP10NRP-UC-021) 34
4.2.6 Define kinematic chain (SP10NRP-UC-022) 34
4.2.7 Save virtual environment (SP10NRP-UC-023) 35
4.2.8 Visualise the virtual environment (SP10NRP-UC-024) 35
4.3 Functional Requirements 35
4.3.1 Assemble and model a virtual environment (SP10NRP-FR-008) 35
4.3.2 Import/export/load/save (SP10NRP-FR-009) 35
4.3.3 Edit virtual environment (SP10NRP-FR-010) 36
4.3.4 Define a kinematic chain (SP10NRP-FR-011) 36
4.3.5 Visualise the virtual environment (SP10NRP-FR-012) 36
4.3.6 User interaction (SP10NRP-FR-013) 36
4.3.7 Automatically generate a virtual environment from a script (SP10NRP-FR-014) 36
4.4 Non-Functional Requirements 36
4.5 Architectural Overview 37
4.6 Relations to other Platforms 38
5. Brain Interfaces & Body Integrator 38
5.1 Overall Goals 38
5.2 Use Cases 41
5.2.1 Selection of the brain model (SP10NRP-UC-025) 41
5.2.2 Selection and grouping of neurons (SP10NRP-UC-026) 41
5.2.3 Selection and grouping of neurons – query (SP10NRP-UC-027) 42
5.2.4 Transfer modules creation (SP10NRP-UC-028) 42
5.2.5 Transfer modules connection – simple (SP10NRP-UC-029) 42
5.2.6 Transfer modules connection – chain (SP10NRP-UC-030) 43
5.2.7 Transfer modules connection - direct feedback (SP10NRP-UC-031) 43
5.2.8 Transfer modules connection - environment connection – sensors (SP10NRP-UC-032) 44
5.3 Functional Requirements 44
5.3.1 Brain model (SP10NRP-FR-015) 44
5.3.2 Transfer modules (SP10NRP-FR-016) 44
5.3.3 Connect group of neurons to actuators and sensors (SP10NRP-FR-017) 45
5.3.4 Project load/save (SP10NRP-FR-018) 45
5.3.5 Re-calibration of the brain model (SP10NRP-FR-019) 45
5.3.6 Snapshot region (SP10NRP-FR-020) 45
5.4 Non-Functional Requirements 46

5.4.1 Format 46
5.5 Architectural Overview 46
5.6 Relations to other Platforms 46
6. Experiment Designer 47
6.1 Overall Goals 47
6.2 Use Cases 48
6.2.1 Selection of a configuration element (SP10NRP-UC-33) 48
6.2.2 Defining a measurement point (SP10NRP-UC-34) 48
6.2.3 Defining an action sequence (SP10NRP-UC-35) 49
6.2.4 Configuring a protocol (SP10NRP-UC-36) 49
6.2.5 Defining an experiment set-up (SP10NRP-UC-37) 50
6.2.6 Saving an experiment set-up (SP10NRP-UC-38) 50
6.2.7 Loading an experiment set-up (SP10NRP-UC-39) 50
6.3 Functional Requirements 51
6.3.1 Configuring an experiment set-up (SP10NRP-FR-21) 51
6.3.2 Storage of experiment set-ups 52
6.4 Non-Functional Requirements 53
6.4.1 Exchangeability and interoperability 54
6.5 Architectural Overview 54
6.6 Relations to other Platforms 54
7. World Simulation Engine 55
7.1 Overall Goals 55
7.2 Use Cases 55
7.2.1 Configuration and control via API (SP10NRP-UC-40) 55
7.3 Functional Requirements 56
7.3.1 Configuration (SP10NRP-FR-30) 56
7.3.2 Load world state (SP10NRP-FR-31) 56
7.3.3 Save world state (SP10NRP-FR-32) 56
7.3.4 Simulation control (SP10NRP-FR-33) 56
7.3.5 World maintenance (SP10NRP-FR-34) 57
7.3.6 World simulation components (SP10NRP-FR-35) 57
7.4 Non-Functional Requirements 58
7.4.1 Physical significance 58
7.5 Architectural Overview 58
7.6 Relations to other Platforms 59
8. Closed Loop Engine 59
8.1 Overall Goals 59
8.2 Use Cases 61
8.2.1 Offline scenario with high fidelity visualisation (SP10NRP-UC-41) 61
8.2.2 Offline scenario without high fidelity visualisation (SP10NRP-UC-42) 62
8.2.3 Online scenario with a high fidelity visualisation (SP10NRP-UC-43) 62
8.2.4 Developer scenario (SP10NRP-UC-44) 62
8.3 Functional Requirements 63
8.3.1 Communication (SP10NRP-FR-36) 63
8.3.2 Online experiments (SP10NRP-FR-37) 63
8.3.3 Offline experiments (SP10NRP-FR-38) 63
8.4 Non-Functional Requirements 63
8.4.1 Future development 63

8.4.2 General architecture 63
8.5 Architectural Overview 64
8.6 Relations to other Platforms 64
9. Experiment Simulation Viewer 65
9.1 Overall Goals 65
9.2 Use Cases 66
9.2.1 Loading an experiment set-up (SP10NRP-UC-45) 66
9.2.2 Configuring and starting the simulation (SP10NRP-UC-46) 66
9.2.3 Time interaction (SP10NRP-UC-47) 67
9.2.4 Brain interaction 67
9.2.5 Robot interaction 68
9.2.6 Environment interaction 68
9.2.7 Record/replay 69
9.3 Functional Requirements 70
9.3.1 Control the experiment in simulation (SP10NRP-FR-39) 70
9.3.2 Brain (SP10NRP-FR-40) 70
9.3.3 Robot / environment (SP10NRP-FR-41) 70
9.4 Non-Functional Requirements 70
9.4.1 Hardware platform for simulation 70
9.5 Architecture Overview 71
9.6 Relations to other Platforms 72
10. Functions 72
10.1 Software engineering methodology 72
10.2 Scrum: Roles and Procedures 73
10.3 Project management and monitoring 74
10.3.1 Scrum review 74
10.3.2 Backlog 75
10.3.3 Development progress 77
10.3.4 Quality 78
10.3.5 User satisfaction 78
10.4 Stakeholders & science and technology officers protocol 79

Table of Figures

Figure 1 Overall user workflow in the neurorobotics platform 11
Figure 2 Example of a virtual skeleton robot on a virtual treadmill. 16
Figure 3 Example of a mouse robot in a virtual environment 17
Figure 4 Architectural overview of the Neurorobotics Platform 19
Figure 5 Neuron simulation frameworks to integrate in the NRP. 22
Figure 6 Brain Atlas Embedding Module (BAEM) prototype 23
Figure 7 Brain interactive simulation viewer 23
Figure 8 Robot Designer architectural overview 31
Figure 9 Environment Designer architectural overview 37
Figure 10 Brain Interfaces & Body Integrator mapping overview 39
Figure 11 User steps needed to create a Neurobot out of a brain model and robot model. 41
Figure 12 Overview of the data needed for an experiment. 53
Figure 13 Architecture overview of the Experiment Designer 54
Figure 14 Architecture overview of the World simulation engine 58
Figure 15 Physical architecture scenarios depending on the experiment 60
Figure 16 Resources location depending on the experiment (see previous figure) 61
Figure 17 Closed loop engine role as orchestrator 64
Figure 17 Architecture overview of the Experiment Simulation Viewer 71
Figure 18 Agile scrum iteration workflow: from the product owner definition of the backlog to a finished product. 74
Figure 19 Risk reduction using a short feedback loop with users. 75
Figure 20 Interactions between the request backlogs, the release backlogs and the iteration backlogs. 75
Figure 21 View of the backlog of the portal team. 77
Figure 22 Burndown chart example. 78

1. Executive Summary

The Human Brain Project (HBP) is a ten-year research project whose goal is to lay the foundations for a new model of brain research. The HBP is driving integration between data and knowledge from different disciplines, and catalysing a community effort to achieve a new understanding of the brain, new treatments for brain disease and new brain-like computing technologies.

Neurorobotics is a novel science that combines neuroscience and robotics. One important object of investigation is the embodiment of neural systems (e.g. brain-inspired algorithms and computational models of biological neural networks) as controllers for robotic devices.

The current understanding of neurorobotics is largely bound to the idea that the environment in which the robot interacts must be the real world. But the gap between simulation and reality is decreasing. There is no compelling argument that a rich, multimodal and noisy environment cannot be provided by a computer simulation, provided enough computational power is available.

If, however, a nearly ideal simulation environment is provided, studies of brain-based robots can be performed orders of magnitude faster than studies with real robots, which need to be designed, built, programmed, re-designed, etc. While current neurorobotics research is focused on real robots, the HBP's Neurorobotics Subproject will combine high performance computing and sophisticated simulation methods in one comprehensive toolbox for experiment simulation: the Neurorobotics Platform.

The Neurorobotics Platform will provide a software and hardware infrastructure that will allow non-robotics researchers, like neuroscientists, to perform simulated experiments (i.e. experiments *in silico*), which will facilitate insights into the links between low level brain circuitry and high-level cognitive and behavioural functions. It will also facilitate the development of a new type of robot controller derived from those insights.

This document describes the software requirements for the software to be built within the ramp-up phase and outlines work beyond that. The first section describes the Neurorobotics Platform in general. Sections 2 to 9 describe each component of the platform, with each section following the same organisation, so that the reader can switch from one component to the other easily. Section 10 describes the software development methodology proposed, as well as a set of KPIs for measuring progress, quality and user satisfaction.

2. The Neurorobotics Platform

This section presents an overview of the Neurorobotics Platform (NRP). It explains the overall goals of the Platform, the roadmap for achieving them and an outline of the workflow. Requirements are derived from the use cases. Relationships to other platforms are introduced, as well as an overview of the Neurorobotics Platform's architecture and components.

2.1 Overall Goals

The Neurorobotics Platform's objective is to allow non-robotics researchers like cognitive neuroscientists to perform experiments *in silico*. In such experiments, a brain model is coupled to a simulated robot (body), interacting within a simulated world. The level of abstraction of the brain models ranges from micro via meso to macro scale connectomes: it could be a model of a particular neuronal circuit, a region like a Brodmann area or even the whole brain.

Using a simulation approach with a variable degree of model abstraction will allow us to replicate classical experimental paradigms, and eventually develop new ones. Our goal is to gain new insights into the causal relationships linking basic neural constituents to perception, cognition and behaviour.

Simulating an experiment also implies simulating a robot's brain. After running a brain simulation on a dedicated computer node, it is only a small step to transfer this software from a simulated robot to a real robot.

The Neurorobotics Platform aims to develop and establish a sustainable and open source software solution. Software modules will be derived from established tools with a strong developer community, and from software already developed in the Blue Brain Project. Developers from the robotics community and the open source community are invited and encouraged to take part in this continuous effort.

2.1.1 Roadmap

Ramp-up phase: At the end of the ramp-up phase, a functional prototype of the Neurorobotics Platform will exist. This prototype will be ready for community release (MS196, Month 30). It will conduct a simulated experiment containing a proof of the Weber-Fechner law. The Weber-Fechner law states that the subjective sensation of stimuli is proportional to the logarithm of the objective physical stimulus intensity. The experiment set-up for the proof will focus on visual perception and will rely on the capabilities of the Brain Simulation and High Performance Computing platforms.

Lifetime of the HBP: In the second phase of the project, the scope of these investigations will be broadened to match experimental investigations in the brain function and cognitive architectures Subprojects. The spectrum of considered perceptions may then include auditory and somatosensory processing, including multisensory perception, object recognition (recognition of faces, body parts, houses, words etc.), action recognition and novelty detection (e.g. auditory novelty detection through mismatch negativity).

Future Prospects: While at the beginning, the focus will be on visual perception, the long-term vision is to include modalities that are more complex to simulate, such as odour. Besides broadening the range of stimuli, higher cognitive functions as planning, memory,

decision-making, error-correction or even speech (language processing and production) will be focused on.

Natural changes in the brain will then also be researched by simulating developmental processes like growth or ageing. Furthermore, the platform will be used to validate hypotheses about brain illnesses and disorders. It will then be possible to simulate neurological or neuropsychological disorders (e.g. agnosia or epilepsy) and study the causal relationships between alterations or lesions and behaviour, for example a malfunction in motion after injuries in the motor cortical circuits.

2.1.2 Workflow

The typical user workflow needed to conduct this experiment (and other types of experiments) is shown in Figure 1. It is divided in two main phases: the design and the execution of the experiment. The design phase requires a set of design tools:

- The **Robot Designer** enables the user to choose or design a robot.
- The **Environment Designer** enables the user to choose or design an environment in which the robot will interact.
- The **Brain Interfaces & Body Integrator** enables the user to choose a brain model from the Brain Simulation Platform (SP6). It then lets them draw and design the connections between the chosen brain model and the chosen robot.
- The **Experiment Designer** enables the user to program what should happen during the simulation phase.

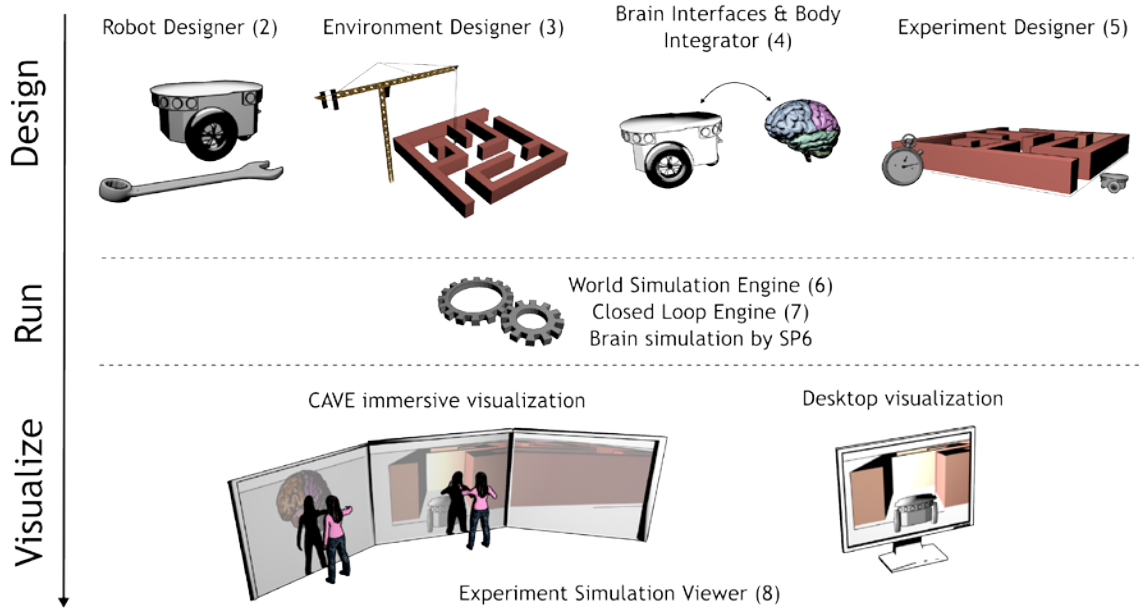

Figure 1 Overall user workflow in the neurorobotics platform

Once the experiment is properly designed, it can be executed. This phase happens with help of three components:

- The **World Simulation Engine** (described in Section 7) simulates the environment and the robot. It is the robotic equivalent of the brain simulator on the neuroscience side.

- The **Brain Simulation** simulates the activity within the brain. The brain simulation will be provided by SP6 (more details about types of simulation can be found in 2.8.2)
- The **Closed Loop Engine** (described in Section 8) is the component that allows and organises the communication between the two simulators (brain and world).

The interaction of the user with an experiment can happen in two possible scenarios:

- The **online scenario** where the simulation of the different components is done in real time. Users can furthermore actively interact with the robot, the environment and the brain during the simulation.
- The **offline scenario**. The simulation is treated as a regular HPC computing job. Users analyse the outcome after its completion.

During the ramp-up phase, the development of the platform will focus first on the offline scenario. The online scenario is technically more complicated.

However, regardless of the user scenario, one specific component will let the user both visualise and analyse his or her experiment. This is the Experiment Simulation Viewer, described in Section 9. One important particularity of this component is that it has to support an immersive environment like a CAVE, as well as a desktop computer screen.

2.1.3 Users

The Neurorobotics Platform will bridge the gap between robotics and neuroscience research. It will hence address a very large user base, ranging from researchers in mechanical robotics to brain modelling neuroscientists. The platform will address needs in both domains.

The philosophy of the Neurorobotics Platform concerning the level of expertise of the users is to let them focus on what they understand and make available generic solutions to cover aspects they are less familiar with. A scientific user should be able to program or configure detailed parts of an experiment related to their field of knowledge. They should be able to implement whatever is needed for them to make progress in their research area. For the other parts of their experiment set-ups, they should be able to benefit from other people's work in order to ease the configuration process.

A neuroscientist expert user in spinal cord modelling (see 2.3.4) doesn't necessarily know how to design a complex mouse-like robot. The NRP will let them choose one from an existing library (possibly developed by other users of the NRP). With that, they will be able to perform experiments centred on their spinal cord models connected to the mouse-like robot.

2.1.4 Neurorobotics platform development team

An external partner will execute the NRP software development, starting on 1 April 2014. This partner will be the winner of an open call initiative that took place between September 2013 and February 2014. The NRP team within the HBP will collaborate closely with them during throughout the ramp-up phase. The external partner will be integrated in the Description of Work (DoW) by the addition of a new work package. The NRP team within HBP is part of WP10.1.

We will start the collaboration by transforming this user-oriented specification in user stories (user stories can be seen as the task & functions used by other SP). Then, the development team will evaluate the user stories and we will start the implementation. The

software development methodology that will be used with the selected partner is Scrum (described in more details in 10.1).

2.2 Terminology

This section is meant to define terminology used in this document that might otherwise be ambiguous or have semantic divergence with common uses of the same terminology.

2.2.1 Definitions

BAEM - Brain Atlas Embedding Module.

BIBI - Brain Interface & Body Integrator, a component of the NRP that allows the sensors/actuators of a robot to interconnect with a brain model.

CAVE - A CAVE (Cave Automatic Virtual Environment) is an immersive virtual reality environment. Typically, four to six walls of a CAVE room have projectors or screens.

CLE – Closed Loop Engine one of the building blocks of the neurorobotics platform that orchestrates the brain simulation and the world simulation

COLLADA - From collaborative design activity is an interchange file format (XML) for interactive 3D applications.

Connectome - A comprehensive map of neural connections in the brain.

GPL – GNU General Public License, a common open source license, also see https://www.gnu.org/licenses/gpl-3.0.en.html

HPC - High Performance Computing.

Kinematic chain - An assembly of rigid bodies connected by joints.

NEST - A simulator for spiking neural network models that focuses on the dynamics, size and structure of neural systems rather than on exact morphology of individual neurons.

Netsim – Netsim will be an improved version of NEST that scales up more efficiently on super computer. The development of NetSim is part of SP6 – Task 6.2.3.

Neurobot - A robot that is already equipped with or interconnected to a brain model.

NEURON - A simulation environment for modelling individual neurons and networks of neurons.

NRP – Neurorobotics platform.

ROS - Robot Operating System.

SpiNNaker – SpiNNaker ("Spiking Neural Network Architecture") is a massively parallel, low power, neuromorphic supercomputer currently being built at Manchester University in the UK.

Unifying Portal (UP) - The unifying web interface through which the web accessible components of the 6 HBP Platforms are delivered. The UP also provides transversal services to each platform including user management, document management and security.

URDF – "Unified Robot Description Format" is an XML format for representing a robot model.

WSE – "World Simulation Engine", one of the building blocks of the neurorobotics platform that takes care of the robots and environment simulation.

XML - Extensible Markup Language, a markup language that defines a set of rules for encoding documents in a human-readable and machine-readable format.

2.2.2 Roles

Neuroscientific User (NSU) - A user with expertise in neuroscience, but limited technical computing skills, and no expertise in robotics or 3d modelling

Experienced Neuroscientific User (ENSU) - A user with expertise in neuroscience, and good experience in the Neurorobotics Platform who wants to go into more detailed scenarios

Scientific Developer (SCIDEV) - A user who is developing software to directly achieve the scientific objectives. This user is usually working in close collaboration with scientists as NSUs and ENSUs.

2.3 Use Cases

The following use cases describe in a rather coarse grain what a user will be able to achieve using the Neurorobotics Platform. They are inspired from the contacts that the neurorobotics team has with potential early adopters of the platform. Each step is described more precisely in the use cases section related to their component.

2.3.1 Test a retina model on a brain model (offline simulation) (SP10NRP-UC-001).

Primary Actors: One Neuroscientific User: Abigail, and one Scientific Developer: Bill.

Success Scenario:

1) Abigail starts the design of the experiment with the Robot Designer. She chooses a pre-defined robot model with an eye sensor.
2) Abigail navigates to the Brain Interface & Body Integrator component. She loads a retina model that has been previously programmed by Bill using a known language (such as Python).
3) In the same component, Abigail selects a brain model that has been programmed through the Brain Simulation Platform (SP6).
4) Abigail selects groups of neurons from the brain model (through the Brain Atlas Embedding Module) and connects them to the retina model.
5) Abigail connects the robotic eye sensor to the retina model.
6) Abigail navigates to the Environment Designer and builds an environment composed of an illuminated screen in a dark room.
7) Abigail navigates to the Experiment Designer and chooses to run the experiment for 5 minutes. She also programs the illuminated screen to have its light intensity exponentially increasing over that period.
8) Abigail saves the setup she just did.
9) Abigail launches the experiment. She then starts doing something else.
10) Abigail receives a notification that the experiment simulation is finished.

11) Abigail goes in the CAVE of Munich (good thing she lives in Munich). She loads her experiment simulation and replays it.

12) Abigail pauses the replay and, in the brain visualisation component, navigates to a particular region of brain she is interested in.

13) Abigail continues the replay and observes the brain region she has selected when different patterns appear on the virtual illumined screen.

2.3.2 Test a retina model on a brain model while interacting with the environment (SP10NRP-UC-002).

Primary Actors: One Neuroscientific User: Abigail and one Scientific Developer: Bill.

Success Scenario:

1) Abigail loads the experiment setup she did in the previous use case.

2) Abigail navigates to the Experiment Designer and chooses to run the experiment continuously.

3) Abigail goes in the CAVE of Munich. She loads her experiment setup (in the Experiment Simulation Viewer) and plays the experiment.

4) Abigail, in the brain visualisation component, navigates to a particular region of the brain she is interested in.

5) Abigail observes the brain region she has selected while interacting with the virtual illumined screen.

2.3.3 Test bio-inspired learning algorithms (offline simulation) (SP10NRP-UC-003).

Primary Actors: One Neuroscientific User: Abigail and one Scientific Developer: Bill.

Success scenario:

1) Abigail starts the design of the experiment with the Robot Designer. She builds a human skeleton robot with muscles attached to its legs, using robot parts previously set up by Bill.

2) Abigail navigates to the Environment Designer. She chooses an empty environment.

3) Abigail places a treadmill on the environment. Then, she puts the skeleton robot on the treadmill.

4) Abigail places a virtual recording device on the treadmill. This device records the distance made by the skeleton robot.

5) Abigail navigates to the Brain Interface & Body Integrator component. She chooses a predefined model of the spinal cord.

6) In the same component, Abigail selects a brain model that has been programmed through the Brain Simulation Platform (SP6).

7) Abigail selects groups of neurons from the brain model (through the Brain Atlas Embedding Module) and connects them to the spinal cord.

8) Abigail connects then the spinal cord model to the muscle of her skeleton robot.

9) Abigail loads a learning algorithm that has been programmed by Bill using a known language (such as Python). The platform has provided him with interfaces to the brain model in order to achieve that.

10) Abigail connects her algorithm to the recording device of the environment. This way, the learning algorithm can have a "reward" input: the travelled distance of the robot.

11) Abigail chooses to have a snapshot of the brain taken at the end of the experiment.

12) Abigail navigates to the Experiment Designer. She chooses to have 100 runs of her experiment.

13) Abigail launches the experiment. She then starts doing something else.

14) Abigail receives a notification that the experiment simulation is finished.

15) Abigail navigates to the Experiment Simulation Viewer and loads her experiment simulation.

16) Abigail replays the last run of the experiment. She can visually appreciate how much her algorithm was successful by looking at the robot walking. She can compare that to other algorithms she may have programmed in the past.

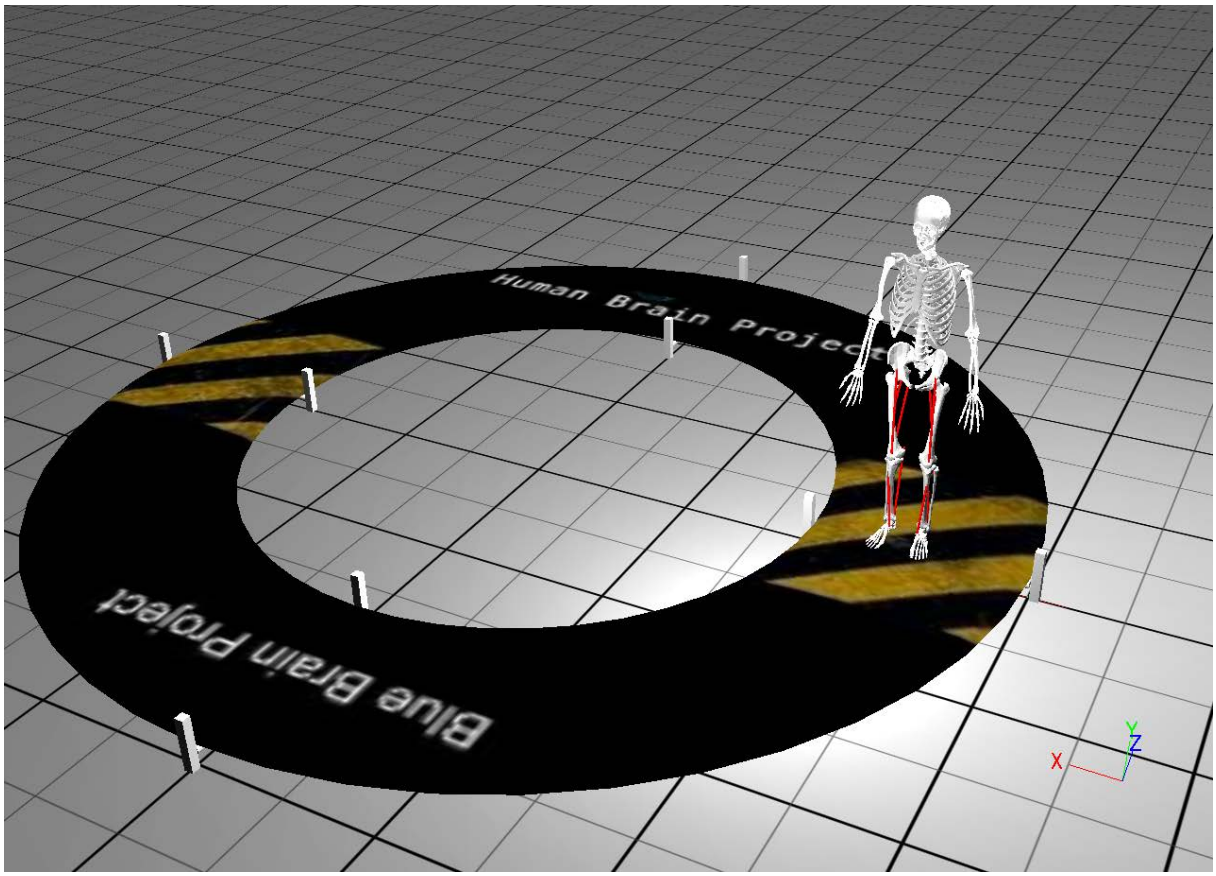

Figure 2 Example of a virtual skeleton robot on a virtual treadmill.

2.3.4 Test a spinal cord model while interacting with the brain model (SP10NRP-UC-004).

Primary Actor: One Neuroscientific User: Abigail

Success Scenario:

1) Abigail starts the design of the experiment with the Robot Designer. She chooses to use an existing model of a mouse.
2) Abigail navigates to the Environment Designer. She chooses to use an empty room environment.
3) Abigail navigates to the Brain Interface & Body Integrator component. She chooses a predefined model of the spinal cord.
4) In the same component, Abigail selects a brain model that has been programmed through the Brain Simulation Platform.
5) Abigail selects an existing set of connections between the robot, the spinal cord and the brain model.
6) Abigail navigates to the Experiment Designer and chooses to run the experiment continuously.
7) Abigail goes in the CAVE of Munich. She loads her experiment setup (in the Experiment Simulation Viewer) and plays the experiment.
8) Abigail visually navigates to a particular region of the brain she is interested in.
9) Abigail stops the experiment and takes a snapshot of the world and brain for later use.

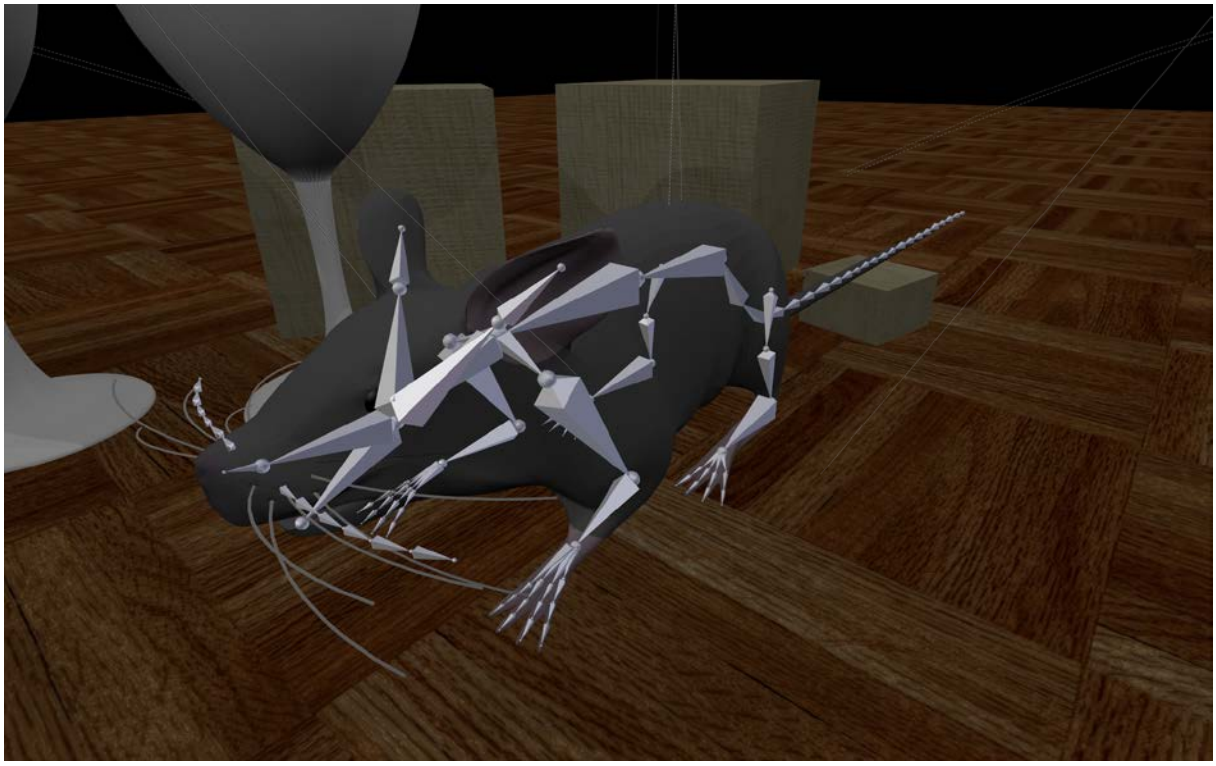

Figure 3 Example of a mouse robot in a virtual environment

2.4 Functional Requirements

The functional requirements of the overall system are specified in detail in the next sections (3 to 9).

2.5 Non-Functional Requirements

The following non-functional requirements apply to every component of the Neurorobotics Platform.

2.5.1 Usability and user experience

- The platform is easily accessible for non-field experts: a neuroscientist is not assumed to have knowledge about robots and vice-versa for a robotics expert:
  - The user interaction is designed towards a non-expert user.
  - It uses intuitive tools and favours user-centric design.
  - It also provides guidance for expert users when they need to implement detailed and complicated parts of an experiment (robot, brain connections, environment or brain model).
- The user experience has to be consistent among all parts of the platform.

2.5.2 Open source

In order to fulfil its ambition, the NRP has to rely on existing building blocks. On one side of the platform, the building blocks are pretty well defined: the brain simulators provided by SP6 (starting with NetSim, an already available open-source point-neuron simulator). On the other side, we have to make a careful choice when looking for robotic simulation software and related components. The first main requirement is to select open source building blocks. Indeed, since the NRP will be released to a wide audience at the end of the ramp-up phase, using closed source software is to be avoided. The use of components with viral open source licenses (as GPL) is allowed only if strictly necessary.

2.5.3 Community building

The NRP development team will collaborate with WP 10.3 ("Neurorobotics Platform: user support and community building") leaders in order to help them reach their goals:

- The usability of the platform must be very high.
- Sufficient documentation (including tutorials, help pages, websites) for the users as well as for developers must be provided.
- The platform must be extensible by community contributions.
- The NRP development team participates in the open source communities of the chosen building blocks. They also try to attract developers from these communities.

2.5.4 Interoperability

- Each software part that may save data should allow, wherever possible, the export of data in other well-known formats.
- The platform should be compatible, whenever possible, with open source models libraries (especially in the Environment Designer).

2.5.5 Best practices / quality

The development of every component of the platform will have to follow software engineering best practices:

- Good infrastructure (task tracking system, bug tracking system, version control).
- Whenever possible, these best practices and the associated tools should be the same as the one used by other SP, in particular SP6 and SP7.
- Methods like continuous build, unit testing, code review, static analysis, code conventions will enforce high quality.

2.6 Architectural Overview

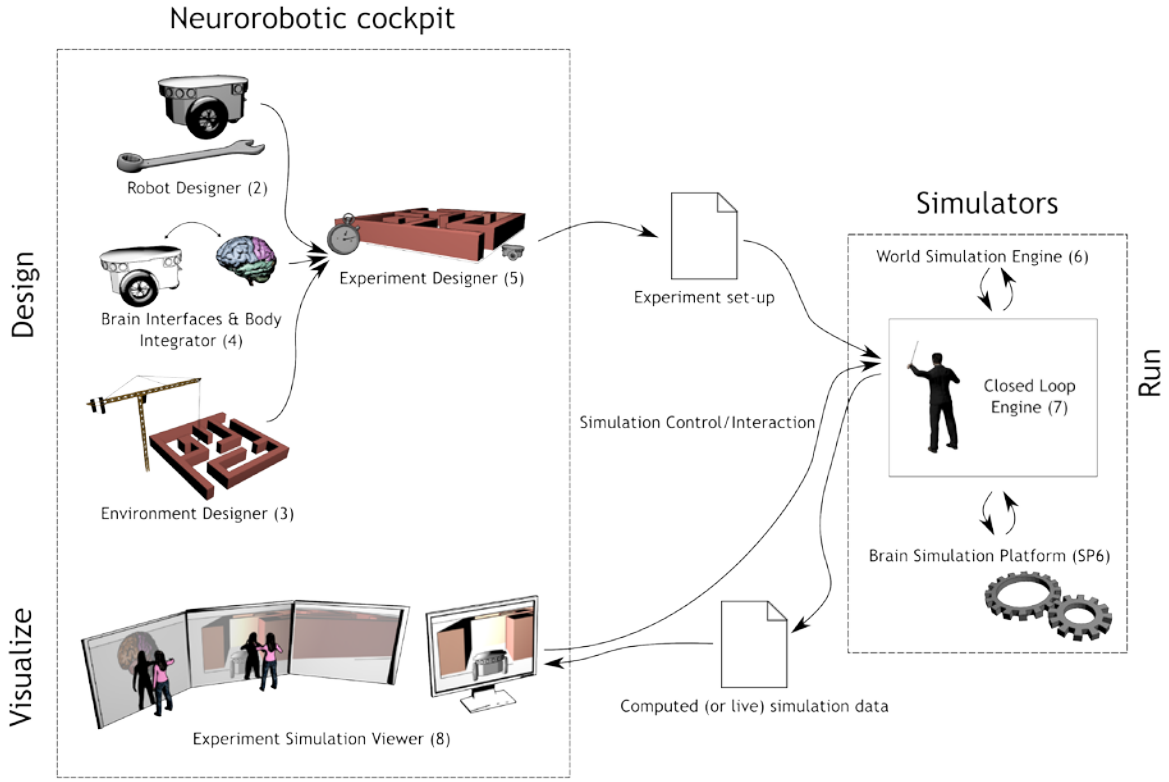

Figure 4 Architectural overview of the Neurorobotics Platform

The overall architecture can be broken down into two logical blocks:

The Neurorobotics Platform's Cockpit

- The NRP Cockpit consists of five sub-components, four of which are concerned with the first phase, namely experiment designing
- Within the Robot Designer, a user can select one of a number of robots to use in an experiment. Together with the Brain Interfaces & Body Integrator, a Neurobot can be created.
- As a result, a (dynamic) virtual environment is created within the Environment Designer.
- The output of both is the input to the Experiment Designer. Besides configuring an experiment set-up in terms of the Neurobot and environment to use, a sequence of

actions is defined, as well as measurements. This information constitutes the output of the Experiment Designer, namely an experiment set-up.

- One further component, the Experiment Simulation Viewer, is used during the execution phase. It is used as a human interface to the simulation components described below.

The Simulation Components

- **Closed Loop Engine**. This serves as a mediator between the separated simulations of the Neurobot, in the environment and the brain respectively. The output is the computed simulation; the visualisation on the other side will take place in the Experiment Simulation Viewer.
- **World Simulation Engine**. This modular structured component will simulate the dynamics of the Neurobot as a physical object in its environment.
- **Brain Simulation**. The brain simulation components are not a direct part of the NRP, but will be interfaced through the Closed Loop Engine.

Each major component of the NRP is described in more detail in Sections 3 to 9.

2.7 Physical Architecture

The physical architecture depends on the experiment. More precisely, it depends on the required visualisation complexity (CAVE setup or desktop computer) and the required level of interactivity (online or offline experiment). The resulting different physical architecture setups are described in Section 8 - Closed Loop Engine.

2.8 Relations to other Platforms

The NRP will consume services offered by other platforms, but will not provide any services to other platforms itself. The needed services from the other platforms can be sorted into several categories:

2.8.1 Unifying Portal (developed by SP6)

All the components of the NRP use several services provided by the Unifying Portal (see specification of the Brain Simulation Platform, D67.1, for more details):

- Authentication service, access rights and user profile. The users should not have a different login/password than for the other platforms. The Unifying Portal will provide this service using an open standard.
- Document repository. The users should be able to use the same document repository as the other platforms when they have to save and load their projects. This repository is exposed through the "Document Service REST API". It supports the NRP's sharing requirement: A user should be able to share its models (brain, connections, environment, robots or experiments) with other people.

2.8.2 Brain Simulation (SP6) and neuromorphic hardware (SP9)

The first simulator to be integrated within the NRP will be NetSim (Network simulation - Task 6.2.3 in the Description of Work). NetSim is the continuation of the development of NEST, an existing point-neuron simulator. NetSim is a simulator that requires less computing power than detailed simulators (such as NEURON). It can be used directly in its current version (Task 6.2.3 is mainly about improving its performance). The NRP development team will collaborate closely with the NetSim development team in Jülich in order to implement some features required only by the NRP.

During the integration of NetSim, we will choose communication interfaces carefully, to minimise the amount of work required to integrate other simulators.

Depending on the effort needed and the resources available, the next simulator that the NRP will integrate will be the SpiNNaker. The SpiNNaker is a many-core processor platform (NM-MC) dedicated to neural simulation, being developed by WP9.2. The hardware is already available, but the software requires some further development to allow integration in the NRP. The SpiNNaker will be more powerful than NetSim and will help the NRP platform to reach near-real time simulation.

SP9 is developing another hardware-based simulator: the NM-PM system (in WP9.1). The difference with the SpiNNaker is that this system is built upon a physical emulation of brain models. The NM-PM system is targeted to have a simulation time 1,000 to 10,000 times faster than real-time. This highly accelerated feature will allow faster than real-time offline experiments. This will be useful, for example, when repeating an experiment many times with different parameters. Depending on the availability date of the NM-MP platform, integration with the NRP should be possible toward the end of the ramp-up phase.

The final goal of the NRP will be to integrate detailed simulations provided by WP6.2 after the ramp-up phase is complete. Integration of the two other simulation frameworks will facilitate achievement of this goal, as the same communication interfaces will be used.

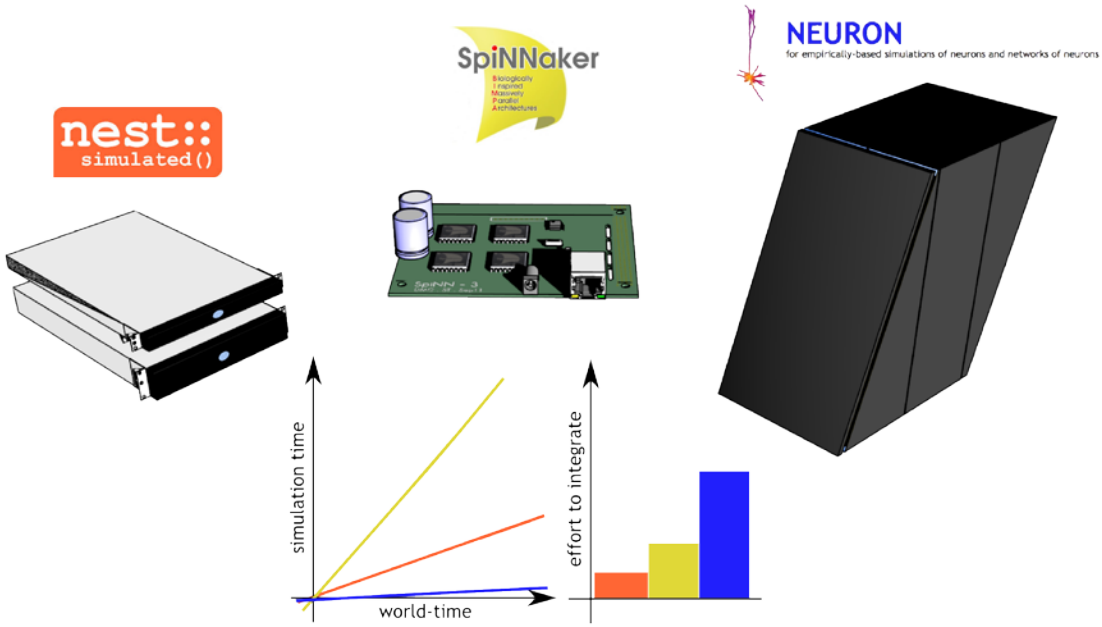

Figure 5 Neuron simulation frameworks to integrate in the NRP.

2.8.3 Brain models (SP3/SP4)

By using SP6's Brain Simulation Platform or SP9's Neuromorphic Hardware, the NRP will automatically let users benefit from the brain models developed by SP3 (Cognitive Architectures) and SP4 (Mathematical and Theoretical Foundations of Brain Research).

2.8.4 Brain Visualisation (SP6/SP7)

The NRP will make use of the atlas of the brain provided by SP6 (Task 6.1.2). The use cases are described in Section 5 - Brain Interfaces & Body Integrator.

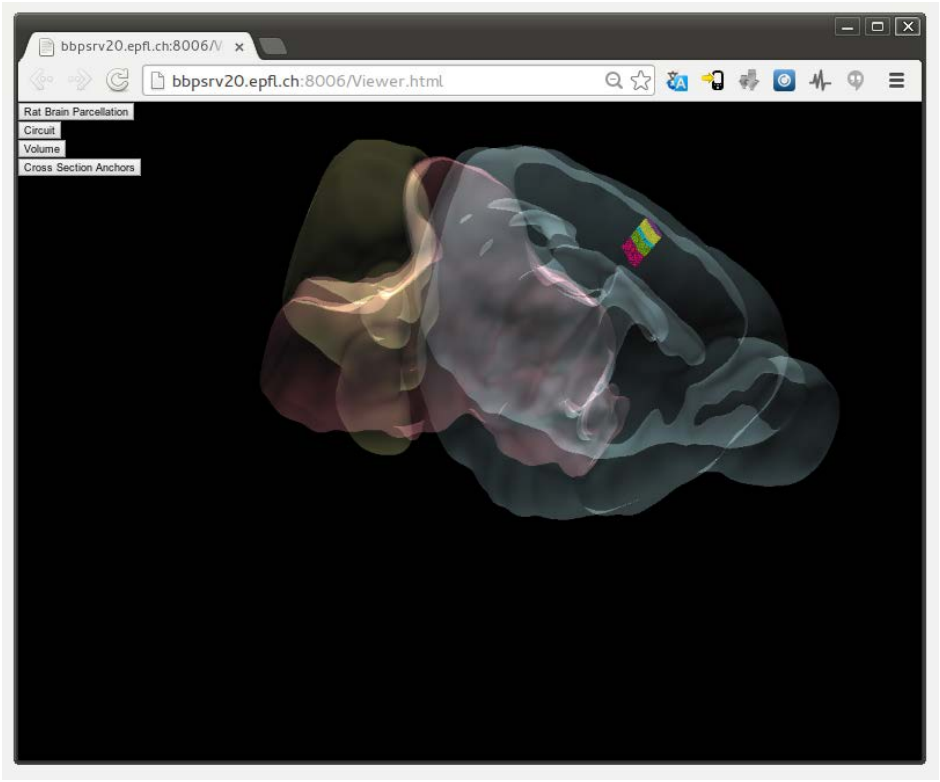

Figure 6 Brain Atlas Embedding Module (BAEM) prototype

The platform will also make use of the brain interactive visualisation provided by WP7.3 - Interactive visualisation, analysis and control. The use cases are described in Section 8 - Closed Loop Engine.

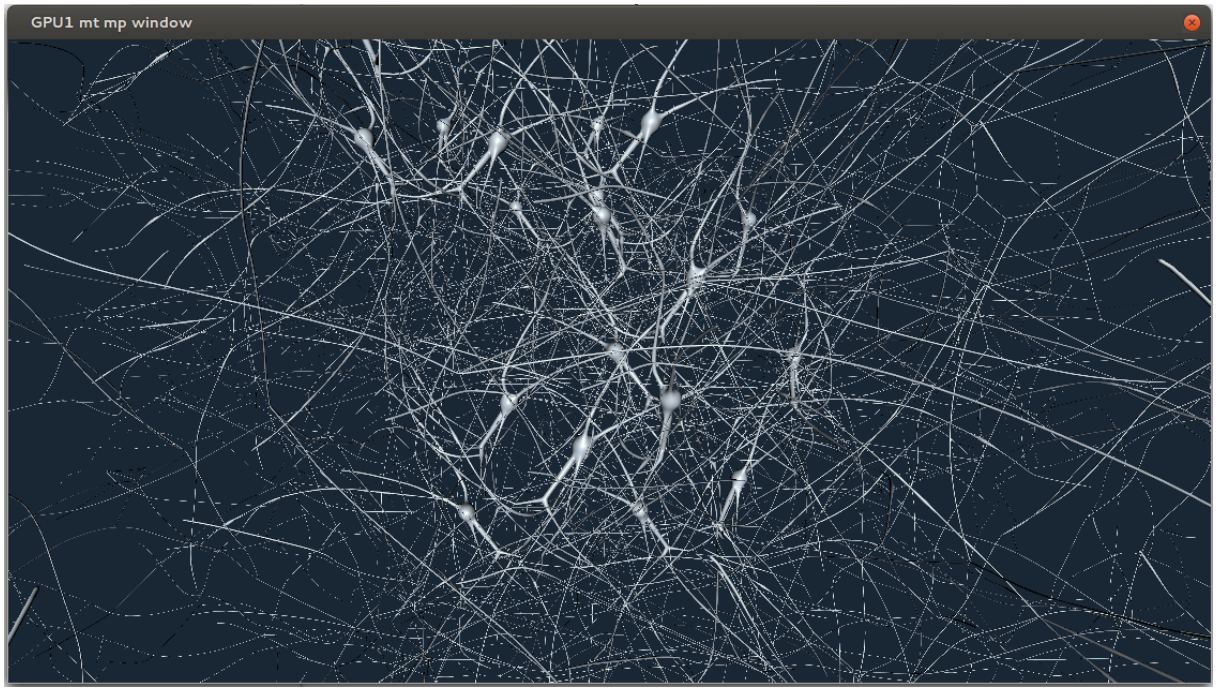

Figure 7 Brain interactive simulation viewer

3. Robot Designer

3.1 Overall Goals

The Robot Designer will allow a user to easily set up a virtual robot for a given simulated experiment. The word "easily" is here very important since the user may be a neuroscientist, and unfamiliar with robotics details. The robot shall be integrated into an environment later on. This environment can be minimalistic in the case of proprioceptive experiments, but will in any case be linked to a Brain using the Brain-Body Integrator. User acceptance should be good, leading to a very quick, intuitive learning process.

The purpose of designing a virtual robot is to test the brain's sensing and acting capabilities in a virtual environment, or in response to imposed stimuli. A virtual robot allows researchers to observe the advantages of a brain controller, as opposed to a conventional algorithmic controller. Therefore, the tool shall be focused to deliver high usability.

The Robot Designer provides the user with user-friendly tools to specify the robot's graphical and physical properties, hiding the technical fine-tuning under a simple GUI. In the final state, to be completed after the ramp-up phase of the project, users will be equipped with libraries of very simple "robot parts" (arms, legs, sensors), or "design primitives" (links, bones, joints and muscles) and tools to interconnect them intuitively. This approach will allow users to quickly assemble a whole virtual robot and will support musculoskeletal robots, as well as conventional ones.

In the first version of the Robot Designer, to be completed at the end of the ramp-up phase, the Designer will provide the user with pre-assembled robots from a "robot library", or from imported files for quick customisation. Therefore, the robot model should be exportable, reusable and standardised for interoperability by using common file formats as like URDF or COLLADA.

Likewise, sensors and actuators should be presented in graphical easy-to-use libraries. Addition of noise in sensors or actuators should be possible and straightforward.

3.2 Use Cases

Let us go through a typical use case for the Robot Designer. Let us assume that the Designer was successfully installed. We have a Neuroscientific User (NSU), Abigail, who wants to perform high-level or simple tasks to quickly set up a robot. Another Neuroscientist (ENSU), Charlie, has good experience in the platform and wants to go into more details of the robot's properties. A Scientific Developer (SCIDEV), Bill, wants to access lower-level and scriptable fine-grained functionalities. The three users have the tools to design a complete robot, because the software has three user interface layers, called "user mode", "expert mode" and "guru mode" (see functional requirements). The "user mode", which Abigail wants to use, enables a user to build a complete robot, without prior knowledge and without going into robotics details. The "expert mode" enables a user to fine-tune the robot by editing its kinematic or physical properties. The "guru mode" provides scripting capacities to the interface.

The user first starts the Robot Designer as part of the NRP toolchain. Use cases are defined below (some use cases may be combinations of other use cases).

Primary actors:

- Abigail: NSU (Neuroscientific user, no experience in the Robot Designer).
- Charlie: ENSU (Neuroscientific user, good experience in the Robot Designer).
- Bill: SCIDEV (Scientific developer).

3.2.1 Main use case: Assemble a complete virtual robot (SP10NRP-UC-005)

Goal: After starting the Robot Designer, set up a complete virtual robot.
Preconditions:

- Abigail or Charlie has started the Robot Designer (this will be assumed in all subsequent use cases).

Success scenario:

1) 3.2.2: Abigail assembles a robot from robot parts OR,
2) 3.2.3: Abigail loads a robot from a library OR,
3) 3.2.4: Abigail imports a robot from a file.
4) Optional: 3.2.5: Abigail customises graphical properties.
5) Optional: 3.2.6: Abigail removes/adds sensors and tunes actuators.
6) Optional: 3.2.7: Charlie adjusts the kinematic constraints.
7) Optional: 3.2.8: Charlie adjusts the physical properties.

3.2.2 Assemble a virtual robot from robot parts (SP10NRP-UC-006)

Not yet scheduled for the ramp-up phase.
Goal: Build a highly customised robot using ready-made parts.
Success scenario:

1) Abigail opens the "robot parts" library and picks ready-made robot parts from it.
2) Abigail lays them on the Designer building space.
3) Abigail easily connects the robot parts together.
4) Optional: 3.2.7: Charlie defines a kinematic chain.

3.2.3 Load a virtual robot from robot library (SP10NRP-UC-007)

Goal: As an alternative to 3.2.2, load a preassembled typical robot from a robot library and customise it slightly. In this user-friendly mode, detailed sub-use cases like "3.2.7 Define a kinematic chain" are possible but the idea is to skip those technical processes and have a robot set up within minutes.
Success scenario:

1) Abigail opens the robot library.
2) Abigail chooses a preassembled robot (humanoid robot, mouse, automated vehicle...)
3) Abigail loads it into the design space.

3.2.4 Import a virtual robot from file (SP10NRP-UC-008)

Goal: As an alternative to 3.2.2, import a preassembled robot from a file, either previously created in this Robot Designer, or in any compatible one.
Success scenario:

1) Abigail opens the import dialog.
2) Abigail chooses local or remote file.
3) Robot loads into Designer space.

3.2.5 Customise robot's graphical properties (SP10NRP-UC-009)

Goal: Adapt robot's appearance (e.g. colour, reflectance, material).
Preconditions:

- Abigail has set-up a robot.

Success scenario:

1) Abigail selects robot part(s).
2) Abigail chooses from a list of graphical properties.
3) Abigail changes the selected property or cancels all changes.

3.2.6 Attach sensors and actuators (SP10NRP-UC-010)

Goal: Provide the robot with moving and sensing capabilities. Sensors can be optical (cameras, light or other electromagnetic wave sensors, IR sensors), acoustic (microphones), haptic (vibration sensors, pressure sensors), thermic etc. Actuators can be motoric (rotative motors, longitudinal cylinders, muscles/tendons) or emittive (light emitter, ultra-sound loudspeaker, speech synthesiser, vibrator).
Preconditions:

- Abigail or Charlie has set-up a robot.

Success scenario:

1) Abigail selects a link where she wants to add an actuator.
2) Abigail pops up an actuator library.
3) Abigail chooses an appropriate actuator.
4) Abigail tunes the actuator properties as wished (speed, resolution...)
5) Optional: Charlie opens a Noise addition dialog.
6) Optional: 3.2.9: Charlie adds noise to actuator
7) Abigail selects a place on the robot surface.
8) Abigail attaches a sensor following items 2 to 6, where the word "sensor" replaces "actuator".

3.2.7 Define a kinematic chain (SP10NRP-UC-011)

Goal: Define constraints on robot's moving capabilities, for example angle limits (like for a human forearm) or course ends. Define degrees of liberty and passive reactions of robot

parts on specific actuator action, for example the lift of an arm will move the forearm, wrist and hand accordingly.
Preconditions:

- Charlie has set-up a robot.

Success scenario:

1) Charlie selects a link on the robot.
2) Charlie opens a kinematics dialog.
3) Charlie defines kinematics (degrees of liberty, constraints w/ regards to linked parts).
4) Charlie closes dialog.

3.2.8 Customise robot's physical properties (SP10NRP-UC-012)

Goal: Adapt robot's physics to Abigail's wishes (materials, mass, friction etc.)
Preconditions:

- Charlie has set up a robot.

Success scenario:

1) Charlie selects robot part(s).
2) Charlie chooses from a list of physical properties.
3) Charlie changes the selected property or cancels all changes.

3.2.9 Add noise to sensors and actuators (SP10NRP-UC-013)

Goal: Add a noise transfer function to actuators (hysteresis, non-linearity, signal inaccuracy…) and to sensors (white acoustic noise, light flickering, random failures…)
Preconditions:

- Charlie has set-up a robot.
- Charlie has attached sensors and actuators to the robot.

Success scenario:

1) Charlie selects a sensor or an actuator.
2) Charlie pops up a noise dialog.
3) Charlie chooses from a list of standard noise types.
4) Charlie customises noise properties (amplitude, frequency).
5) Charlie closes dialog.

3.2.10 Save robot (SP10NRP-UC-014)

Goal: Save an assembled or partly assembled robot into a file for future reuse. Export to compatible robot editors.
Preconditions:

- Abigail has set up a complete robot.

Success scenario:

1) Abigail opens save or export dialog.

2) Abigail chooses a local or remote location.
3) Abigail validates.
4) The robot is saved in a NRP project file OR,
5) The robot is exported to standard file formats (NRP-specific features may be discarded, depending on target platform capabilities).

3.2.11 Change visualisation settings (SP10NRP-UC-015)

Goal: navigate into the designer space using 3D controllers or the mouse.
Preconditions:

- Abigail has set up a complete robot.

Success scenario:

1) Abigail selects the rotation controller and rotates the whole view.
2) Abigail selects the zoom controller and zooms into the robot.
3) Abigail opens a view dialog and selects what should be displayed (axes, bounding boxes, etc.)

3.2.12 Load and customise a robot from script (SP10NRP-UC-016)

Goal: Access expert level functions from script
Success scenario:

1) Bill loads a script from within the Designer.
2) The script imports a ready-made robot.
3) The script changes some graphical, physical and kinematic properties.
4) Bill loads another script.
5) The script adds sensors and actuators to the robot.
6) The script adds noise to some sensors.
7) Bill adds another sensor interactively using the script editor.

3.3 Functional Requirements

The purpose of the Robot Designer is to assemble a complete robot with sensors, actuators and realistic kinematics. This process will be kept simple since it should be accessible to neuroscientists or other interested people outside of the robotics domain. More detailed tuning should remain possible, though hidden from the easy assembly processes. Every level of use ("user mode" (Abigail), "expert mode" (Charlie), "guru mode" (Bill)) is accessible from the interface, but the user can choose to filter out the functions that are not relevant for them. For example, if Abigail chooses the "user mode", she will be presented all the straightforward tools, like loading a ready-made robot from a library, but not the more technical tools, like fine-tuning the kinematic chain. If she wants to fine-tune the kinematic chain, she will switch to "expert mode".

3.3.1 Robot assembly

3.3.1.1 Assemble a virtual robot (user, expert, guru) (SP10NRP-FR-001)

User must be able to:

- Load a ready-made robot from a graphical library ("user mode").
- Assemble a robot using ready-made robot parts from a graphical library[1] ("user mode").
  - User must be able to use drag-and-drop for manipulating the visually presented robot model ("user mode").
  - It must be possible to group elements in order to use multiple elements as if they were one ("user mode").
  - Duplicating (groups of) elements via "Copy and Paste" must be supported ("user mode").
- View/edit the sensors and actuators of the ready-made or block-assemble robot ("expert mode").
- Add standard sensors and actuators to the ready-made or block-assembled robot ("expert mode").
  - User must be able to define own sensors and actuators ("expert mode")
  - User must be able to position, rename sensors and actuators and configure any of their parameters ("expert mode")
  - User must be able to attach and combine noise(s) to sensors and actuators, either through selection of predefined noise types, or through a "noise composer" ("expert mode")
  - The model-component of the Robot Designer holding the information about the virtual robot (shape and physical properties) must in principle be able to be exposed for reading and writing via an API ("guru mode")

3.3.1.2 Define a kinematic chain (expert, guru) (SP10NRP-FR-002)

User must be able to:

- Define a kinematic chain by building a tree-like structure of the single links (parenting) ("expert mode")
- Group kinematic chains ("composite pattern") ("expert mode")
- Add constraints on a kinematic chain (X, Y, Z, Alpha-Beta, Gamma-constraints) ("expert mode")
- Use advanced methods for calculating forward kinematics ("guru mode")

3.3.1.3 Import design primitive (expert) (SP10NRP-FR-003)

- User must be able to import design primitives by specifying the file to be opened in an import dialogue ("expert mode")
- System must validate input and must not try to import the design primitives if not possible ("expert mode")

[1] Not scheduled for the ramp-up phase

3.3.2 Robot edition (expert) (SP10NRP-FR-004)

User must be able to:

- Select parts of the robot ("expert mode")
- Edit a robot part's graphical attributes ("expert mode")
- Edit a robot part's physical attributes ("expert mode")
- Edit a robot's sensor attributes ("expert mode")
- Edit a robot's actuator attributes ("expert mode")
- Modify a sensor's or actuator's noise property ("expert mode")
- Edit a robot's kinematic chain ("expert mode")

3.3.3 Storage

3.3.3.1 Save designed robot (user, expert) (SP10NRP-FR-005)

- User must be able to save a model of a virtual robot ("user mode")
- It must be possible to save a model with a simple GUI interaction (e.g. clicking a button) ("user mode")
- The file format must be well-defined as e.g. XML
- The file format used should be as exchangeable as possible (e.g. a standard file format like COLLADA)
- User must be able to export the robot for use in other software platforms ("expert mode")

3.3.3.2 Load designed robot (user, expert) (SP10NRP-FR-006)

- User must be able to load a model of a virtual robot from a file ("user mode")
- It must be possible to load a model with a simple GUI interaction (e.g. clicking a button) ("user mode")
- The import of a virtual robot model from an arbitrary file must be validated
- It must be able to import different de facto standard file formats which describe robots such as URDF
- User must be able to import a robot model from a compatible Robot Designer ("expert mode")

3.3.4 Change visualisation settings (user, expert) (SP10NRP-FR-007)

User must be able to:

- Rotate the view along the x-, y-, z-axis ("user mode")
- Zoom in or out in order to modify the virtual robot ("user mode")
- User must be able to show/hide the coordinate system, bounding boxes, contact points, connector/joint axes, camera frustums, (distance/light) sensor rays, (spot)lights, centre of mass etc. ("expert mode").

3.4 Non-Functional Requirements

The Robot Designer inherits all Non-Functional Requirements exposed in Non-Functional Requirements of Section 2.

3.4.1 Predefined models

- A set of predefined (including anthropomorphic/musculoskeletal) robots has to be included within the software.
- A set of predefined noise types (acoustic white, random peaks, regular/random light flickering…) should be available.
- Every function of the Designer should have a scriptable version.

3.5 Architectural Overview

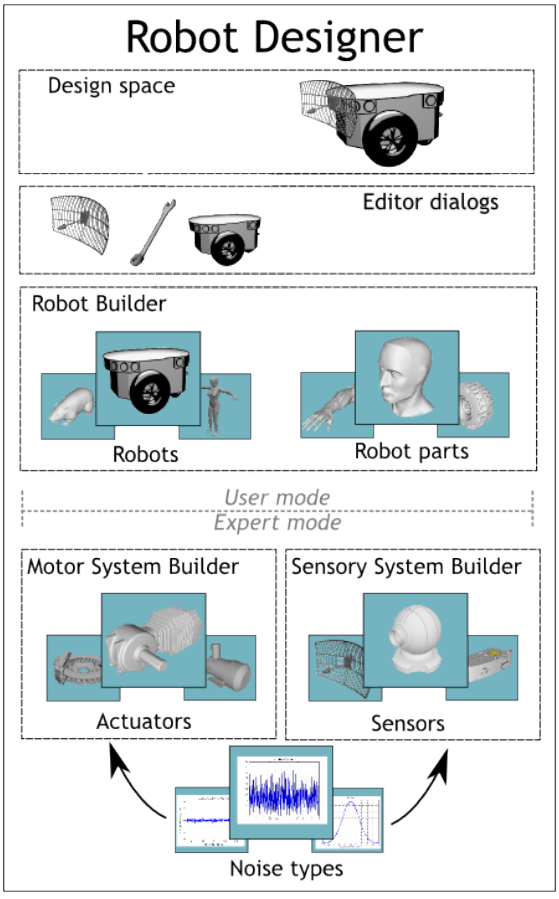

Figure 8 Robot Designer architectural overview

The Robot Designer component consists of five logical main building blocks:
Model (not represented in the diagram)

- Internal (memory) representation of the shape and physical properties in a tree-like structure can be read from a well-defined file (e.g. XML).
- Structure must, in principle, be able to expose its internals to external "consumers" for reading and writing.
- The model component only holds the information and is not in charge of its visualisation.

Design space

- This view-component must be able to display the information of the model in 3D space and offer interaction capabilities to edit properties in an intuitive way.

Editor dialogs

- They enable detailed properties of the robot and parts to be set in an intuitive way.
- They interface with the libraries.

Robot builder

- The robot library, robot parts library are graphical tools that enable the user to pick high-level elements from remote or local storage and build a complete robot.

Motor and Sensory System builder

- They use the sensors, actuators and noise types libraries to set up kinematic and sensory capabilities on the robot.

The beginner user should see a "user mode" level of the application, which shows only the robot builder GUI, for quick setup. The more advanced user should have further access to "expert mode" level GUIs like the motor and sensory system builder. They allow them to change or customise default actuators/sensors or add new ones from scratch.

3.6 Relations to other Platforms

The Robot Designer (as well as other component of the NRP cockpit) will use the services provided by the Unifying Portal (See 2.8.1 Unifying Portal (developed by SP6)).

The Robot Designer has no other specific dependencies to other platforms.

However, the UI and the overall user experience in the Environment Designer needs to be consistent with the Unifying Portal from SP6 as well as with the other components of the NRP Cockpit.

4. Environment Designer

4.1 Overall Goals

The Environment Designer allows users to design virtual environments for interactions with simulated robots in the context of virtual experiments. The user can either refer to a library of built-in design primitives, or model completely new objects by specifying their appearance as well as physical properties. Environments also support rigid and soft bodies as well as fluid, smoke and particles. Furthermore, dynamic objects – which may move or

fall down – can be placed within virtual scenery. These dynamic objects expose their properties and functions for other components as e.g. for the Experiment Designer. By providing a scripting interface, rich (dynamic) environments can be created programmatically. Since not every user is familiar with programming, different levels of expertise are supported.

The Environment Designer strives for usability and understandability. A user should be able to intuitively create a suitable environment without any knowledge in programming. Nonetheless can experienced users access more complex editing options, if they want to.

High exchangeability is another major goal of the Environment Designer. The users should be enabled to exchange their creations with any other user – regardless if they use the Environment Designer as well or other tools. Hence standard formats and respective import and export capabilities are supported. An easy storage facility (within the Unifying Portal) will further support this goal.

In order not to reinvent the wheel existing open source solutions provide the basal layers. They are then extended for enabling previews in different fidelities (up to physically correct rendering) and support different visualisation types, from browser-based applications to a CAVE.

4.2 Use Cases

4.2.1 Assemble and model virtual environment (SP10NRP-UC-017)

Primary Actors: One Neuroscientific User: Abigail
Success scenario:

1) Abigail opens the model editor.
2) Abigail draws/models 2D and 3D contents using intuitive graphical user interfaces' elements OR,
3) Abigail selects existing 2D and 3D objects from a local library of available 2D and 3D objects.
4) Abigail loads the selected 2D and 3D objects into environment design space.
5) Abigail places the 2D and 3D objects together to assemble the virtual environment.

4.2.2 Define and save a building block (SP10NRP-UC-018)

Primary Actors: One Neuroscientific User: Abigail
Precondition: The virtual environment is loaded in the design space.
Success scenario:

1) Abigail selects parts of the environment (single object or a group of objects) and defines and saves them as a building block into the local library.
2) Abigail saves the single building block in the local library.
3) The building block is saved in the same file format as the virtual environment (as in 4.2.7 Save virtual environment).

4.2.3 Load/import/export (SP10NRP-UC-019)

Primary Actors: One Neuroscientific User: Abigail

Success scenario:

1) Abigail opens the Load/Import/Export dialog.
2) Abigail loads the previously modelled and designed building blocks from a local library for 4.2.5 (Edit virtual environment) or 4.2.8 (Visualise the virtual environment).
3) Abigail imports 2D and 3D objects from a library of available objects (support of objects' standard formats).
4) Abigail exports modelled and designed 2D and 3D objects or building blocks into commonly used standard formats.

4.2.4 Automatically generate a virtual environment from a script (SP10NRP-UC-020)

Primary Actors: One Neuroscientific User: Abigail and one Scientific Developer: Bill.
Success scenario:

1) Abigail loads and runs a script (e.g. in a domain specific language) that automatically places building blocks from the library in the design space of the virtual environment.
2) Abigail modifies the placed objects from the script as specified in 4.2.5 (Edit virtual environment).
3) Bill writes, edits and manages scripts.

4.2.5 Edit virtual environment (SP10NRP-UC-021)

Primary Actors: One Neuroscientific User: Abigail
Precondition: The virtual environment is loaded in the design space.
Success scenario:

1) Abigail opens Editing dialog
2) Abigail modifies the spatial properties of the object (position in 3D space, orientation, scale...etc.).
3) Abigail modifies the surface properties and the appearance of the virtual object (e.g. index of refraction, transparency, texture, material...etc.)
4) Abigail modifies the physical properties of the virtual environment (e.g. mass, elasticity...etc.)
5) Each object is loaded in the design space with default properties that can be initially used with.

4.2.6 Define kinematic chain (SP10NRP-UC-022)

Primary Actors: One Scientific Developer: Bill.
Precondition: The virtual environment is loaded in the design space.
Success scenario:

1) Bill opens kinematic chain editor.
2) Bill selects a link on the virtual environment.
3) Bill pops up a kinematics dialogue.
4) Bill defines a kinematic chain (of single links or by grouping already existing kinematic chains). This applies to connected objects with rigid or elastic links, e.g. doors.

5) Bill closes dialog.

4.2.7 Save virtual environment (SP10NRP-UC-023)

Primary Actors: One Neuroscientific User: Abigail
Precondition: The virtual environment is loaded in the design space.
Success scenario:

1) Abigail chooses a local or remote location.
2) Abigail saves the virtual environment in the form of an archive file. The archive file format encapsulates both the standard 3D graphics application file format in addition to the proprietary properties (physical properties) specific to the Environment Designer tool.

4.2.8 Visualise the virtual environment (SP10NRP-UC-024)

Primary Actors: One Neuroscientific User: Abigail.
Precondition: The virtual environment is loaded.
Success scenario:

1) Abigail previews a low fidelity visualisation of the virtual environment.
2) Abigail selects to render the virtual environment with a physical correct condition and a correct light behaviour.
3) Abigail selects the rotation controller and rotates the whole view.
4) Abigail selects the zoom controller and zooms into the robot.
5) Abigail opens a view dialog and selects what should be displayed (axes, bounding boxes, etc.)

4.3 Functional Requirements

4.3.1 Assemble and model a virtual environment (SP10NRP-FR-008)

- The user must be able to instantiate any number of objects.
- The object parameters must be viewable and editable.
- The user is able to select (e.g. drag and drop) objects from a local library of available objects.
- The user is able to define building blocks of the virtual environment.
- The user is able to save the building blocks as assets.

4.3.2 Import/export/load/save (SP10NRP-FR-009)

- The environment must be stored in a well-defined archive file format.
- The file format must include both the standard and original and chosen 3D graphics application and the necessary additional proprietary properties in a sidecar file.
- All required files making up an environment should be compiled into a standard archive.
- Import of objects from commonly used file formats must be possible.

- Export the virtual environment to commonly used file formats must be possible.

4.3.3 Edit virtual environment (SP10NRP-FR-010)

- The virtual environment objects and content must be editable and viewable.
- Both graphical and physical properties of the virtual environment blocks must editable.
- The virtual environment blocks are loaded with meaningful default properties.

4.3.4 Define a kinematic chain (SP10NRP-FR-011)

- Ability to define a kinematic chain by building a tree-like structure of the single links (parenting).
- Possibility to group kinematic chains ("composite pattern").
- Possibility to add constraints on a kinematic chain (X, Y, Z, Alpha-Beta, Gamma-constraints).
- Advanced methods for calculating forward kinematics must also be supported.

4.3.5 Visualise the virtual environment (SP10NRP-FR-012)

- The user must be able to preview and visualise the environment at any time.
- The user must be able to rotate the view along the x-, y-, z-axis ("user mode").
- The user must be able to zoom in or out in order to modify the virtual environment ("user mode").
- The user must be able to show/hide the coordinate system, bounding boxes, contact points, camera frustums, (distance/light) sensor rays, (spot)lights, centre of mass etc. ("expert mode").

4.3.6 User interaction (SP10NRP-FR-013)

- The user interaction should support navigation and authoring using intuitive 3D interaction techniques.
- The basic visualisation should be oriented towards authoring the environment with all the relevant object attributes.

4.3.7 Automatically generate a virtual environment from a script (SP10NRP-FR-014)

- The world model must be accessible in a programmatic fashion from within the designer.
- User must be able to load and run domain specific user scripts that automatically design parts of an environment.

4.4 Non-Functional Requirements

The Environment Designer inherits all Non-Functional Requirements presented in Section 2.

4.5 Architectural Overview

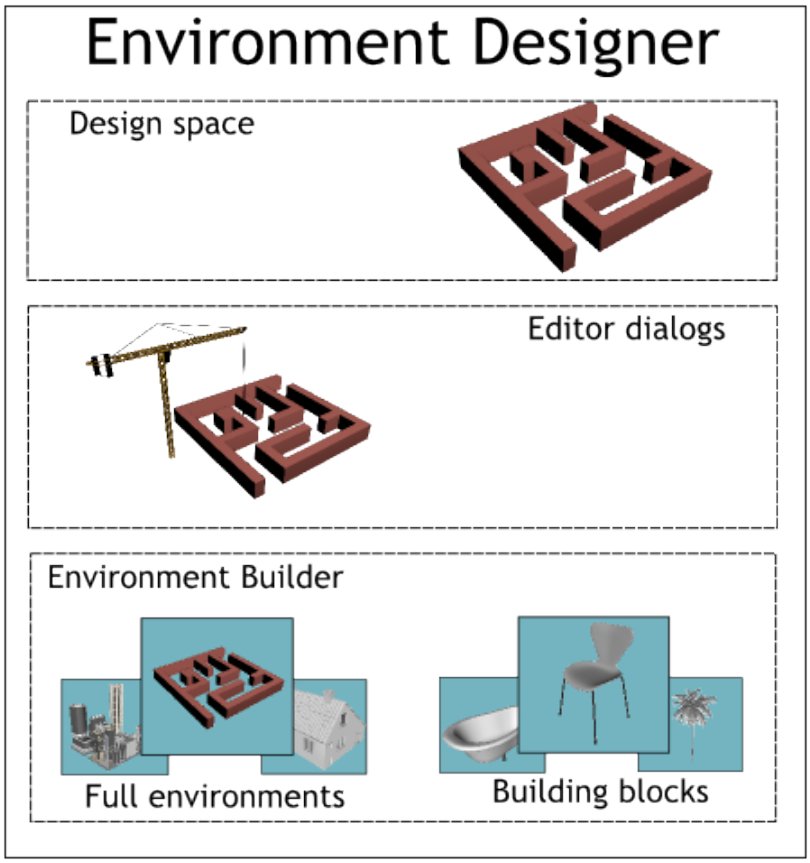

Figure 9 Environment Designer architectural overview

The Environment Designer consists of four logical blocks:
Model (not represented on the diagram)

- Representation of the environment (composition of all the individual objects)
- Each object includes its geometrical representation and some proprietary parameters.

Design space (View)

- The design space shows the 3D representation of the spatial arrangement of the objects
- Various optical properties of the objects should be displayable.
- The arrangement should directly be modifiable (drag and drop, editor dialogs).
- The system should also support immersive interaction in a CAVE like environment.

Editor dialogs

- These more detailed dialogs enable to change detailed properties and to have access to the libraries.

Libraries

- Ready-made environments and building blocks should be available via library.

4.6 Relations to other Platforms

The Environment Designer (as well as other component of the NRP cockpit) will use the services provided by the Unifying Portal (See 2.8.1 Unifying Portal (developed by SP6)).

The Environment Designer has no other specific dependencies to other platforms. However, the UI and the overall user experience in the Environment Designer needs to be consistent with the Unifying Portal from SP6 as well as with the other components of the NRP cockpit.

5. Brain Interfaces & Body Integrator

5.1 Overall Goals

The Brain Interfaces & Body Integrator (BIBI) main goal is to let the user choose a brain simulator and a brain model to run on it, and then let them wire it to a virtual robot. This wiring is used during the simulation to transfer data from the sensors of the virtual robot to the brain simulation and from the brain simulation to the actuators of the virtual robot. This data flow is taken care of in the Closed Loop Engine.

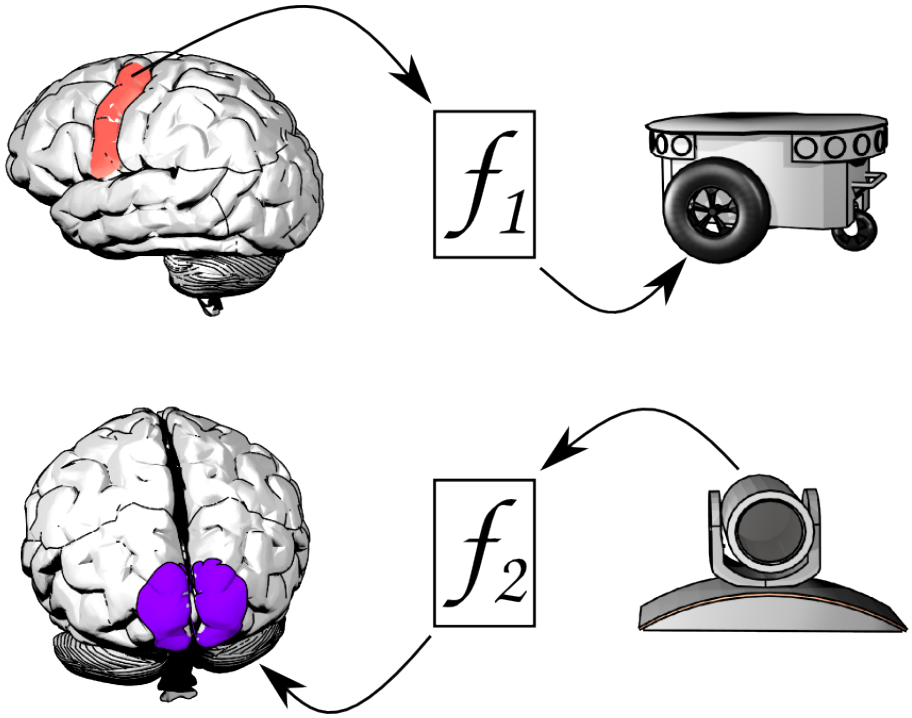

Figure 10 Brain Interfaces & Body Integrator mapping overview

At the beginning of the brain interfaces' setup, the BIBI will let the user choose one brain model available from the "brain builder" component of the Brain Simulation Platform. The "brain builder" is defined in the DoW:

> "Development of a Brain Builder that allows the specification and construction of models to be simulated. Important functions to be provided include automated workflows to reconstruct the whole brain, brain regions, microcircuits, neurons, glia and synapses. The Brain Builder will be a key component of the Brain Simulation Platform, allowing collaborative reconstruction of brain models and collaborative planning and execution of in silico neuroscience experiments by groups of scientists working over the Internet (T6.1.1)."

The Unified Portal Services (provided by SP6) is the glue that allows users to access brain models built by the "brain builder" (also provided by SP6) from within the BIBI. These services take care of the access rights. They also let the user choose the level of simulation and the platform, i.e.:

- Network simulators (also referred as point-neuron simulators).
- Neuromorphic hardware based simulators.

The network simulators category contains several well-known simulators (like NetSim) that are ready to use. They need less computing power than the detailed simulators. The NRP will try to make use of them from the beginning of the ramp-up phase. Depending on the state of SP9 - WP9.2 (Neuromorphic computing with digital many-core implementation of brain models) and the availability of the hardware under development by this SP, the NRP may support neuromorphic hardware during the ramp-up phase.

Using detailed simulators running on HPC hardware (for instance NEURON) within the NRP may happen towards the end of the ramp-up phase or after it. The main challenge is that SP6 is not yet ready for interactive simulation with detailed models. Another challenge is to have enough computing power for both the detailed model simulation and the world simulation. Nevertheless, the NRP's development will be based on communication interfaces that will be compatible with detailed simulators. This will facilitate their future integration.

Once a simulator and a model are chosen, users are able to map brain signals (spikes and currents) to the robot sensors and actuators. In a first step, users select groups of neurons that they want to map to one robotic function (either a sensing or an actuating function). The BIBI provides them with a navigable view of the brain model they have chosen (see Figure 11). The displayed neurons should be selectable graphically or using some sort of query language. For example, users should be able to select all neurons of a particular region of the Occipital Cortex. The navigation and selection tool is based on the "Brain Atlas Embedding Module" provided by SP6.

Users should be able to map selected groups of neurons to transfer modules and then to the robot actuators and sensors. The different brain simulators provide or consume spikes, spike rates and current. The robot sensors and actuators on the other side provide and consume other types of data such as forces, images or sounds. Transfer modules will act as a translating mechanism between the brain simulation and the robot. They are scriptable objects that users create or choose from a provided library.

Transfer modules can also be connected in a special way to let users implement reflexes for the robot and learning from the environment:

- A transfer module can provide a signal to actuators and at the same time take a signal from a sensor. This kind of bypass of the brain is allowed in order to simulate reflexes (see transfer function 2 in Figure 11)
- A transfer module can take input from the overall experiment simulation. This kind of input allows the validation and the early usage of the platform: If the experiment is about studying brain reaction through sensory input to the retina, one could imagine linking images directly to the retina instead of having the images displayed on a virtual screen in the virtual environment and viewed by a virtual robot sensor.

Transfer modules can also be chained (see transfer modules 1, 2 and 3 in Figure 11). Some part of the translating mechanism between the brain and one type of sensor can be used for other types of sensors (or actuators). Therefore, the user should be able to define it once and re-use it several times.

The NRP aims to let users write and use transfer modules between the retina and the brain, between the muscle and the brain and between the ears and the brain (see overall goals of the platform). The transfer module framework will have to technically allow this kind of mapping. It is designed to handle various kinds of transfer modules: simple ones like image processing algorithms and more complex ones like a spinal cord model. Moreover, the transfer modules framework should allow the transfer module code to retain its state between each simulation step (brain to robot and environment to brain).

After setting up the brain simulation and its interfaces with the robots, users should be able to save their work in a standard format. The NRP platform team will make sure that this format is open and re-usable by any third party robotics platform that would like to connect to brain simulators. If the Neurorobotics Platform is able to attract other robot

simulators by allowing them to use the HBP brain simulation capabilities, the research community will be able to cross-check their experiments.

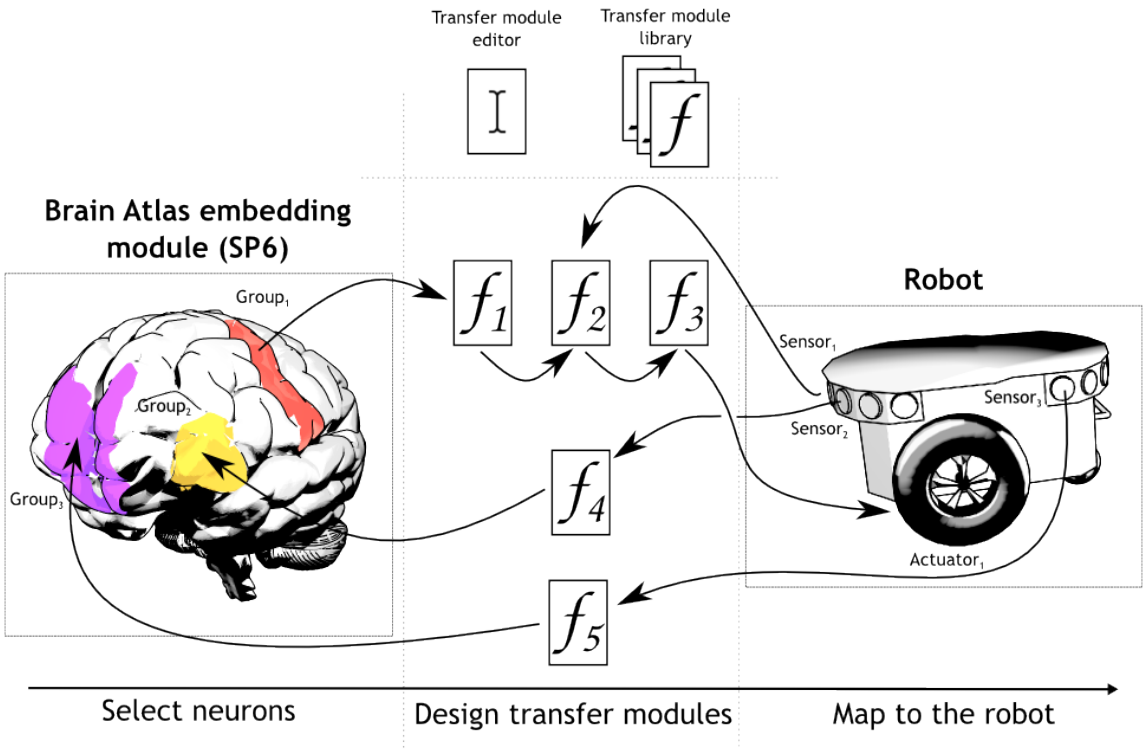

Figure 11 User steps needed to create a Neurobot out of a brain model and robot model.

5.2 Use Cases

5.2.1 Selection of the brain model (SP10NRP-UC-025)

Primary Actors: One Neuroscientific User: Abigail
Success Scenario:

1) Abigail works on a brain model on the brain builder component of the Brain Simulation Platform.
2) Once she has finished her model, she saves it.
3) She navigates in the Neurorobotics Platform.
4) She opens the Brain Interface & Body Integrator module.
5) She can select the brain model on which she was working on.

5.2.2 Selection and grouping of neurons (SP10NRP-UC-026)

Primary Actors: One Neuroscientific User: Abigail
Precondition:

- Abigail has created a brain model in the brain builder component of the Brain Simulation Platform.

- She has selected this model in the NRP.
- She is logged in the NRP.

Success Scenario:

1) She sees on her screen a 3D representation of the brain model she chose.
2) With her mouse and her keyboard, she navigates (zoom and move) from the complete view of the brain to a region containing several thousand neurons.
3) She selects all these neurons and assigns them a name: $group_1$.
4) She saves the project. The grouping of neurons $group_1$ is saved.

5.2.3 Selection and grouping of neurons – query (SP10NRP-UC-027)

Primary Actors: One Neuroscientific User: Abigail
Precondition:

- Abigail has created a brain model in the brain builder component of the Brain Simulation Platform.
- She has selected this model in the NRP.
- She is logged in the NRP.

Success Scenario:

1) She sees on her screen a 3D representation of the brain model she chose.
2) Through a query interface, she selects all neurons belonging to layer 5 of the column and assigns them a name: $group_3$.
3) She saves the project. The grouping of neurons $group_3$ is saved.

5.2.4 Transfer modules creation (SP10NRP-UC-028)

Primary Actors: One Scientific Developer: Bill
Success Scenario:

1) Bill logs in the Neurorobotics Platform.
2) He navigates to the new transfer module edition interface and creates a new empty transfer module.
3) He defines the input as the spike rate of a group of neuron.
4) He defines the output as a force.
5) He programs the algorithm transforming the input into the output.
6) He saves the transfer modules $transfer_1$ with group access rights letting Abigail access it.

5.2.5 Transfer modules connection – simple (SP10NRP-UC-029)

Primary Actors: One Neuroscientific User: Abigail
Precondition:

- Abigail has selected a brain model and has several defined group of neurons at her disposal.

- Abigail already went through the Robot Designer and has a robot with sensor and actuators at her disposal.

Success Scenario:

1) She selects the neuron group $group_1$ and visually links it to the transfer module $transfer_1$ done by Bill previously.
2) She selects the $transfer_1$ transfer module and visually links it to one actuator $actuator_1$ of the robot.
3) She saves the project. The connections are saved.

5.2.6 Transfer modules connection – chain (SP10NRP-UC-030)

Primary Actors: One Neuroscientific User: Abigail
Precondition:

- Abigail has selected a brain model and has several defined group of neurons at her disposal.

Success Scenario:

1) She selects the neuron group $group_1$ and visually links it to the transfer module $transfer_1$ done by Bill previously.
2) She selects the $transfer_1$ transfer module and visually links it to another transfer module $transfer_2$ publically available.
3) She selects the $transfer_2$ transfer module and visually links it to one actuator $actuator_1$ of the robot.
4) She saves the project. The connections are saved.

5.2.7 Transfer modules connection - direct feedback (SP10NRP-UC-031)

Primary Actors: One Neuroscientific User: Abigail and one Scientific Developer: Bill.
Precondition:

- Abigail has selected a brain model and has several defined group of neurons at her disposal.

Success Scenario:

1) Bill logs in the Neurorobotics Platform.
2) He copies his transfer module $transfer_1$ and rename it $transfer_3$.
3) He adds one more type of input to the transfer module: a pressure sensor input.
4) He modifies his algorithm so that when the pressure sensor input is on, the output force of the transfer modules is directly changed whatever comes out of the brain. He is implementing reflexes.
5) He saves the transfer module $transfer_3$ with group access rights letting Abigail access it.
6) Abigail logs in the neurorobotics platform and opens her project.
7) She selects the neuron group $group_1$ and visually links it to the transfer module $transfer_3$ done by Bill previously.

8) She selects the *transfer*$_3$ transfer module and visually links it to one actuator *actuator*$_1$ of the robot.
9) She selects a sensor *sensor*$_1$ of the robot and visually links it to the transfer module *transfer*$_3$.
10) She saves the project. The connections are saved.

5.2.8 Transfer modules connection - environment connection – sensors (SP10NRP-UC-032)

Primary Actors: One Neuroscientific User: Abigail and one Scientific Developer: Bill.
Precondition:

- Abigail has selected a brain model and has several defined group of neurons at her disposal.

Success Scenario:

1) Bill logs in the Neurorobotics Platform.
2) He navigates to the new transfer module edition interface and creates a new empty transfer module.
3) He defines the output as a current applied to a group of neurons.
4) He defines the input as an image coming from a given folder.
5) He programs the algorithm transforming the input into the output (a retina model).
6) He saves the transfer module *transfer*$_5$ with group access rights letting Abigail access it.
7) Abigail logs in the Neurorobotics Platform and opens her project.
8) She selects the neuron group *group*$_3$ and visually links it to the transfer module *transfer*$_5$ done by Bill previously.
9) She selects a virtual screen in the virtual environment and links it to the transfer module *transfer*$_5$. Doing like this means that the robot does not need a visual sensor. The environment plays this role.
10) She saves the project. The connections are saved.

5.3 Functional Requirements

5.3.1 Brain model (SP10NRP-FR-015)

- User must be able to use whatever brain model they built in the brain builder component of the Brain Simulation Platform.
- User must be able to use whatever brain model they have access to in the brain builder component of the Brain Simulation Platform.

5.3.2 Transfer modules (SP10NRP-FR-016)

- User must be able to create transfer module.
- Transfer module algorithms are written in a well-known language (like Python).

- Transfer modules can have as input or output variables that are produced or consumed by the brain simulation (currents, spikes and spike rates).
- Transfer modules can have input or output variables that are produced or consumed by the sensor or the actuators of the robot.
- Transfer modules can have as input variables produced by the environment.
- Transfer modules can handle intensive computations such as the simulation of a spinal cord. They should not be the bottleneck of the complete simulation. Therefore, they should allow parallelisation on HPC resources.
- Transfer modules can hold a state. State is defined as a set of variable that keeps some value in between the loops.
- Transfer modules can be saved and reloaded.
- Transfer modules can be shared to a group of people or to everyone.

5.3.3 Connect group of neurons to actuators and sensors (SP10NRP-FR-017)

- User must be able to select groups of neurons on different levels (connectomes on micro, meso and macro scale).
- User must be able to select groups of neurons based on criteria (query language)
- Groups of neurons can be connected to a transfer module and vice-versa.
- Sensor and actuators can be connected to a transfer module and vice-versa.
- Environment objects can be connected to a transfer module.

5.3.4 Project load/save (SP10NRP-FR-018)

The main data entity (or container) is referred here as the user project.

1) Selection of the brain model is saved in a user project.
2) Groups of neurons are saved in a user project.
3) Connections between groups of neurons, transfer modules, sensor / actuator and environment modules are saved in a user project.

5.3.5 Re-calibration of the brain model (SP10NRP-FR-019)

After an experiment, based on some results, a user can tune the parameters of the brain models (like changing weights on connections between neurons) and re-run the experiment.

5.3.6 Snapshot region (SP10NRP-FR-020)

The Experiment Designer allows the user to save snapshots of some parts of the brain after an experiment. The snapshot consists of all the data required to set again the brain region in the same state (mainly currents and synapses connections).

- User must be able to select brain regions and named them for the snapshot functionality of the Experiment Designer.
- Named brain region snapshot should be schedulable in the Experiment Designer.
- Brain region snapshot definitions and actual snapshots are saved in a user project.

5.4 Non-Functional Requirements

The Environment Designer inherits all Non-Functional Requirements exposed in Non-Functional Requirements of Section 2.

5.4.1 Format

- User project is saved in a state-of-the-art, well-defined format.
- The robotic sensors and actuators descriptions used in the brain body integrator are saved in an open format, preferably in an existing standard.

5.5 Architectural Overview

The Brain Interface & Body Integrator can be separated into the following building blocks:

- The Brain Atlas Embedding Module (provided by SP6) along with the neuron grouping capabilities (select, name and save a group of neurons).
- The transfer modules management (editor, library, import/export).
- The view of robot sensors and actuators.
- The view of the environment connectable objects.
- The connection graphical interface between the atlas, the transfer modules, the robot and the environment.

5.6 Relations to other Platforms

The BIBI (as well as other components of the NRP cockpit) will use the services provided by the Unifying Portal (See 2.8.1 Unifying Portal (developed by SP6)).

The BIBI allows the user to select the models (use case 6.6.2.1) they built using the "brain builder" component provided by SP6. The brain builder is described in WP 6.1 (Data-driven reconstruction of brain models), Task 6.1.1. For the navigation and the selection of neurons in the brain (use case 6.6.2.2), the NRP integrates the "Brain Atlas Embedding Module" also provided by SP6. The Brain Atlas Embedding Module (BAEM) is described in WP 6.1, Task 6.1.2. Figure 6 shows the current prototype of the BAEM.

Since the development of the brain builder and the Brain Atlas Embedding Module spans the complete ramp-up phase, there must be continuous integration of these two components in the NRP. Combining the two sides at the end of the ramp-up phase is too risky. On the same subject, the point neuron simulations will be the first ones available at a real time rate. The NRP platform will mainly use point neuron simulation and neuromorphic hardware during the ramp-up phase. SP6 task 6.2.3 "The Network Simulator" holds the development effort of point neuron simulations. SP9 - WP 9.2 holds the development effort for the neuromorphic hardware.

6. Experiment Designer

6.1 Overall Goals

The Experiment Designer should enable a user (e.g. a neuroscientist) to define an experiment within a GUI. The result of such a definition is called an experiment set-up. It includes a specific combination of a virtual environment and one (or more) Neurobot(s) placed in this environment. Each of these elements has to be in a configured state within the experiment set-up: The Neurobot's appearance, physical and dynamic properties (e.g. kinematic chains) must have already been specified within the Robot Designer. The same holds for the objects defining the environment and the Environment Designer respectively.

First, the user of the Experiment Designer has to combine these single configurations, by specifying which Neurobot should take part in the experiment and in which environment.

In a second step the protocol of the experiment is specified. This includes:

- **Number of runs** of the experiment or,
- **Terminating condition(s)**. For example an experiment can be defined to terminate if the expected responses are encountered before the predefined number of runs is reached.
- **Measurement of data,** i.e. the definition of data to be recorded ("What to record when?"). Since writing measurement data to a file is very costly in terms of computation time it must be possible for the user to very selectively define what data should be recorded and at which time intervals. For example in an experiment with a mobile robot, only the end positions of multiple runs may be of importance but not the positions between start and end. Measurement is done through "measuring devices" which can be placed within the virtual environment (e.g. a light barrier) or within the brain model (as a multimetre for example). These measuring devices are physical objects that represent the measurement and are managed in a library of predefined devices.
- **Action Sequence** ("When to stimulate what?"). Users can specify time triggers or event triggers for actions. Furthermore they can define properties such as duration or intensity (depending on the concrete action). Therefore every dynamic object of the environment exposes its functionality. The user can then edit it via a graphical interaction element or via a script.

The user can also define that snapshots will be taken within the simulated experiment run as an own action. A snapshot saves the current execution state either of parts of the experiment, i.e. the brain state and/or the world state (Neurobot and environment) or of the entire experiment. Such snapshots can be loaded (in an action) in further runs, e.g. as new initial experiment conditions.

The Experiment Designer's usability and understandability should be very high. Therefore, it must also be easy to define experiment set-ups for users not familiar with programming (e.g. by a drag and drop-approach, visual programming or usage of a domain specific language). A set of pre-defined experiment set-ups should increase the efficiency and understandability. By using these as a reference, the user should have a quick start and fast progress towards defining their own experiments.

The storage format should be guided by high exchangeability of the experiment set-ups. It must be easy to interchange complete experiment set-ups from one user to another. This

must also hold for large data (e.g. brain models), where inclusion of all data is not feasible and references or links are more appropriate. In order to process the recorded experiment data, exportation of these data to *de-facto* standard file formats for statistical analysis tools must also be provided.

6.2 Use Cases

The use cases below describe success scenarios for Neuroscientific Users and Scientific Developers in high-level interactions with the Neurorobotics Platform. If not stated otherwise the Primary Actor of the use cases is a Neuroscientific User named Abigail.

6.2.1 Selection of a configuration element (SP10NRP-UC-33)

The following use case describes the selection of a configuration element of an experiment in the Experiment Designer. A configuration item is one of the following:

- A virtual environment.
- A Neurobot, i.e. a virtually assembled robot with a connection to a brain (model)

Since the procedure is very similar for each element the following description combines them.
Preconditions:

- Abigail already started the Experiment Designer, the Experiment Designer is running.
- The configuration element (virtual environment or Neurobot) to be selected is already successfully loaded.

Success Scenario:

1) Abigail navigates through a list of available (loaded) elements in the GUI.
2) A preview of the component is presented to her.
3) By using the preview and the name of the element Abigail clicks on the element to be selected for this specific experiment.
4) Abigail is able to check her selection in the overview of the experiment set-up.

6.2.2 Defining a measurement point (SP10NRP-UC-34)

Measurements within the experiment run are accomplished by placing devices for measurement in the virtual environment or within the simulation of the brain. For example one could place a stopwatch or a light barrier in the virtual environment or a millimetre in a certain area of the virtual brain.

Precondition:

- The virtual environment and the Neurobot are already loaded.

Success Scenario:

1) Abigail opens the virtual environment she wants to use for the experiment run.
2) In this environment (as e.g. a maze) she installs a light barrier that she selects from a predefined library of recording devices. She installs the recording device by simply drag-and-dropping it to the desired place (which may be the finish of the maze).

6.2.3 Defining an action sequence (SP10NRP-UC-35)

Primary Actors: One Neuroscientific User, Abigail as well as one Scientific Developer, Bill.
Precondition:

- The virtual environment, the robot model as well as the brain connection information is already configured within the Experiment Designer

Success Scenario:

1) Abigail opens the Action Sequence Editor
2) Abigail defines the duration of the experiment in a form. After entering the information a timeline with the information of the overall duration is displayed.
3) Abigail zooms to the point in time where she wants an action to occur. Units of time displayed on the timeline enable her to find the specific point quickly.
4) After Abigail clicked on a specific point in time a form pops up providing her with the possibility to create actions. While the point in time is already filled Abigail enters a name for this action.
5) The form displays a list of all non-static objects of the environment such as lights or moveable objects.
6) Abigail selects a light by its name (name was assigned in the Environment Designer).
7) A dialog opens with the exposed functionality of the light. These include elements as "turn on", "turn off", "set brightness to x".
8) Abigail selects "turn on" and saves the current state of the experiment set-up.
9) Abigail wants the light to increase its intensity according to a specific function after 50 seconds the experiment started. Since she is not able to develop this functionality herself she asks Bill to provide her a script implementing the desired functionality. Bill writes a small script and sends it to Abigail. Abigail includes this script in order to set up a second action.
10) After saving this second action the timeline displays two actions.

6.2.4 Configuring a protocol (SP10NRP-UC-36)

Precondition:

- The environment as well as the Neurobot to be used is already loaded.
- Measuring devices are already in place.
- An Action Sequence for the experiment is already defined.

Success Scenario:

1) Abigail opens the Protocol Editor.
2) Abigail enters the *number of runs* the experiment should be conducted. She wants the experiment to be repeated 15 times unless a certain behavioural pattern of the Neurobot does not occur earlier. Since she is able to describe this behavioural pattern in terms of measurement data she also enters as a termination condition. (As an example this could be a mouse robot finding the way out of a maze that can be specified by activating a light barrier.)
3) Abigail opens the form for data measurement.

4) Abigail wants to record the firing rates of the connectome in V1 that is connected to the robots visual sensor while the experiment is running. Per default no data is recorded. In order to enable the recording Abigail opens the form for selecting the recording data.
5) From the list of recordable data Abigail selects the connectome in V1 she is interested in and where she put a recording device into (e.g. by activating a checkbox)
6) Abigail closes the form. The system informs her to specify a storage location for the data to be recorded.
7) Abigail selects a storage location on her local disk and saves the state.
8) After that she closes the Protocol Editor. The Experiment Designer shows that the experiment set-up is in a ready to run state (e.g. green check) and that the simulation can now be started.

6.2.5 Defining an experiment set-up (SP10NRP-UC-37)

An experiment set-up is defined by executing the Use Cases:

1) 6.2.1 Selecting a virtual environment,
2) 6.2.1 Selecting a Neurobot
3) 6.2.3 Defining an action sequence as well as
4) 6.2.4 Configuration of protocol

6.2.6 Saving an experiment set-up (SP10NRP-UC-38)

Precondition:

- The experiment set-up is in a well-defined state; its configuration is in such a state that it can be run in simulation

Success Scenario:

1) Abigail selects the experiment set-up out of a list of currently loaded experiment set-ups in the GUI.
2) After selection Abigail clicks an element on the GUI. There she is provided with two possibilities namely to store the data either within the local file system or within the Unifying Portal. In the first case a file system browser is displayed which she uses to navigate to the storage location on her local computer. In latter case she has to authenticate herself in an own dialog and should then be able to store the data in the UP.
3) When saving the experiment set-up is completed the system informs the user. This also holds in case of an error e.g. when the directory is not writeable for the user.

6.2.7 Loading an experiment set-up (SP10NRP-UC-39)

Success Scenario:

1) Abigail clicks on an element on the GUI which opens a either a file system browser which she uses to navigate to the file wanted or list of stored experiment set-ups within the UP. In order to access it she has to authenticate herself.

2) After selecting either the file on the local disk or within the UP the system tries to load the given file.

3) The system shows the loaded experiment set-up in the overview of all loaded experiment set-ups. In case the file cannot be load a message is displayed to inform the user.

6.3 Functional Requirements

6.3.1 Configuring an experiment set-up (SP10NRP-FR-21)

The configuration of an experiment set-up consists of three parts: (i) a selection of a virtual environment and an Neurobot (ii) the definition of a protocol (including specifying number of runs and/or terminating conditions, defining the data to measure as well as an action sequence) as well as (iii) meta-information as e.g. the name of the experiment. Therefore the following is required:

- The system must enable the user to select the virtual environment and the Neurobot to use in the experiment.
- The system must enable the user to define the protocol of the experiment, i.e. specify the number of runs and/or terminating conditions, to define the course of events of the experiment that is called an “Action Sequence” and to define the measurements i.e. the data to be recorded while the experiment is running.
- The system must also enable the user to define additional metadata.

6.3.1.1 Defining a control flow (SP10NRP-FR-22)

- An experiment set-up must be configurable to terminate after a fixed number of runs.
- Additionally an experiment must be configurable to define a terminating condition that will end the run of the experiment set-up even before it reached the specified number of runs.

6.3.1.2 Defining action sequences (SP10NRP-FR-23)

- User must be able to define events (e.g. change of light intensity, moving of an object) occurring on a certain point in time
- User must be able to define the properties of an event (e.g. duration)
- Non-static objects of the virtual environment must expose their functionalities in order to trigger them from within the Action Sequence Editor
- It must be possible to specify complex events (by combining single or atomic events)
- User must be able to define own complex events (composed of several single or atomic events)
- In order to define more complex actions a scripting-interface must be available for Scientific Developers
- It must be possible to define several actions which will run in parallel

6.3.1.3 Measurement (SP10NRP-FR-24)

- User must be able to specify which data will be collected during each run of the experiment (e.g. sensor data by choosing from a list of available sensors, motor/muscle data, neuronal activities etc.), i.e. to define "when to measure what"
- Hence the user must be enabled to put "measuring devices" in the virtual environment, to the robot or in the brain

6.3.1.4 Specifying metadata of experiment set-ups (SP10NRP-FR-25)

- An experiment set-up must carry an identification possibility as e.g. a name chosen by the user

6.3.2 Storage of experiment set-ups

6.3.2.1 Snapshots (SP10NRP-FR-26)

- It must be possible to define actions to store snapshots (i.e. current execution status) as well as load already saved snapshots. (This feature is not scheduled for the ramp-up phase.)
- This holds for a simulated experiment in its entirety, but also for parts of it, namely brain models and world models (Neurobot and environment). (This feature is not scheduled for the ramp-up phase.)

6.3.2.2 File formats (SP10NRP-FR-27)

- A range of common (de facto) standard file formats must be supported for the output of recorded data (as tabular and spreadsheet formats, database formats, scientific data formats)
- User must be able to specify the format of measurement data in the protocol (e.g. CSV data)
- The complete configuration or definition of an experiment set-up must be storable in its entirety
- It must be possible to exchange stored definitions of an experiment set-up with other users of the platform
- It must be possible to exchange stored execution states with other users of the platform
- The experiment set-up must include every information needed to run the respective simulation

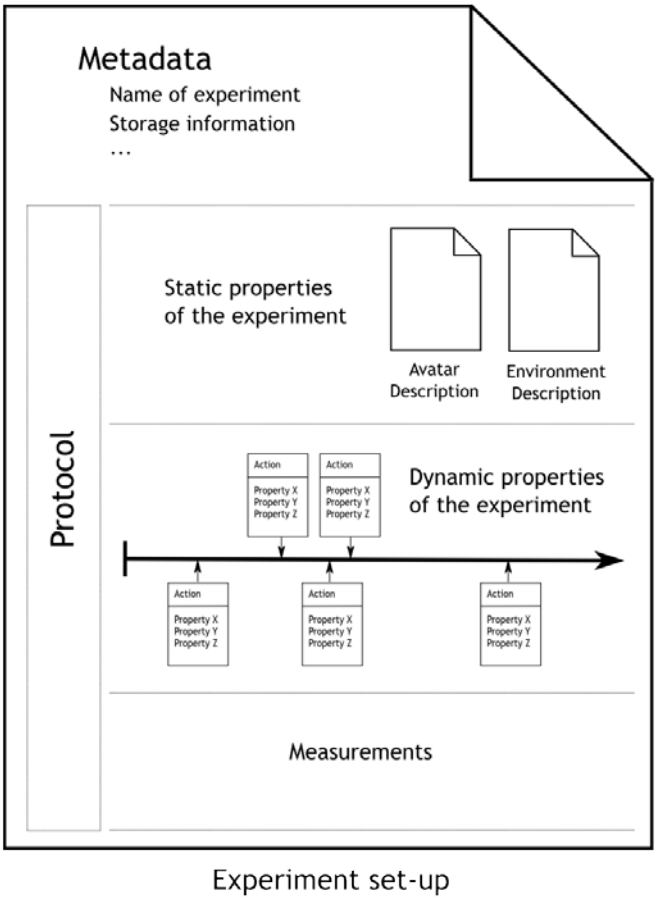

Figure 12 Overview of the data needed for an experiment.

6.3.2.3 Loading (SP10NRP-FR-28)

- It must be possible to load a definition of an experiment set-up
- It must be possible to load a execution state of an experiment run
- There must be a message about success or error of the finished loading process

6.3.2.4 Saving (SP10NRP-FR-29)

- It must be possible to save the current definition of the experiment set-up to the file system as a single file
- GUI must provide a possibility to save the current execution state of an experiment run.

6.4 Non-Functional Requirements

The Environment Designer inherits all Non-Functional Requirements exposed in Non-Functional Requirements of Section 2.

6.4.1 Exchangeability and interoperability

- The Experiment Designer should enable its users to exchange their experiment set-ups very easily.
- Furthermore the acquired measurement data should easily be processed with (de facto) standard analysis applications (e.g. for statistical analysis).

6.5 Architectural Overview

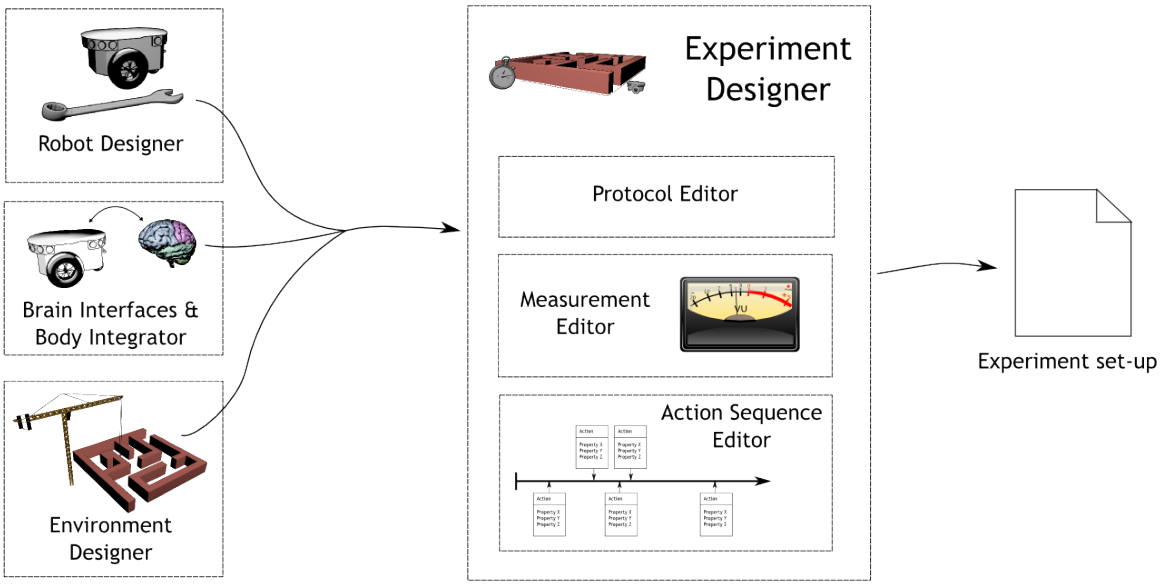

Figure 13 Architecture overview of the Experiment Designer

This section describes the high-level components that make up the Experiment Designer.

- Protocol Editor where the user can specify the properties of the experiment (metadata as number of runs, configuration of a combination of environment, robot and brain robot connection information, file format for measured data).
- Measurement Editor, which lets the user place virtual measuring devices in the virtual environment.
- Action Sequence Editor that enables the user to define events on a timeline.

6.6 Relations to other Platforms

The Experiment Designer (as well as other component of the NRP cockpit) will use the services provided by the Unifying Portal (See 2.8.1 Unifying Portal (developed by SP6)).
The Experiment Designer has no other specific dependencies to other platforms. However, the UI and the overall user experience in the Environment Designer needs to be consistent with the Unifying Portal from SP6 as well as with the other components of the NRP cockpit.

7. World Simulation Engine

The main responsibility of the World Simulation Engine (WSE) is to maintain the state of the virtual world during the simulated experiment. In this context, this encompasses the virtual environment, the virtual robot and their interaction.

7.1 Overall Goals

The World Simulation Engine (WSE) is responsible for simulating the virtual environment, based on its intrinsic dynamics (e.g. physical forces) and on its response to external influences (for example robot actions or experimenter interventions). The WSE is not directly visible by the user. It takes descriptions of virtual environments (and a robot which is part of it) and computes the physical processes therein for one or more time intervals. The calculated data can then be used by other components to visualise the simulation.

Depending on the experiment, it has to support physically realistic simulations on the one side (offline experiments) and fulfil real-time constraints on the other (online experiments). The WSE therefore provides a comprehensive modular simulation framework in which it is easy to switch single simulation modules.

In the ramp-up phase, the primary focus in simulation is on high-fidelity light simulation (e.g. for computing the sensed image for procession by a virtual retina) and on simulation of classical mechanics (rigid- and soft-body simulation). Further sensing modes of the robot directly necessitate further simulation modules to provide the corresponding stimuli. This includes for example the simulation of sound propagation for hearing stimuli or the simulation of heat and temperature.

As a direct consequence of this approach, the simulation framework must be highly modular. Each component should individually be exchangeable to provide a spectrum of different stimuli and different individual fidelities for different experiments. For example for an experiment running on a desktop machine, a simpler GPU-based light simulation can be sufficient, whereas another experiment may require physical correct light simulation performed on a HPC cluster.

Since the world simulation needs to be synchronised with the brain simulation to create a coherent experience for the experiment viewing user, the WSE provides a control interface (API) to set the simulation parameters; as for example from the Experiment Simulation Viewer.

7.2 Use Cases

The WSE does not expose its functionality directly to the user, but is instead controlled and configured via the Experiment Simulation Viewer. Thus the use cases are not directly applicable to the WSE with the exception of controlling the engine via an API.

7.2.1 Configuration and control via API (SP10NRP-UC-40)

Primary Actors: One Scientific Developer (SCIDEV), Bill.
Precondition:

- A fully configured experiment is available.

Success Scenario:

1) Bill instructs the WSE to load the world data from the experiment set-up via an API call.
2) Bill selects the individual simulation components to be used and configures them, also via API calls.
3) Bill specifies an amount of time or a condition as termination criteria of the simulation via an API call.
4) Bill starts the simulation via an API call.
5) The WSE simulates the experiment in conjunction with the Closed Loop Engine.
6) During the simulation, Bill can query its state (simulation time, simulation speed, real-time factor etc.) via API calls.
7) After the end of the simulation, Bill can query the results, as defined in the experiment set-up, and either decides to run further experiment simulation runs or not.

7.3 Functional Requirements

7.3.1 Configuration (SP10NRP-FR-30)

- The framework must support selection of individual simulation components for the different simulation categories (as light, mechanics etc.).
- It must further support extension by a plug-in mechanism in order to install additional simulation components.
- Each simulation component must support individual configuration of its parameters.

7.3.2 Load world state (SP10NRP-FR-31)

- The simulation framework must load the state of the world from a suitable archive file.
- This includes the state of the virtual environment and the state of the virtual robot including all parameters that are relevant to the simulation (for example physical mass or optical properties, kinematic information).
- This further includes scripted or injected influences as defined by the experimenter.

7.3.3 Save world state (SP10NRP-FR-32)

- The framework must be able to store the current state of the execution of a running (or finished) experiment
- This includes all parameters that are already listed in the loading section
- Further includes the current simulation time and all additional user interventions that were input during the runtime of the experiment.
- The saved state must be again loadable as a valid experiment at a later time.

7.3.4 Simulation control (SP10NRP-FR-33)

- The simulation framework must support start, stop and pause of an experiment at any time.

- The system must be capable of exposing its internal simulation execution speed.
- This can be presented to the user in the NRP cockpit.

7.3.5 World maintenance (SP10NRP-FR-34)

- The simulation framework needs to maintain a consistent model of the world during the execution of an experiment.
- The framework maintains a world clock and updates the world state in suitable time-slices.
- The world clock needs to be synchronised to the brain simulation clock to achieve a consistent, overarching notion of time.
- This synchronisation can for example be achieved through a mediator instance, such as the Closed Loop Engine.
- The simulation framework needs to be capable to inject actions into the simulation of the virtual world as defined by the experimenter.
- This may include either scripted actions which are triggered by certain simulated conditions (for example time passed or robot moves to specified position) or which may be created by the user during the runtime of the experiment (via the cockpit).
- To control the details of the simulation, an interface to modify any variable parameters must be exposed. This interface will be controlled via the Closed Loop Engine.
- The simulation should be completely reproducible.
- Thus an execution of the same experiment using the same starting values at a later time should produce exactly the same results as earlier executions.
- This is of special concern to the generation of random numbers during the execution of an experiment.
- The simulation framework must support additional spectator entities that monitor a running experiment.

7.3.6 World simulation components (SP10NRP-FR-35)

- The simulation framework consists of different sub-components responsible for different simulation categories.
- These sub-components are responsible for the actual computation of the simulation results.
- For each category at most one component can be active during the execution of an experiment.
- The fundamental required categories are light simulation and classical mechanics simulation.
- Different engines can be used in different components to provide a spectrum of fidelity. For examples:
  - Light simulation components: Desktop based rendering / GPU accelerated physically based rendering / HPC based physical correct rendering

– Classical mechanics: Rigid body simulation / Soft-body simulation

7.4 Non-Functional Requirements

The Environment Designer inherits all Non-Functional Requirements exposed in Non-Functional Requirements of Section 2.

7.4.1 Physical significance

- The simulated experiment needs to be relatable to the real world in terms of physical quantities.
- This requires the rigorous correspondence between the virtual simulation and real world physical measurements.
- Examples are the luminance of sensed light, the force of physical bodies colliding or the sound pressure of sensed sounds.

7.5 Architectural Overview

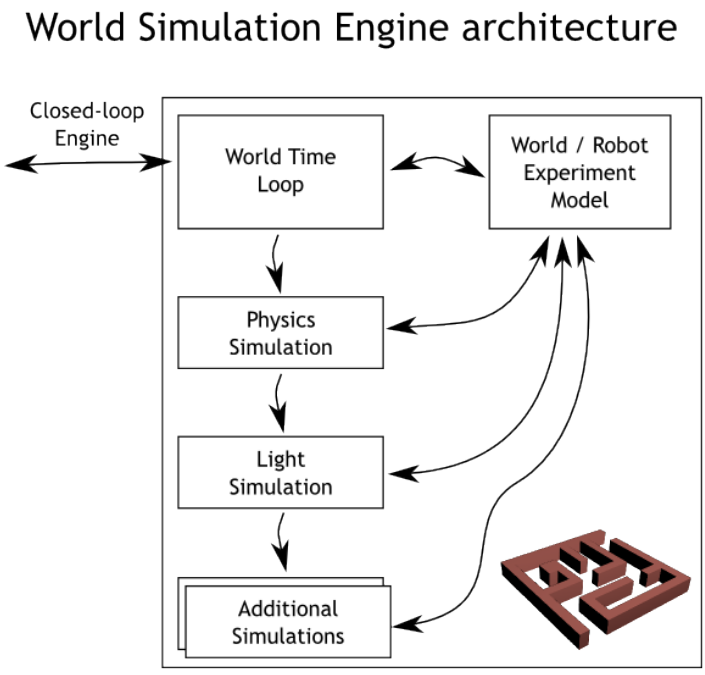

Figure 14 Architecture overview of the World simulation engine

The world simulation is structured in a modular way.

- The world time loop is a time management module that synchronises with the Closed-Loop Engine to manage steps of simulation.
- The physics simulation is the main simulation module taking care of fluid, soft and rigid body dynamics (including collision detection).

- Then, other modules, which can be added as wished, take care of various physical stimuli, like light (here mentioned in the figure), but also sound, etc.
- All the modules interact with and update the world model, which is a separate module.

7.6 Relations to other Platforms

The WSE is able to run on HPC resources. Therefore, collaboration with SP7 - WP75 (High Performance Computing Platform: integration and operation) is needed. The NRP will benefit from their extensive knowledge on HPC and the resources they provide in their platform.

8. Closed Loop Engine

8.1 Overall Goals

The Closed Loop Engine (CLE) is a reliable and robust integration and mediation layer that connects the brain simulation, the world simulation and the Experiment Simulation Viewer during the experiment. The CLE intervenes in the process after the experiment has been properly defined in the various designers (Robot Designer, Environment Designer, Experiment Designer and Brain Interfaces and Body Integrator). More precisely, the CLE comes on the scene when the user pushes the play button.

Regarding the user interaction with an experiment, two main scenarios can be distinguished:

- The offline scenario, where the simulation runs (non real-time) and produces results. Only after the experiment, these results are analysed by the user. The simulation results (what happened during the simulation) can be replayed as many times as needed. The user benefits from the visualisation components provided by the platform to inspect what happened during the simulation on the brain side as well as on the robotic/environment side.
- The online scenario, where the simulation runs in soft real-time. The user can interact with the brain part of the simulation or with the robotic/environment part of the simulation.

The two simulators used during an experiment (the brain simulation and the world simulation) are arranged in three possible setups (see Figure 15):

- The "Munich" setup, where the brain simulation and the world simulation are ran in Munich. The main reason of this setup is to allow the online experiments. Soft real time simulation on both sides as well as soft real time synchronisation needs the computing facilities to be collocated. The brain simulation has a typical cycle time of 0.1 [ms]. It's not yet possible to communicate in less than that with the world simulation located in Lugano and a Brain simulation located in Jülich for example. The CAVE in Munich is used for the Experiment Simulation Viewer.
- The "Web" setup where the brain simulation and the robotic simulation are ran in a facility that is not collocated with the user. Indeed, they use a light client or a web client interface to view or interact with the experiment. For the online scenario, the "Munich" setup without the CAVE is used.

- The “Developer" setup where every component is ran on a state of the art developer computer or a small development cluster.

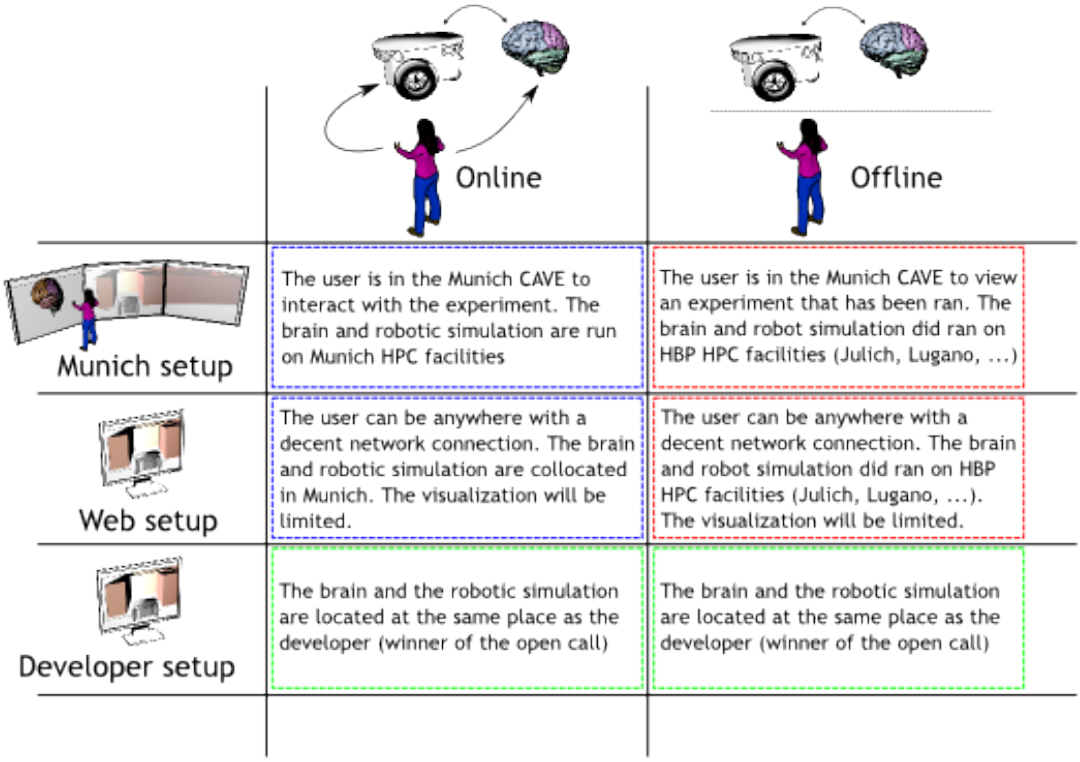

Figure 15 Physical architecture scenarios depending on the experiment.

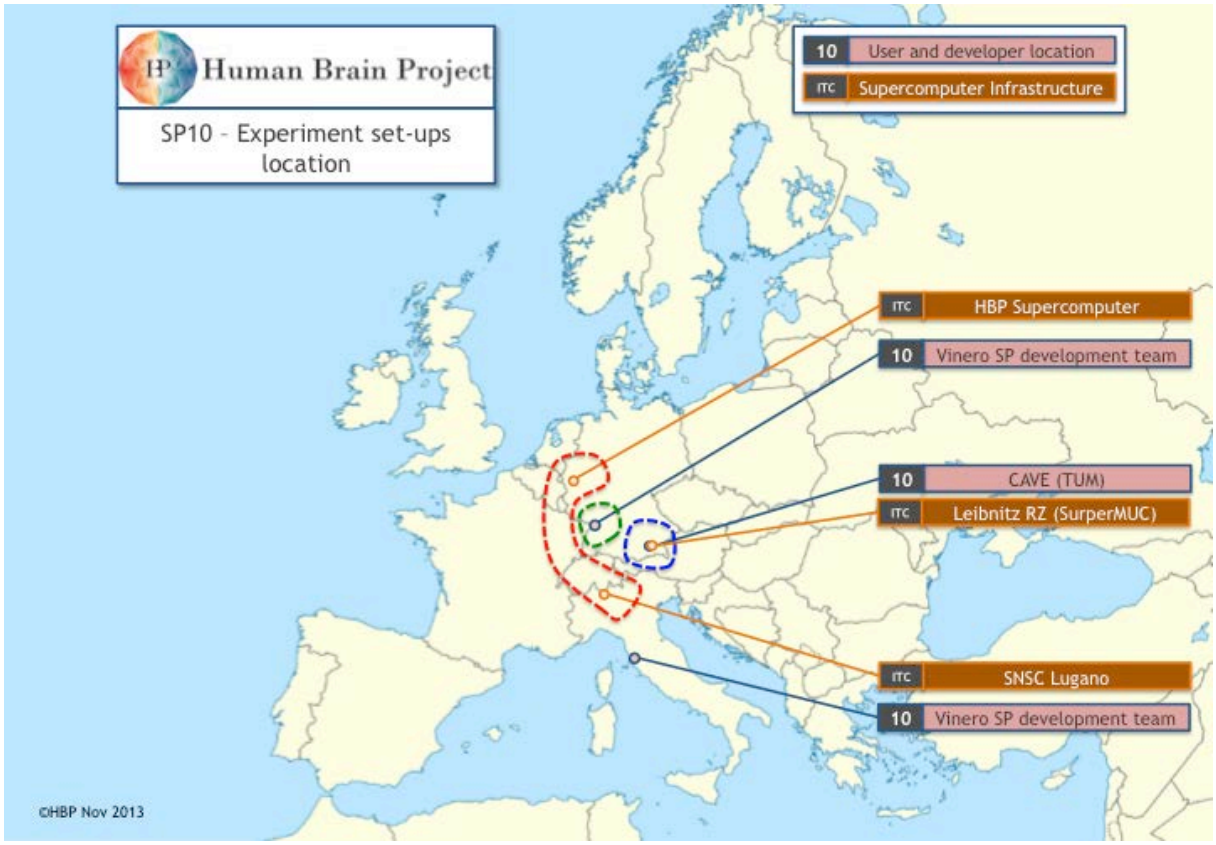

Figure 16 Resources location depending on the experiment (see previous figure)

8.2 Use Cases

8.2.1 Offline scenario with high fidelity visualisation (SP10NRP-UC-41)

Primary Actors: One Neuroscientific User: Abigail
Precondition:

- Abigail has prepared the simulation with the Robot Designer, the Environment Designer, the Brain Interfaces and Body Integrator and the Experiment Designer.

Success Scenario:

1) She launches the simulation in the Experiment Simulation Viewer from her computer.
2) She lets the simulation run and starts doing something else.
3) The brain and the world simulation run either in Jülich or Lugano (or both).
4) She gets notified that the simulation is completed.
5) She travels to the CAVE installation of Munich.
6) She loads the simulation in the CAVE (in the Experiment Simulation Viewer).
7) She replays it several times, each time changing her point of view in the brain visualisation and in the world visualisation.
8) From what she has seen, she can change the setup (in one of the designer) and repeat the full process.

8.2.2 Offline scenario without high fidelity visualisation (SP10NRP-UC-42)

Primary Actors: One Neuroscientific User: Abigail
Precondition:

- Abigail has prepared the simulation with the Robot Designer, the Environment Designer, the Brain Interfaces and Body Integrator and the Experiment Designer.

Success Scenario:

1) She launches the simulation in the Experiment Simulation Viewer from her computer.
2) She lets the simulation run and starts doing something else.
3) The brain and the world simulation run either in Jülich or Lugano (or both).
4) She gets notified that the simulation is completed.
5) She loads the simulation on her computer (in the Experiment Simulation Viewer)
6) She replays it several times, each time changing her point of view in the brain visualisation and in the world visualisation.
7) From what she has seen, she can change the setup (in one of the designers) and repeat the full process.

8.2.3 Online scenario with a high fidelity visualisation (SP10NRP-UC-43)

Primary Actors: One Neuroscientific User: Abigail
Precondition:

- Abigail has prepared the simulation with the Robot Designer, the Environment Designer, the Brain Interfaces and Body Integrator and the Experiment Designer.
- She is in the CAVE of Munich.

Success Scenario:

1) She launches the simulation in the Experiment Simulation Viewer from the CAVE.
2) The brain and the world simulation run in Munich.
3) She views the simulation in real-time (soft real time) in the CAVE.
4) She interacts with the simulation.

8.2.4 Developer scenario (SP10NRP-UC-44)

Primary Actors: One platform developer: Charlie
Success Scenario:

1) Charlie prepares the simulation with a relevant sample configuration in the Robot Designer, the Environment Designer, the Brain Interfaces and Body Integrator and the Experiment Designer.
2) Charlie launches the simulation on his local computer or on a local cluster.
3) Charlie can interact with the simulation. (The brain model may simplistic in order to have it running smoothly).

8.3 Functional Requirements

8.3.1 Communication (SP10NRP-FR-36)

- The loop between the brain simulator and the WSE needs to happen at a rate of an order of magnitude of 0.1 millisecond of simulated time.

8.3.2 Online experiments (SP10NRP-FR-37)

- For online experiments, the brain simulator, the WSE and the Experiment Simulation Viewer needs to be able to communicate at a rate allowing a soft real-time experiment.

8.3.3 Offline experiments (SP10NRP-FR-38)

- The infrastructure should let non-collocated simulator to be used for this kind of experiment.
- The user is able to find an approximation of the duration of the experiment.
- The user is able to find a status of the experiments they launched (in order to track the simulation computation's progress).
- The infrastructure prevents a user to use all the available computing power.

8.4 Non-Functional Requirements

8.4.1 Future development

- The architecture of the CLE will facilitate the integration of real hardware after the ramp-up phase.
- The architecture of the CLE will facilitate the integration of detailed brain simulation once SP6 is ready with them.

8.4.2 General architecture

The CLE is robust and fault tolerant in the sense of providing a well-defined behaviour for all inputs (e.g. a software malfunction within one simulator must not lead to a malfunction of the CLE).

8.5 Architectural Overview

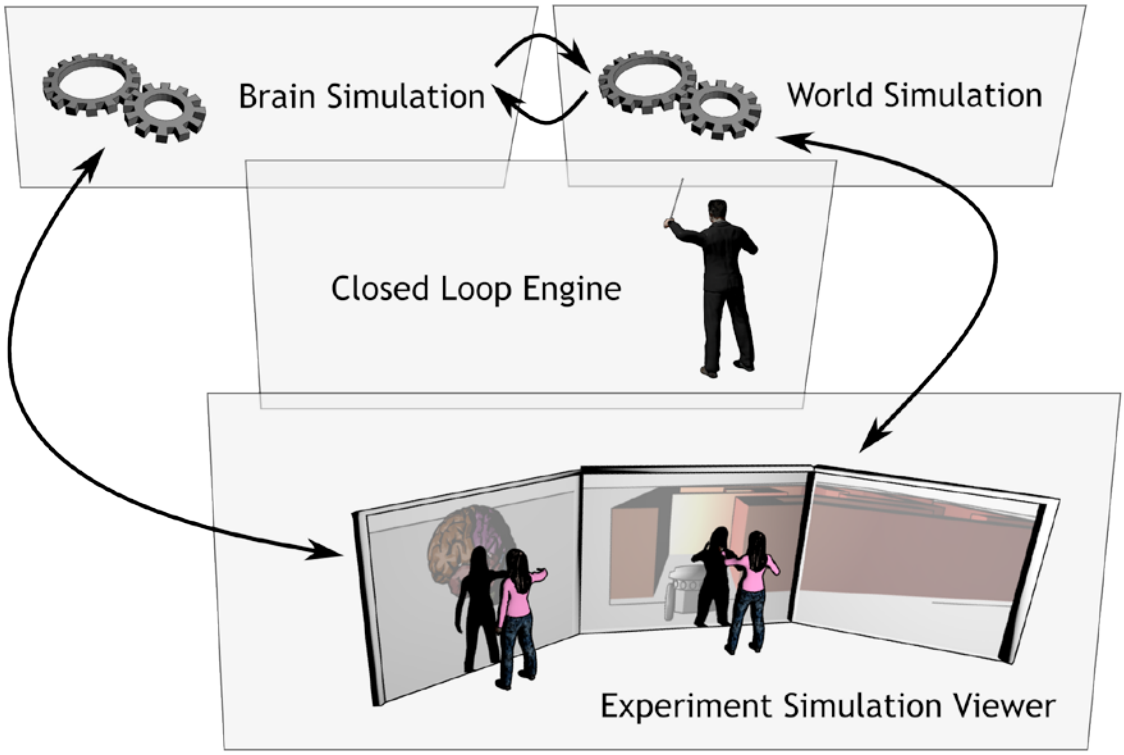

Figure 17 Closed loop engine role as orchestrator

As illustrated in Figure 17, the CLE mainly orchestrates the execution of experiments. It has to control the brain simulation as well as the world simulation. It exchanges information between them to coordinate the data flow. In order to keep the simulations synchronised the CLE maintains the global time.

The visualisation is not done within the CLE, but the CLE provides the respective components with the necessary information where the computed data can be gathered. Furthermore it exposes control functionality to let external components supervise the run or simulation of an experiment.

8.6 Relations to other Platforms

Having a real-time interaction with a simulation is a task identified as part of SP7's High Performance Computing Platform. The HPC team will deliver a layer that allows real time communication for the purpose of having a real time visualisation (in a first step) and then real time interaction capabilities (in a second step). WP7.2 (Mathematical methods, programming models, and tools) provides the building blocks of an interactive, real time brain simulation. WP7.3 (Interactive visualisation, analysis and control) uses these building blocks to provide an interactive visualisation of a brain simulation.

The NRP makes use of this interactive visualisation in the Experiment Simulation Viewer (see Section 9). The Closed Loop Engine uses the same building blocks to allow the communication between the brain simulation and the world simulation and vice-versa.

9. Experiment Simulation Viewer

9.1 Overall Goals

As opposite to the designer components, the Experiment Simulation Viewer (ESV) is the central component for the user in the execution phase. It should provide a convenient view and facilitate a coherent user experience for the simulated experiment. The graphical output of the simulation of the environment wherein a Neurobot (re)acts as well as the visualisation of its simulated brain should be displayed to the user simultaneously. Whilst it is always possible to change the viewpoints of the simulated parts (brain and environment), the availability of user interactions depend on the type of simulation scenario:

Offline Scenario: In this scenario, there is no user interaction. The course of action is determined *a priori* within the experiment set-up. If the user has set up the experiment they can queue it as a job to be run on a high performance computer. After the computation is done, the user gets informed and can receive the calculated data. These data serve as an input to the visualisation. Within the simulation visualisation, the user can control its speed or pause it. Furthermore, they can adapt the viewing point by rotating or zooming. Since the calculated simulation data are stored, it is also possible to replay the experiment run as often as desired.

Online Scenario: An online Simulation requires the simulation to be run in soft real-time. The course of events cannot be computed in advance, since user interactions can obviously not be foreseen. Nonetheless, the same requirements hold regarding the visualisation: the point of view (of the brain as well as in the environment) can be changed in terms of rotating or zooming. This scenario will be available depending on the type of hardware the platform sits on.

The first step to simulate an experiment is to select the respective experiment set-up. This may be either a local data, or data stored and shared by others on the Unifying Portal from SP6. In addition to selecting and loading, the user will specify the initial simulation properties such as the fidelity of physical rendering or constraints on the simulation speed.

Regarding the visualisation, the user should be able to navigate (rotate and zoom the viewing point) within both the brain on one side and the environment on the other side, at any time. The user may focus on a specific connectome and observe its activity throughout the whole experiment. Depending on the hardware, the world and robot simulation will be available in high (i.e. parallel rendering) or low fidelity.

In order to allow a detailed visualisation control of the computed experiment simulation the user must be able to pause, stop, restart or even single-step the (re)play of a computed simulation. Furthermore they should be able to adjust the speed e.g. for replaying a scene in slow motion.

After the ramp-up phase, more complex scenarios should be supported, such as throwing a ball at the Neurobot or pushing it on its body. Likewise, interaction with the environment

should be possible, such as lighting up a lamp, projecting an image on a wall or moving an artefact.

Eventually—possibly after the ramp-up phase—the user should be able to interactively request an experiment snapshot (brain + world) or brain-only or world-only snapshot. This feature has been introduced in the Experiment Designer as programmable within the experiment set-up. However, in interactive (online) simulations, the user might wish to request a snapshot of the brain at any time based on unforeseen brain state observations or after a successful learning phase. Likewise, they may want to request a snapshot of the whole experiment to start it again from this point at a later time.

9.2 Use Cases

The use cases below describe success scenarios for a Neuroscientific User (NSU) in high-level interactions with the NRP. If not stated otherwise the Primary Actor of the use cases is a Neuroscientific User named Abigail. Furthermore it is assumed as a precondition that the ESV is already running.

9.2.1 Loading an experiment set-up (SP10NRP-UC-45)

Precondition:

- Abigail has defined her own experiment set-up with the Experiment Designer OR,
- There are experiment set-ups stored in the UP and accessible to Abigail and Abigail is logged in.

Success Scenario:

1) Abigail opens the selection dialog for choosing an experiment set-up to run.
   a) She picks the experiment definition she just created with the Experiment Designer OR,
   b) She browses the UP until she finds the experiment set-up she wants to run
2) After selecting the experiment set-up and confirming the selection the ESV informs Abigail that loading was successful.

9.2.2 Configuring and starting the simulation (SP10NRP-UC-46)

Preconditions:

- Abigail has loaded an experiment set-up

Success Scenario:

1) Abigail starts the simulation by clicking on a play-like button
2) A configuration wizard pops up.
3) Depending on the hardware the platform sits on, the rendering options described in the Closed Loop Engine part are selectable (see Figure 15):
4) Offline (non interactive) with low-fidelity
5) Offline with high fidelity.
6) Online (interactive) with low-fidelity.

7) Online with high fidelity.
8) Abigail can choose the level of fidelity of each WSE sub-module (e.g. light simulation, physics simulation).
9) For each simulation module, the user can select from a mutual exclusive list of simulators suitable for this category (e.g. HPC based light simulation, GPU rendering).
10) The simulation starts and the robot and brain become active on the views.

9.2.3 Time interaction (SP10NRP-UC-47)

Preconditions:

- The simulation has been started (9.2.2 Configuring and starting the simulation).

Success scenario:

1) Abigail pauses the simulation.
2) Abigail requests the next simulation step.
3) Abigail requests the next simulation step.
4) Abigail resumes to normal execution.
5) Abigail pauses the simulation.
6) Abigail restarts the whole simulation.

9.2.4 Brain interaction

In these use cases, the common precondition is that the simulation has been started (9.2.2 Configuring and starting the simulation) in online (interactive) mode.

9.2.4.1 Reading neurons (SP10NRP-UC-48)

Success scenario:

1) Abigail navigates into the brain and observes neurons or groups of neurons.
2) Abigail gets current and spikes measurements from these neurons.
3) Abigail attaches a recorder on these neurons and starts it.

9.2.4.2 Exciting neurons (SP10NRP-UC-49)

Success scenario:

1) Abigail selects a neuron by navigation.
2) Abigail attaches a virtual electrode to it.
3) Abigail excites the neuron by setting the voltage on the electrode.
4) Abigail observes the neuron's reaction (reading neurons).
5) Abigail removes the virtual electrode.
6) Abigail selects a group of neurons.
7) Abigail inhibits this group.
8) Abigail observes the brain reaction (reading neurons).

9.2.4.3 Snapshotting the brain (SP10NRP-UC-50)

Success scenario:

1) Abigail observes an interesting reaction of a brain region.
2) Abigail pauses the simulation.
3) Abigail selects the brain region in the brain view.
4) Abigail opens up the brain snapshot tool.
5) Abigail takes a snapshot of the given brain region (she defined the snapshot parameters in 5.3.6 - Snapshot region).
6) Abigail saves it to the project locally or on the Unifying Portal.
7) Abigail resumes simulation.

9.2.5 Robot interaction

In these use cases, the common precondition is that the simulation has been started (9.2.2 Configuring and starting the simulation) in online (interactive) mode.

9.2.5.1 Throwing artefacts at robot (SP10NRP-UC-51)

Success scenario:

1) Abigail opens an artefact library (balls, water…)
2) Abigail selects a ball from the library.
3) Abigail sets impact speed and angle (optional).
4) Abigail clicks on robot to define the impact location.
5) The ball is thrown at the robot.
6) Abigail observes the reaction of the brain and of the robot.

9.2.5.2 Pushing the robot (SP10NRP-UC-52)

Success scenario:

1) Abigail opens up a list of actions.
2) Abigail chooses to push.
3) Abigail sets action force and direction.
4) Abigail clicks on robot to define action location.
5) The action is performed on the robot.
6) Abigail observes the reaction of the brain and of the robot.

9.2.6 Environment interaction

9.2.6.1 Moving objects (SP10NRP-UC-53)

Precondition:

- The simulation has been started (6.8.2.1) in online (interactive) mode.

Success scenario:

1) Abigail opens up a list of possible actions.

2) Abigail chooses to turn off one lamp.
3) Abigail validates and observes the system's reaction.
4) Abigail opens up again the actions list.
5) Abigail chooses to move the trash bin (part of the environment).
6) Abigail sets the speed of the movement.
7) Abigail sets the new position of the trash bin by clicking in the scene.
8) The trash bin starts to move to its new position at the given speed.
9) Abigail observes the reaction of the robot.

9.2.6.2 Snapshotting the world (SP10NRP-UC-54)

Success scenario:

1) Abigail wants to use the state of the world as an initial condition for another experiment.
2) Abigail pauses the simulation.
3) Abigail opens up the world snapshot tool.
4) Abigail takes a snapshot of the world.
5) Abigail saves it.
6) Abigail resumes simulation.

9.2.7 Record/replay

9.2.7.1 Online simulation (SP10NRP-UC-55)

Success scenario:

1) Abigail presses the record button.
2) Abigail starts the simulation (9.2.2 Configuring and starting the simulation).
3) Abigail interacts with the simulation (9.2.4 Brain interaction).
4) Abigail stops the simulation (9.2.3 Time interaction).
5) Abigail presses the replay button.
6) A replay viewer pops up and shows the recorded simulation.
7) Abigail sets playing speed.

9.2.7.2 Offline analysis (SP10NRP-UC-56)

Success scenario:

1) Abigail presses the record button.
2) Abigail starts the simulation (9.2.2 Configuring and starting the simulation).
3) Abigail waits 2 days for the simulation to finish.
4) Abigail stops the simulation (9.2.3 Time interaction).
5) Abigail presses the replay button.

6) A replay viewer pops up and shows the recorded simulation.
7) Abigail sets playing speed to make it look real-time.

9.3 Functional Requirements

9.3.1 Control the experiment in simulation (SP10NRP-FR-39)

The user must be able to:

- Start and stop an experiment within the GUI.
- Step the simulation within the GUI.
- Modify the experiment set-up while the simulation is already running.

9.3.2 Brain (SP10NRP-FR-40)

- The user can display the 3D brain model during a simulation.
- The user can navigate inside the 3D brain model during a simulation.
- The user can provide external excitation or inhibition on a group of neuron during the simulation.
- The user can attach virtual recording devices to group of neurons and see the measurement of these recording devices during the simulation (most probably spikes and currents).
- The data recorded by the device can be exported after an experiment.
- The user can take a snapshot of the brain state (or brain region) (see Experiment Designer) for later reuse as an initial brain state.

9.3.3 Robot / environment (SP10NRP-FR-41)

- The user can see the robot evolve in the environment.
- The user can use the mouse to interact with the robot.
- The user can use the mouse and menus to interact with the environment.
- The user can navigate within the environment (change camera position or add cameras).
- The user can take a snapshot of the world (robot + environment) state (see Experiment Designer) for later reuse as an initial world state.

9.4 Non-Functional Requirements

9.4.1 Hardware platform for simulation

- The models will run on three types of hardware platforms: HPC in Jülich, neuromorphic hardware and single desktop computers (or small cluster of them).
- The viewer should be able to run on some parallel rendering clusters (such as the CAVE or the display Wall in Lausanne).

- The brain interfaces used during simulation (brain I/O -> transfer functions -> sensors & actuators) must be able to run on another hardware platform as the simulation engine.

9.5 Architecture Overview

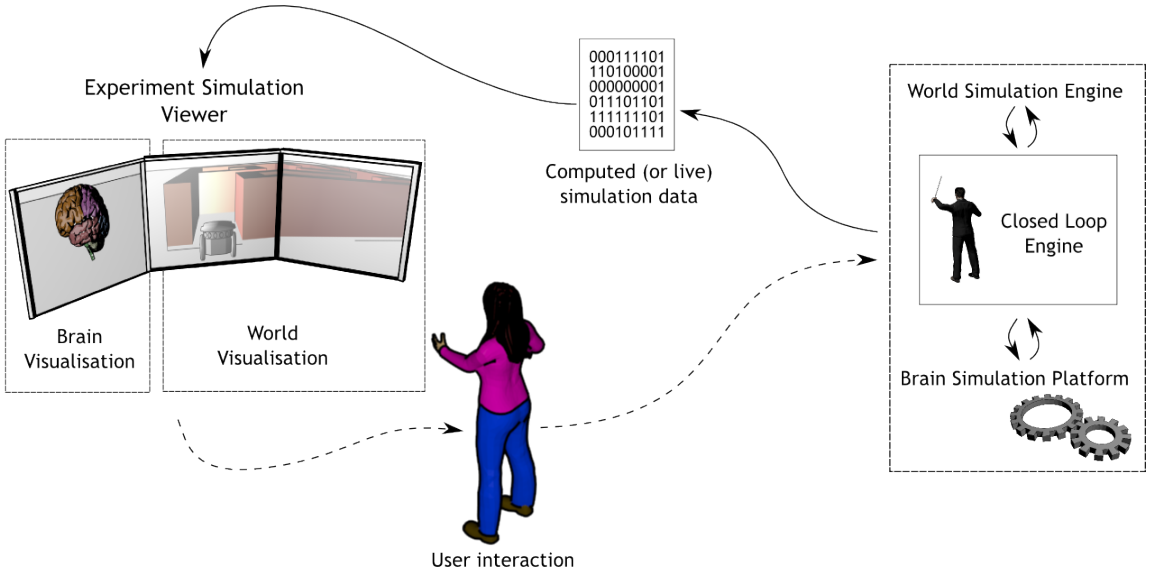

Figure 17 Architecture overview of the Experiment Simulation Viewer

The ESV consists of at least three logical separable components, namely two for visualising the World and Brain Simulation respectively and one for controlling the simulation.

In order to have a coherent simulation both simulations (world and brain) are mediated by a third component, the CLE. The ESV is therefore highly decoupled from the actual simulation and its main task is to gather the information and display it in a coherent manner, i.e. it is responsible for dealing with network issues such as latencies as well as synchronising the single simulation visualisations. As described in the CLE part, the Experiment Viewer may be remote (in a simplified form), but both simulations always run on the same hardware.

World Visualisation. Inquired and gathers the output of the WSE which renders the environment as well as the Neurobot (re)acting in it.

Brain Visualisation. This component would make use of the Brain visualisation component provided by SP7 - T 7.3.4 Integrative visualisation and analysis tools for the HBP cockpits

Visualisation Synchroniser. Since the World and Brain Simulation are two different engines and can have different execution times, they must synchronised before they actually get displayed to the user.

Simulation Control. This component forwards user interactions regarding simulation speed, as well as further simulation configuration parameters (e.g. fidelity), to the responsible component.

9.6 Relations to other Platforms

The brain visualisation component used in the Experiment Simulation Viewer (see Figure 7) is provided by SP7 - Task 7.3.4 (Integrative visualisation and analysis tools for the HBP cockpits). Using this component provides many advantages among which:

- The same component will be used in other platforms (especially in SP6) for online (interactive) or offline experiments. Therefore, the user experience will be consistent among the platforms regarding simulation visualisation.
- This component is designed to run on high fidelity and parallel rendering installations such as the CAVE in Munich. It is also able to run in low fidelity on a state-of-the-art desktop computer.

The Experiment Simulation Viewer needs to be consistent with the Unifying Portal from SP6 as well as with the other components of the NRP cockpit.

10. Functions

The following sections summarise the methodology and the metrics used to control the progresses of the development. They also outline the collaboration that will happen between the stakeholders of the platform, the users, the selected partner, and the science and technology office.

10.1 Software engineering methodology

One of the goals of the Human Brain Project's ICT platforms is "to demonstrate how the platforms could be used to produce immediately valuable outputs for neuroscience, medicine and computing". This can only be achieved by means of a very closely tied mutual collaboration with future users; an approach that challenges classical software engineering process models. We will therefore use a more agile approach – namely the Scrum methodology. This way, we can mitigate risks early, ensure implementation of the real requirements and let users participate as key partners in the development process.

It would be very difficult to achieve this goal – namely to demonstrate immediately valuable outputs of the NRP – if a beta release is delivered at the very end of the first year and a final version at the very end of the second year. The users' requirements may have changed within the two years of development or other external partners will show interest in the platform in the middle of the ramp-up phase. They shouldn't be blocked by the fact that they did not take part in the initial requirements definition.

We address this risk by establishing future users as key partners in the development process as early as possible. Each and every user should be able to provide the software team with feedback as often as possible, rather than being a customer who has to wait for finished product at the end of the ramp up phrase. The chosen methodology that will allow this change is Scrum.

Scrum is a well-established iterative and incremental development framework that already was (and still is) used and gained successful results within other HBP software teams.

10.2 Scrum: Roles and Procedures

In Scrum a shippable increment of the NRP is produced at the end of each iteration (also known as sprint). The iteration or sprint is the basic unit of development in Scrum. The duration for each sprint is fixed in advance, normally between one and four weeks (typically two weeks). There is a planning meeting for identifying and estimating tasks at the beginning and a review meeting for progress monitoring and future prospects at the end of each sprint.

For the NRP, it means that at the end of each iteration, there will be a version available for the users to use or test. Hopefully, research groups and academicals partners that don't need all the features of the platform will also be able to use the platform before the end of the ramp-up phase to produce valuable outputs for neuroscience, medicine and computing.

There are three roles in Scrum:

1) The Product Owner represents the users as well as the Stakeholders.
2) The Development Team is in charge of delivering the platform in terms of potentially shippable increments.
3) The Scrum Master ensures that the team can do its work without impediments and oversees the development process.

At the beginning of the development in the first weeks of April, the Product Owner, with the help of the team, will transform this specification into small increments called User Stories and containers of stories called Epics. These will be collected in the product backlog, which can be considered an ordered list of requirements.

The Product Owner and the Stakeholders will prioritise each story. Regarding the NRP open call configuration, a member of the open call winner organisation and a member of the HBP neurorobotics team will likely take on the Product Owner role. The NRP leaders will be part of the Stakeholders Committee.

At the beginning of every iteration, the team will look at the stories with the highest priority and will decide which of them can be implemented in the iteration. At the end of every iteration, the team will demonstrate that the stories are perfectly implemented (tested and documented) to the Product Owner and the users.

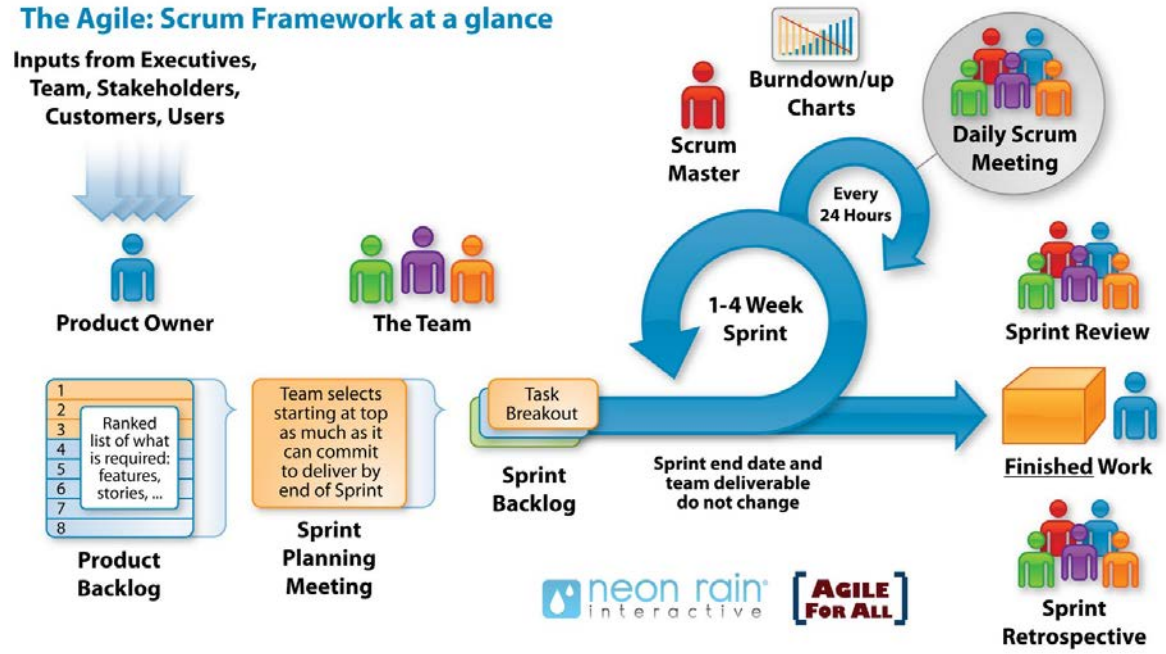

Figure 18 Agile scrum iteration workflow: from the product owner definition of the backlog to a finished product.

The Scrum process model outlined above therefore enables high adaptability to the users' needs and is inherently able to deal with changes in users requirements: Let one user realise – for example – that a new feature will be required which was not thought of in the beginning. By giving feedback the Product Owner can write new stories and will (re-) prioritise them with all the Stakeholders. The key point is that whatever their feedback is, the Product Owner can take quick concrete actions in order to satisfy the users' needs.

10.3 Project management and monitoring

10.3.1 Scrum review

Small iterations methodologies allow for more rapid feedback from the users than methodologies with a long release cycle. At the end of each iteration, a working platform will be released along with a description of the features that have been implemented during the iterations. It will allow the users as well as the stakeholders to see the NRP capabilities at any time during the ramp-up phase.

Gathering feedback from users on small iterations is a difficult but rewarding task. In a first step, before the platform reaches the Minimum Viable Product (MVP) state, internal users will have to test a subset of features. These users will be people involved in the HBP Consortium. The MVP state is scheduled for the beta release milestone in the exact middle of the ramp-up phase. Once it's reached, more users will test the platform. During development, the stakeholders and the team will make sure that enough advertisement of the platform is done to attract beta users.

The feedback of the users will be addressed at every iteration when the backlog is discussed among the stakeholders, the product owner and the science and technology officers (see next section). Integrating this feedback is vital for the platform. It will reduce

the risk having the final delivery not corresponding to the users expectations (see Figure 19).

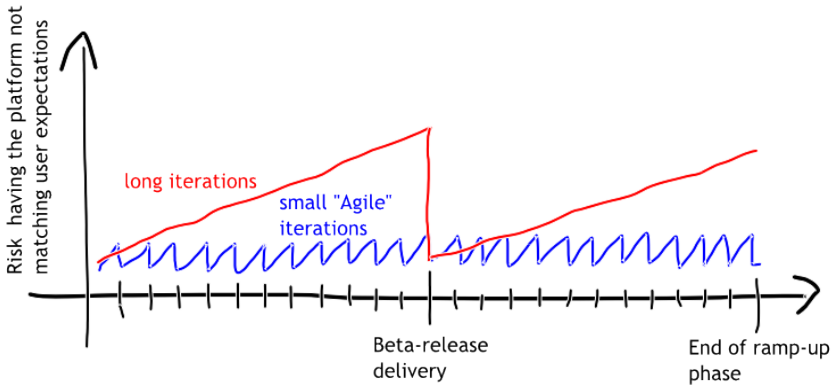

Figure 19 Risk reduction using a short feedback loop with users.

For every iteration, the development team will present the new features developed in a demo-oriented presentation (see Figure 18). Stakeholders, science and technology officers and users will be invited to this presentation.

10.3.2 Backlog

In the Scrum methodology, the backlog is the container of all user stories (see Figure 21). Every user story has an effort estimate (done by the development team) and a priority. The NRP will separate the backlogs for each work package (currently: WP1 – Robot Designer, WP2 – Virtual Environment Designer, and WP3 – Closed Loop Engine). Each one of them will be again separated into three parts: the request backlog, the release backlog and the current iteration backlog (see Figure 20). There will be only one request backlog per work package. There will be releases backlogs for each planned official release. For the time being, two releases are planned, one at the middle of the ramp-up phase and one at the end. The iteration backlog will be redefined at every iteration.

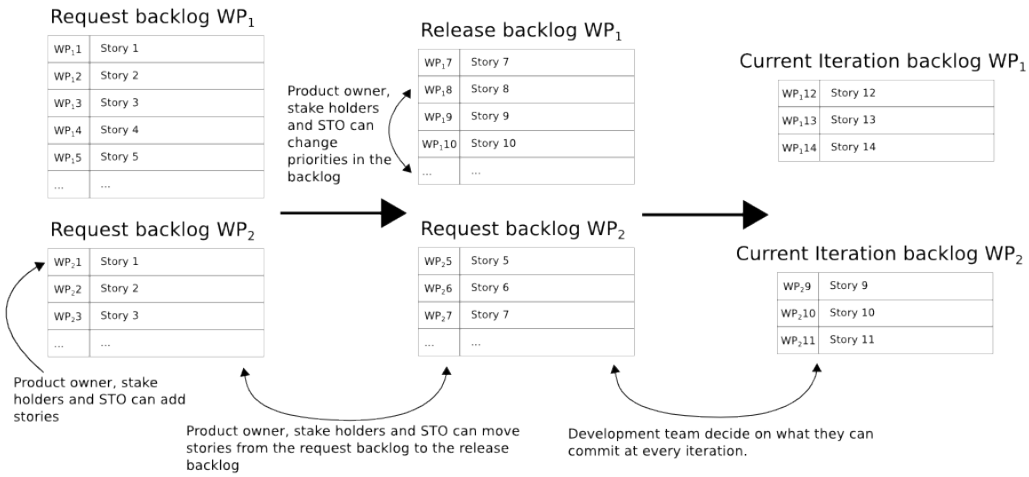

Figure 20 Interactions between the request backlogs, the release backlogs and the iteration backlogs.

The release and request backlogs will be available to the stakeholders and the science and technology officers at any moment. They will be discussed before each iteration to figure out if modifications have to be made. There are several possible types of modifications:

- A change of priority in the release backlog: For example, one user needs a specific feature in order to make a demonstration of the software during a conference. The stakeholders along with the science and technology officers can choose to re-prioritise the user stories related to the feature in order to match the conference date.
- An addition of user stories: For example, one user needs a feature for his or her project that is not contained in the release backlog. The story is defined in the request backlog. This user can ask the stakeholders and the science and technology officer to move the story in the release backlog and to prioritise it.
- A removal of user stories: After having added needed user stories to the backlog, the stakeholders and the science and technology officers realise that the software will not be delivered on time. They can decide to remove the less crucial stories of the release backlog in order to make sure that the software is delivered on time. These stories will be put back in the request backlog. They can also be dropped if they are no longer valuable for the platform.
- A change in the estimation of the stories. The development team will regularly re-estimate the stories in the backlog. It's very difficult to have exact estimations at the beginning of a project. Doing re-estimation regularly will improve the overall confidence for meeting the deadlines.

Figure 21 shows an example of a backlog. The upper part shows the current iteration (sprint number 5) of user stories. The lower part shows the rest of the backlog. The order in this list is important: it reflects the priorities decided by the stakeholders.

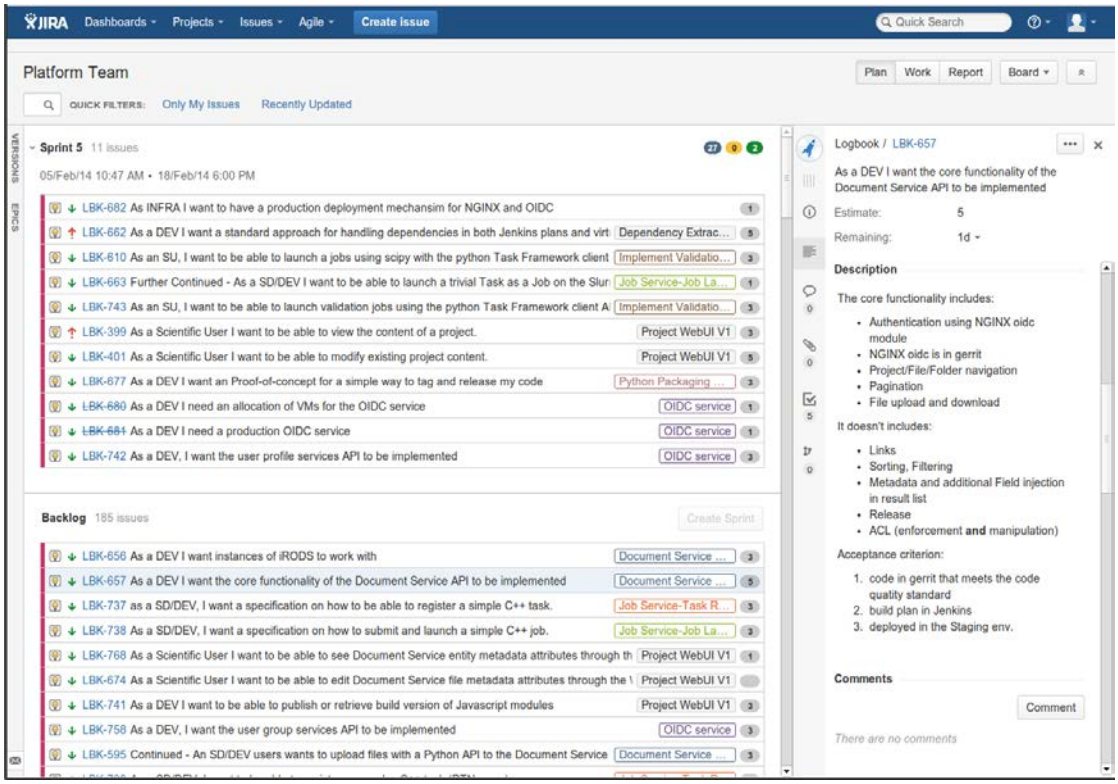

Figure 21 View of the backlog of the portal team.

10.3.3 Development progress

The NRP will use burndown charts to present development progress visually. A burndown chart is a graphical representation of the amount of work that has to be done and the amount of work that has already be done, versus time. The amount of work is measured in the backlog: it's the sum of the estimations of every user story.

A burndown chart describes:

- Progress of development in terms of number of implemented user stories, as well as the proportion of implemented and unimplemented user stories.
- Changes in requirements and estimations. The part below the x-axis represents the workload that has been added after the first iteration.
- An approximation of the final release date, based on what has been done and what has been added to the requirements.

One burndown chart will be provided for each WP. Charts will be updated at the end of each iteration, and made available to the stakeholders and the science and technology officers. The charts will help them when they will have to discuss the addition or the removal of user stories from the backlogs.

During the first months of the ramp-up phase, the NRP team will create the initial backlog. The backlog will consist of all the user stories needed to achieve the goals specified in chapters 2 through 9. Once this step has been achieved, NRP progress monitoring will start.

Figure 22 shows an example of a burndown chart. The goal of this metric is to have the release date always matching the end of the ramp-up phase (M30). More precisely, the worst-case scenario (where the red line crosses the blue line) needs to happen before the end of ramp-up phase.

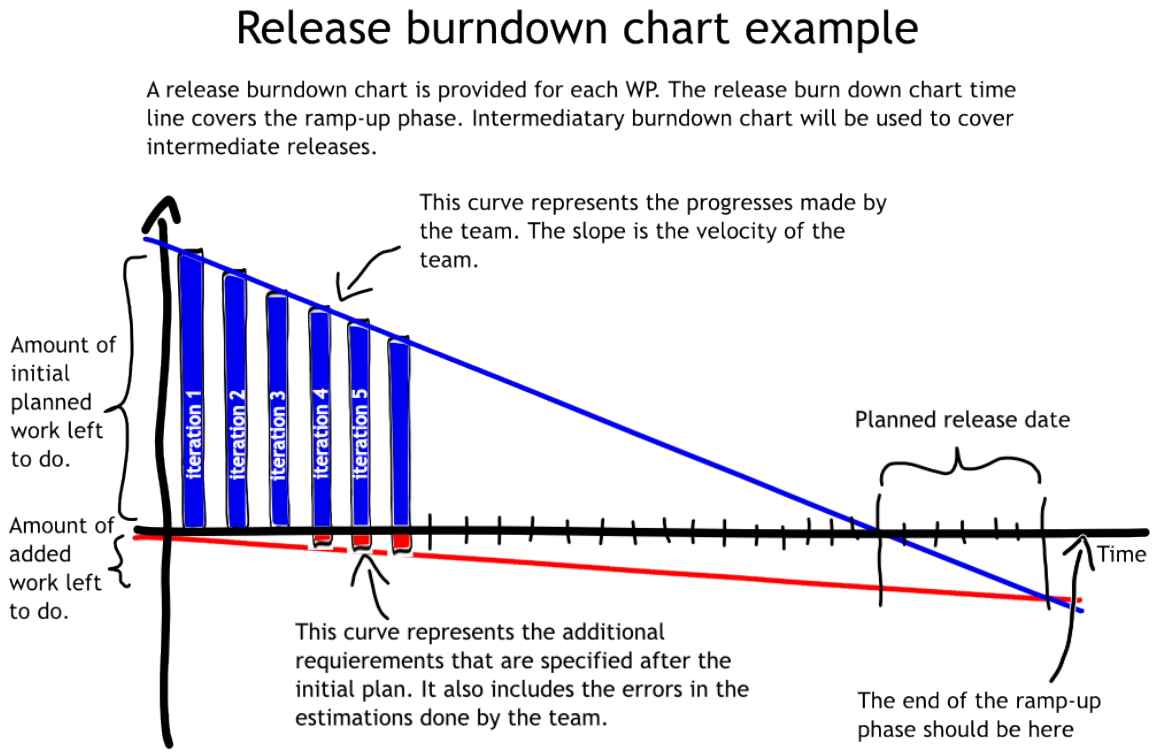

Figure 22 Burndown chart example.

10.3.4 Quality

The software build will take place on a daily basis using Continuous Integration (CI). During CI, code quality will be checked at three different stages:

NAME	DESCRIPTION	TARGET VALUE
STATIC ANALYSIS	Number of static analysis warnings	0
COMPILER WARNINGS	Number of compiler warnings	0
UNIT TESTING	Code coverage through unit tests	90%

The CI will report all these values every time code is added or modified. Developers will have immediate feedback and can apply corrective measures if needed.

10.3.5 User satisfaction

We will have workshops with the users on a regular basis (WP10.4). Those will provide informal feedback allowing direct contact between users and developers. We will also conduct simple user satisfaction surveys in order to have concrete measures.

NAME	DESCRIPTION	TARGET VALUE
USER SATISFACTION	Score (%) based on small standard questionnaire filled by users (number of users not relevant)	The more, the better
PLATFORM ACTIVITY	Number of projects, number of registered users	The more, the better

10.4 Stakeholders & science and technology officers protocol

On a regular basis, at least at the same rate as the Scrum iterations, the stakeholders and the science and technology officers will meet with the Product Owner. The agenda of their meeting will be:

1) Development status: Is the team facing issues in the current iteration? Are there any changes forecast in the estimations of the next stories?
2) Interactions with the users: Are there changes to make on the backlogs to satisfy them? (Priority, addition or removal of user stories).

The outcome of this regular meeting will be an up-to-date backlog reflecting the current priorities.