Project Number:	284941	Project Title:	Human Brain Project

Document Title:	HBP Brain Simulation Platform v1 – Specification document (report)
Document Filename[(1)]:	SP6 D6.7.1 RESUBMISSION FINAL.docx
Deliverable Number:	D6.7.1
Deliverable Type:	Platform Specification
Work Package(s):	WP 6.1, WP 6.2, WP 6.3, WP 6.4, WP 6.5, WP 6.6, WP 6.7
Dissemination Level:	PU
Planned Delivery Date:	M 6 / 31 March 2014
Actual Delivery Date:	M 9 / RESUBMITTED 03 August 2015

Authors:	Henry MARKRAM, EPFL (P1), Jeanette HELLGREN-KOTALESKI, KTH (P33)
Compiling Editors:	EPFL (P1): Jeffrey MULLER, Eilif MULLER
Contributors:	EPFL (P1): Selim ARSEVER, Jean-Denis COURCOL, Fabien DELALONDRE, Mike GEVAERT, Marc-Oliver GEWALTIG, Dan KELLER, Daniel PEPPICELLI, Srikanth RAMASWAMY, Michael REIMANN, Luis RIQUALME, Felix SCHÜRMANN, Werner VAN GEIT, Stefano ZANINETTA JUELICH (P17): Markus DIESMANN, Abigail MORRISON, Moritz HELIAS, Jochen Martin EPPLER, Susanne KUNKEL KI (P31): Sten GRILLNER KTH (P33): Omar GUTIERREZ-ARENAS OIST (P38): Eric DE SCHUTTER UNIPV (P61): Egidio D'ANGELO NMBU (P67): Hans Ekkehard PLESSER UCL (P70): Alex THOMSON YALE (P80): Michael HINES
Reviewers:	EPFL (P1): Richard WALKER, Guy WILLIS, Celia LUTERBACHER

Abstract:	The Brain Simulation Platform (BSP) is one of the six ICT Platforms in the Human Brain Project (HBP), which will be made accessible over the Internet, via an HBP Collaboratory (HBP-COLL or COLL). The HBP's Subproject 6 is developing both the COLL and the BSP. This document sets out the specifications for the COLL and the BSP, including the Initial Brain Models that will run on the BSP. It includes Key Performance Indicators (KPIs) that can be used to track the progress in the COLL and BSP development.
Keywords:	Brain Simulation Platform, Collaboratory, specification, brain builder, molecular simulator, cellular simulator, network simulator, initial brain model

Document Status

Version	Date	Status	Comments
0.16	10 Apr 2014	draft	
0.20	5 May 2104	draft	
0.3	13 May 2014	draft	Additions from E & J Muller, review by Writing Office
2	16 May 2014	draft	Copyedit by CL
3	21 May 2014	draft	Various figures enlarged
4-7	2 June 2014	Draft	Final edits from the compiling editors and the reviewers
8	3 June 2014	draft	Final review, editorial office
9	26 June 2015	Draft	Preparation for resubmission based on review feedback. Merged large blocks of changes from J-D COURCOL, M. GEVAERT, and L. RIQUALME
10	23 July 2015	Draft	Replaced the Unified Portal legacy terminology with the current Collaboratory nomenclature.

Table of Contents

List of Figures and Tables

Executive Summary

The Brain Simulation Platform (BSP) is one of the six information and communications technology (ICT) Platforms that will be developed by the Human Brain Project (HBP). The BSP and other HBP Platforms will be made accessible to scientific, medical and engineering researchers over the internet via an HBP Collaboratory (HBP-COLL or COLL). The HBP's Subproject 6 is developing both the COLL and the BSP.

The purpose of this document is to set out the specifications for the COLL and the BSP, including the Initial Brain Models that will run on the BSP.

The COLL will allow seamless interaction with the ICT Platforms and other HBP online resources, while maintaining sufficient simplicity to encourage use by less technically adept Users. All of the tools used throughout the COLL will track data provenance and software dependencies to provide pragmatic solutions for reproducibile neuroscience. By facilitating the sharing of expertise, data and results, the COLL will generate network effects throughout the HBP community and help to maximise the research impact of the entire project. A key tool within the COLL of particular relevance for the BSP is the Brain Atlas Embedding Module (BAEM).

The BSP itself is made up of a number of high-level components:

- A Brain Builder – a software application to establish a data set and activate a series of data-driven algorithms and workflows to reconstruct multi-level brain models at different levels of fidelity.
- A Molecular Simulator – a tool to simulate brain processes at the molecular level.
- A Cellular Simulator – a tool to simulate morphologically detailed neuron models.
- A Network Simulator – a tool to simulate large numbers of simplified neuron models.

The BSP will be used to develop and validate a set of Initial Brain Models:

- Molecular-level models of neurons, glia and synapses
- A somatosensory cortex model
- A cerebellum model
- A hippocampus CA1 model
- A basal Ganglia model.

This document includes Key Performance Indicators (KPIs) that can be used to track progress in the COLL and BSP development, model building, and adoption by the scientific community. The COLL and the BSP components will be released to research communities outside the HBP in project Month 30 to integrate the Partnering Projects envisaged in the Operational Phase, which will be conducted under the aegis of the EU's Horizon 2020 Programme.

1. Introduction

The Human Brain Project (HBP) is a ten-year research project, funded by the European Commission (EC), to lay the foundations for a new approach to brain research. Neuroscience, medicine and information technology each have important roles to play in addressing this challenge, but their contributions are currently fragmented. The HBP will integrate these inputs and catalyse a community effort to achieve a new understanding of the brain, new treatments for brain disease and new brain-like computing technologies.

It is a central tenet of the HBP strategy that a comprehensive understanding of the brain requires knowledge of structure and function across all levels of brain organisation; this understanding cannot be achieved at any one level alone. The only approach available today that allows simultaneous study of the brain across all levels is brain simulation using structurally and functionally accurate digital computer models. Since such an effort involves expertise from neuroscience, computer science, physics and mathematics, a massive scientific collaboration is required to reconstruct such multi-level models. The social internet and open source software communities have shown that modern information and communications technology (ICT) permits the massive collaborative efforts needed.

Brain simulation within the HBP is the responsibility of Subproject (SP) 6, which is developing the HBP's Brain Simulation Platform (BSP). This is one of six ICT Platforms being developed by the HBP; the others focus on Neuroinformatics, High-Performance Computing, Medical Informatics, Neuromorphic Computing and Neurorobotics.

To facilitate the scientific community's access to the HBP's ICT Platforms and, in a broader sense, to make large-scale collaborations possible in neuroscience, SP6 is also developing the HBP Collaboratory (HBP-COLL or simply COLL). This web-based collaborative scientific platform will provide access to the HBP's research, community and administrative activities, as well as its six ICT Platforms. A key tool within the COLL with particular relevance for the BSP is the Brain Atlas Embedding Module (BAEM), which will provide a deeply integrated search of the Neuroinformatics Platform for modelling and validating data, and a viewer for rich 2D and 3D data.

The COLL will be equipped with a layer of powerful social networking functions to allow fluid sharing of data, theories, applications and models prior to publication, while still maintaining proper attribution. This sharing of research, results and expertise should help to accelerate neuroscience and the achievement of the HBP's ambitious goals.

This document sets out the specifications for the COLL and the BSP Version 1.0 (BSP 1.0). It is intended for a technical and scientific readership. The COLL section focuses on the functionality that the COLL will provide. It also describes the ways in which web-based platform components will interact with the system. The BSP section includes specifications for the components, simulators and models that are delivered in the COLL. The COLL, BSP and BSP components (except for the Initial Brain Models), are described using a standard sequence of section headings: Overall Goals, Use Cases, Functional Requirements, Non-Functional Requirements, Architectural Overview, Relations to other Platforms, and Dependencies.

This document includes Key Performance Indicators (KPIs) that can be used to track the progress in the COLL and BSP development, model building, and adoption by the scientific community. The COLL and the BSP components will be released to research communities outside the HBP in project Month 30 to integrate Partnering Projects envisaged in the

Operational Phase, which will be conducted under the aegis of the EU's Horizon 2020 Programme.

2. The HBP Collaboratory (HBP-COLL)

This section presents the specification for the HBP Collaboratory. It starts with the requirements that informed the design process and the relationship to other Platforms, before dealing with the architecture of the COLL and its components.

2.1 HBP-COLL: Overall Goals

The COLL is a web-based portal intended to provide a single point of access to collaborators participating in all research activity in HBP. The COLL will allow scientists from around world to collaboratively:

- Gather and organise multi-level neuroscience data (via the Neuroinformatics Platform)
- Reconstruct, validate and refine multi-level brain models at different levels of fidelity (via the Brain Simulation Platform)
- Search, analyse and cluster distributed clinical data (via the Medical Informatics Platform)
- Develop and access interactive supercomputing (via the High Performance Computing Platform)
- Configure, train and operate neuromorphic computing systems (via the Neuromorphic Computing Platform)
- Couple brain models to virtual agents acting in virtual environments to perform *in silico* cognition and behaviour experiments (via the Neurorobotics Platform).

The COLL is designed to catalyse research at all levels of the HBP by allowing a) instantaneous sharing of data, models, tools, theories, configurations, methods and applications, b) tracking and crediting researchers for their contributions (provenance), c) crowdsourcing mining of the literature, and d) launching collaborative projects on any level.

The COLL will also be the primary means by which the HBP shares its scientific and technological advances with the scientific, medical and engineering communities. The COLL will therefore provide a platform for Strategic Partners to present their research projects, form collaborative projects with other members of the Flagship Consortium, and share their progress in advancing or using the HBP ICT platforms.

The COLL will eventually provide platforms for European and international collaborations, depositing and licensing IP, subscribing to HBP services, developing knowledge streams and for the accessing and managing of educational services, for managing the dissemination of HBP material to science museums around the world, providing feedback for responsible research innovation, and for general administration and management of the HBP. The COLL will be designed to scale to very large numbers of researchers in science, medicine and engineering, providing a novel virtual environment for distributed, collaborative and multi-disciplinary research and development.

To achieve this goal the COLL must:

1) Serve both technical power Users and non-technical casual Users,

2) Perform pervasive provenance tracking,

3) Facilitate simple access models for HPC and Neuromorphic computing resources,

4) Facilitate simple access models for rich multi-dimensional data sets.

In most cases the modellers, theoreticians and computer scientists are the ones building tools for inclusion in the COLL. These are the *Power Users*. However, it is often difficult to validate or use those tools without input or validation data. Biologists need computational tools in their toolkit, but those tools must be easy to use. These are the *Casual Users*. It is necessary to serve both *Power Users* and *Casual Users* because the goals of the Human Brain Project can only be met by active collaboration between biologists, theoreticians, modellers and computer scientists.

Neuroscientific data are among the most complex (due to their multidimensionality, ontologies, and formats), and opaque in science. Part of the problem in understanding the human brain is finding and organising the data so that they can be searched and used to build models and other applications. By providing web-based tools for searching, viewing and analysing rich neuroscientific data sets, the HBP will shorten the distance between experiment and discovery and make possible worldwide data-centric collaboration.

2.2 HBP-COLL: Use Cases

The Use Cases below describe success scenarios for small numbers of actors. The scenarios describe high-level interaction with the HBP-COLL and its underlying services.

Each Use Case described in this document is attributed a unique identifier. “SP6COLL” indicates that it relates to the COLL, while “SP6BSP” shows that it concerns the BSP. “UC” indicates a Use Case, while “FR” denotes a Functional Requirement. For more on the software development methodology adopted by SP6, see Section 9.1 below.

2.2.1 Roles

- Computational Scientific User (CSU) – A User with scientific development skills and comfort in launching command line HPC jobs.
- Biological Scientific User (BSU) – A User with scientific expertise, but limited technical computing skills.
- Scientific User (SU) – A scientific User, either a CSU or a BSU.
- Scientific Developer (SCIDEV) – A User who is developing software to directly realise the scientific objectives. This User is usually working in close collaboration with scientists, both CSUs and BSUs.
- Developer (DEV) – A User who is developing software to realise engineering, operational and/or scientific objectives.
- Portal User (PU) –A User who accesses Platform functions through the Web GUI.
- Service User (ServU) – A User who accesses Platform functions through a programmatic Service Client API.
- Infrastructure Personnel (INFRA) - An infrastructure System Administrator or Developer, typically responsible for deploying and monitoring Platform services that are offered directly to customers.

2.2.2 Low Volume Scientific Data Sharing, Single COLL Project with Upload (SP6COLL-UC-001)

Abigail needs help analysing some morphologies so she creates a COLL Project to collaborate with another scientific User in a shared workspace.

Primary Actors: Two Scientific Users, Abigail and Bill.

Success Scenario:

1) Abigail has morphology geometry data on her local machine or on a public and permanent internet-accessible URL.
2) She creates a COLL Project, COLL Project 1, in the Portal to hold the morphologies. Abigail is the owner of COLL Project 1.
3) She uploads the morphology files or their URLs to COLL Project 1. The COLL will request that she add HBPMIN metadata (see Glossary) to the Artefacts on upload.
4) If she decides not to add HBPMIN metadata to any uploaded data, the UI will display those data files differently.
5) Some portions of the HBPMIN data can be extracted from the data, if the data type is known to the COLL.
6) If she attempts to use uploaded data that are not HBPMIN annotated in COLL analysis Tasks, she will be required to add HBPMIN metadata before the analysis will be launched.
7) Abigail can also add a simple wiki entry specific to the uploaded morphology. This wiki entry will help to give context to the uploaded morphology to help Users find it from the COLL search functionality.
8) A background process will reflect COLL Project 1 and all of its metadata into the Neuroinformatics Platform.
9) She adds Bill to the COLL Project team and gives him read and write access to the COLL Project.
10) Bill can now download (COLL Project read) or use Portal Tasks (COLL Project read and write) to analyse and model with the morphologies.

2.2.3 Low Volume Scientific Data Sharing, Multi COLL Project (SP6COLL-UC-002)

Abigail and Chris created a COLL Project in a previous Use Case and now Bill needs additional help to review the output of one of his analysis. However, he doesn't want Chris, whom he asked for help, to see the original COLL Project.

Primary Actors: Three Scientific Users, Abigail and Bill and Chris.

Precondition:

- Abigail has created COLL Project 1 with Bill as described in SP6COLL-UC-001.

Success Scenario:

1) Bill creates a new COLL Project, COLL Project 2.
2) Bill creates a COLL link, denoted L, in COLL Project 2 to an analysis output data file, denoted A, in COLL Project 1.
3) Bill then adds Chris to the COLL Project 2 as a COLL Project Administrator.

4) Bill then performs analysis on L, implicitly specifying COLL Project 2 as the output directory. The output of the analysis on L is denoted A'.
5) Chris (or any other User with read permissions in COLL Project 2) is able to see A'.
6) Users with read permissions in COLL Project 2 are not able to see the contents of A' (i.e.: A) unless they have been added to Abigail's COLL Project as a reader, or unless Abigail has made the data public to one of the sharing groups that Chris is a part of.

2.2.4 Viewing of Data in HBP-COLL (SP6COLL-UC-003)

Abigail is interested in some data that are found through a search interface, in a COLL Project, or that are viewed from a site supporting the COLL authentication data. She wants to view those data using a rich visualisation.

Primary Actor: Abigail a Scientific User.

Success Scenario:

1) Abigail has used the Brain Atlas Embedding Module (BAEM) in search mode to find morphology data.
2) Abigail has the option to view her morphology data using an image rendering, a 3D geometry viewer or a provenance viewer, showing where the data came from.
3) She can add the data to a COLL Project where she can use the same viewer options to view the data.
4) The available viewers will be filtered based on semantic content types and can be used in Web UIs throughout the various platforms.

2.2.5 Data Hiding (SP6COLL-UC-004)

Abigail wants to remove some data that are no longer useful from her COLL Project. Because we want to preserve data provenance, the data are hidden rather than deleted.

Primary Actors: Two Scientific Users, Abigail and Bill.

Success Scenario:

1) As COLL Project owner, Abigail is allowed to make data or folder entities inside her COLL Project hidden.
2) Bill has write privileges but this is insufficient to mark data as hidden. Bill must be granted admin privileges to mark data as hidden.
3) All COLL Project group members can toggle a flag in the UI to view or hide hidden data for their own view.
4) Hidden data will still be optionally visible (thought marked as hidden) to anyone they have been shared with.
5) Hiding data is a separate function from true deletion. Deletion of data is a highly privileged operation that must be done by System Administrators at a User's request.
6) Entity Links to hidden data are still visible and the hidden data are still accessible through the link.
7) Provenance Links to hidden are still visible and the hidden data are still accessible through the link.

2.2.6 Data Release (SP6COLL-UC-005)

Abigail has some data that she wants to share with a larger community.

Primary Actors: Two Scientific Users, Abigail and Bill.

Precondition:

- Abigail has created a COLL Project 2 with Bill as described in SP6COLL-UC-001.

Success Scenario:

1) Abigail is confident that her latest cerebellum model in COLL Project 2 is a significant improvement over previous cerebellum models. She wants to make her model available to others to run simulations on, analyse and refine.
2) Abigail selects the folder in her COLL Project that contains the cerebellum model, named "model" and presses the Release button.
3) She is warned that the Release is an irreversible action and that her COLL Project will become read-only.
4) She selects a name for the new Release, "Cerebellum Release 1".
5) She is prompted to update the permissions of the Released model as a convenience. Abigail upgrades the visibility of the new Release to allow everyone in the HBP to see it. This grants access to all data in her new Release to anyone in the HBP.
6) The "model" folder is moved to the new release entity named Cerebellum Release 1. The original folder location in COLL Project 2 is replaced with a link the model folder in Cerebellum Release 1.
7) Releases can be published to the Knowledge Space if required.

2.2.7 Collaborative Scientific Analysis (SP6COLL-UC-006)

Bill wants some help analysing data. He recruits Abigail to his COLL Project and then shares the results with Chris.

Primary Actors: Three Scientific Users: Abigail with strong software development skills, and Bill and Chris with strong biological skills. Abigail and Bill are working on the same COLL Project. Chris does not share any COLL Projects with Abigail and Bill.

Success Scenario:

1) Bill has morphology geometry data in a COLL Project that he would like to understand quantitatively, but he needs additional expertise to write the analysis software and to analyse the data.
2) He adds Abigail to the COLL Project team and gives her access to the morphologies.
3) Abigail can also develop and integrate new Portal software components called Tasks to perform additional analyses to help answer Bill's questions.
4) Abigail can now also use existing Tasks in the Portal Task Registry to analyse data and answer questions for Bill.
5) Abigail or Bill can execute Tasks, new and previously existing, as a Job.
6) Output of the analysis Tasks is linked both to the Task definition and the Task input data through the Provenance service.

7) If Abigail decides that her new analysis Task is useful to others, she can use the Portal (through UI or service interface) to make their analysis Task public for others to run. TBD: This may or may not require a review by the COLL or HPC Platform team.

8) Chris, who also works with morphologies, can now use Abigail's analysis Task. The Portal tracks Chris' output data and knows that Abigail's analysis was used to generate it.

NOTE: Our target is to streamline integration of analyses into the HBP-COLL. The target for packaging and registration in the HBP-COLL is approximately two hours of scientific software developer time. The target for the registration of small subsequent analysis changes is less than 15 minutes of scientific software developer time. These targets assume that no exotic software dependencies are required by the analysis in question.

2.2.8 Portal Developer Services and Component Reuse (SP6COLL-UC-007)

Catherine wants to extend the data visualisation capabilities of the Portal.

Primary Actor: One Portal Developer, Catherine.

Success Scenario:

1) Portal Developer User Catherine has a Web UI extension to the HBP Portal that she would like to implement.

2) Catherine's application will be divided into HTML5 client-side logic and a REST service implemented in Python or Java.

3) For the browser client side of the application, Catherine will be able to take advantage of any HTML5 libraries she wants, or she can use the Angular widgets produced by the HBP COLL Portal team. She will then create the client side portion of her application. She will integrate the client-side connector library to allow her application to talk the HBP COLL Portal container.

4) For the server side of the application, Catherine will use a standardised authentication library to authenticate Users of her application securely against an HBP Central Authentication Service. The integration with the HBP Authentication Service will allow her application to access COLL REST APIs on behalf of the authenticated User.

2.2.9 Scientific Developer Iterative Workflow Development (SP6COLL-UC-008)

Daniel needs to update a workflow to integrate a new type of data constraint.

Primary Actor: One Scientific Developer, Daniel.

Success Scenario:

1) Scientific Developer User Daniel would like to add the use of synthesised morphologies to the circuit building workflow.

2) Daniel will download the Python orchestration code for the workflow. He will also download the dependency description that will allow him to reproduce the workflow execution if he has access to sufficient computing resources.

3) His new synthesis modification makes sense to implement as a reusable Task module, so he first builds a standalone Task and tests it independently.

4) Finally, he modifies the circuit building workflow to include the synthesis Task in the appropriate location.

5) Once he has tested his workflow and container description locally, Daniel will use the Task registration tool to publish his tool to the HBP Platform Task Repository for everyone to use.

2.2.10 Visualisation Developer Component Reuse (SP6COLL-UC-009)

A CAVE Visualisation Developer, Elisabeth wants to find data to view in her application.

Primary Actor: A Visualisation Developer, Elisabeth.

Success Scenario:

1) Visualisation Developer User Elisabeth would like to find cellular models for visualising inside of her CAVE application.
2) Elizabeth's application uses the COLL Search API to search for cellular models matching certain metadata queries.
3) The application then uses the Document Service API to find a local path from which her application can load data.
4) If the data for the cellular model are not available on a locally accessible storage resource, the application can use the Data Transfer API to move the data from the remote location to a locally accessible storage resource.
5) The application can use standard filesystem APIs to load and visualise the cellular model in the CAVE.
6) If needed, her application can also use the Provenance Service REST client API to find source data from which a particular cellular model was constructed.

2.2.11 Interactive Atlas Exploration (SP6COLL-UC-010)

Bill wants to explore the data that are available for the mouse. His intent is to use this in later analyses, but to do so he must first know what is available. This is done through the BAEM of the COLL.

Primary Actors: Scientific User, Bill.

Preconditions:

- One or more atlases for the mouse have been registered in the Knowledge Space.

Success Scenario:

1) Bill wants to explore available data for the mouse.
2) In the COLL, Bill would select the Mouse atlas, which would open the BAEM in the Atlas viewer mode with an initial context of Mouse.
3) From the initial context Bill would be able to view a large number of data types in the spatial context of the default Mouse, Rat or Human reference space:
    a) 2D image slices
    b) 2D parcellations
    c) 3D volumes
    d) 3D parcellations
    e) 3D objects
    f) Registered data links

4) Data can be added to a COLL Project or will be added to the current COLL Project automatically if an analysis Task is run on the data.

2.2.12 The BAEM as a User Interface Component for Cell Subset Selection (SP6COLL-UC-011)

Bill wants to select a subset of cells for the configuration of a Simulation Task.

Primary Actors: Scientific User, Bill.

Preconditions:

- A detailed cellular-level microcircuit, mesocircuit or macrocircuit.

Success Scenario:

1) Bill wants to select the subset of cells in a circuit that will receive a particular stimulation protocol.
2) The Simulation Configuration Interface has a component that is the Stimulation Configuration Interface. The Stimulation Configuration Interface will have a data-typed input, which will be a cell GID list. The default editor for the cell GID list data type will be the BAEM in cell selection mode.
3) In cell selection mode, the BAEM allows ontology-based filtering of the cell selection. The cell selection is displayed so that Bill can verify by inspection that the filter cell set has the expected properties (location, distribution, etc.).
4) Once the selection is correct, Bill can finalise the selection and proceed to the next step of the Stimulation Configuration Interface.

2.2.13 Interactive Exploration of the Circuit Model (SP6COLL-UC-012)

Bill wants to interactively inspect and explore an existing network or cellular-level model.

Primary Actors: Scientific User, Bill.

Preconditions:

- A network or cellular-level model.

Success Scenario:

1) Bill would like to explore a recently built cellular-level model.
2) One of the views available for the cellular-level model content type in the COLL Project browser is the model viewer mode of the BAEM.
3) Bill opens the model viewer mode of the BAEM that allows ontology-based filtering of the cell selection. Bill can interactively view cell properties and their relations to other cells.
4) Bill can save a view as a session that can be shared with another User.
5) Bill can also choose to export parts of the model for more detailed investigation.

2.2.14 Finding Data through Direct Search (SP6COLL-UC-013)

This Use Case describes the interaction with the BAEM for the pre-selection of data that are later used in the configuration of a Job prior to execution.

Primary Actor: Scientific User Abigail.

Success Scenario:

1) Abigail wants to extract improved dendrite synthesis parameters to feed to a prototype neuron synthesis program.
2) Abigail creates a new project, COLL Project 3, to hold her input data and the synthesis parameters she wants to generate.
3) Her current focus is Layer 5, so Abigail is wants to find reconstructed morphologies from Layer 5 of the mouse somatosensory cortex. She selects the search function of the BAEM to execute a search against the NIP. She enters "mouse morphologies 'Layer 5'" and executes the search.
4) The search of the NIP returns a list of results that will be displayed in the BAEM.
5) If the BAEM is in List mode the search results will look similar to a Google search results list.
6) If the BAEM is in Tile mode, the results will look similar to a Project Tile view.
7) If the BAEM is in 2D atlas mode, the results will be shown anchored to the relevant spatial or sematic-spatial location in the current slice of the 2D atlas viewer.
8) If the BAEM is in 3D atlas mode, the results will be shown anchored to the relevant spatial or sematic-spatial location. Results clustering may be required if too many results share the same anchor location in the viewer.
9) Abigail selects the morphologies she thinks will provide a good source for synthesis parameters and adds them to COLL Project 3.
10) Abigail then selects a morphology or collection of morphologies to run her preferred synthesis parameter extraction Task on. She executes the Task as a Job and the extracted synthesis parameters are added to the current COLL Project.

2.2.15 Finding Data During Task Configuration (SP6COLL-UC-014)

This Use Case describes the interaction with the BAEM for the selection of data during the configuration of a Job prior to execution.

Primary Actor: Scientific User Abigail.

Success Scenario:

1) Abigail wants to run a Neuron Builder.
2) Abigail first creates a new COLL Project, COLL Project 4, and selects it as her current COLL Project.
3) Abigail then selects the ontological context in which the Neuron Builder will run.
4) The ontological context will be used to filter the list of available Builders that Abigail can select from.
5) Her current focus is Layer 5, so Abigail selects Layer 5 of the mouse somatosensory cortex as her build context.
6) She then selects the particular Neuron Builder she wants to use.
7) Abigail selects "Reconstructed Neuron, Genetic algorithm fit v1.4".
8) The Job configuration system will then guide Abigail to enter the data that are needed to execute the Task. This is accomplished by leveraging the Collaboratory content type specific editors to provide rich editors for the various content type inputs that the Job

requires. One example of the use of content type editors would be in the selection of an input reconstruction morphology with the Brain Atlas Embedding Module. The content type editor would list candidate morphologies in 2D thumbnail mode and allow the viewing of the morphology in 3D prior to allowing the selection of a particular morphology that matched User criteria.

9) Abigail selects the reconstructed seed morphology editor. Because the build is happening in the context of Layer 5 of the mouse somatosensory cortex, this is automatically entered in the search field when selecting a morphology input.
10) She has the option to modify the search criteria and execute the search using the same functionality described in Use Case 2.3.4 steps 4-8.
11) She can then select the appropriate morphology on which to base her new neuron.
12) Abigail then selects a computational resource on which to run the Builder and executes the Job.

2.3 HBP-COLL: Functional Requirements

2.3.1 Authentication and Authorisation (SP6COLL-FR-001)

1) Users must be authenticated against a central database.
2) Users must have access control based on COLL Project-specific groups of Users.
3) The Portal and its applications will be accessible only through the SSL protected https:// or wss:// protocols.
4) Single sign-on will allow a User to login once for most applications.
5) Certain applications may require re-authentication to perform some privileged operations.
6) Authentication will be standards-based.
7) The Platform will allow limited delegation, i.e. the User will be able to restrict services in the Platform constellation from accessing certain services on their behalf.
8) Access to COLL Projects, Artefacts, Parameters, Tasks, Workflows and their metadata will be controlled by the authorisation system.

2.3.2 Tasks and Workflows (SP6COLL-FR-002)

1) Tasks should be capable of being broken down into workflows.
2) Workflows should themselves be a Task.
3) Tasks should be repeatable. Enough information should be captured in the COLL to reproduce an execution. The Tasks should produce the same results if run with the same inputs (as captured by the COLL).
4) Tasks should be able to execute on the computing resources of multiple distributed sites wherever possible.
5) Some Tasks are expected to be exclusive to a single site. Tasks requiring Neuromorphic hardware would be one example of a single site Task.
6) COLL Tasks will need to have a mechanism to decide where to store output data. The options are either “User specified” or “local”.

7) User-contributed Tasks should execute in a tightly controlled sandbox with limited permissions, if possible.
8) Tasks will be registered in a Task Registry service, which will track the location of their source code and any installation dependencies they may have.
9) Software distribution mechanisms exist for distributing the COLL Task Framework and Analysis Toolkit to Developer Users. Examples of software distribution mechanisms might be a git, apt or yum repository. License restrictions must be obeyed.
10) Software distribution mechanisms exist to distribute Task Framework components to HPC Platform-managed computing infrastructure in a secure, consistent and repeatable manner.
11) Users are able to create new versions of existing software and register their new software in the COLL.
12) The system that allocates HPC Platform-managed computational resources for the running of Tasks as Jobs will respect COLL Project-specific quotas and data permissions. The COLL expects to delegate the management of storage and computational quotas to the HPC Platform.
13) The Task Framework should accommodate single machine, HTC cluster and HPC jobs.
14) The COLL needs to query a central registry of HPC Platform compute resources. Resources available to the HPC Platform should be registered in the central registry.
15) The Task Framework jobs will need to interact with authenticated COLL services at job completion. There is a need to address the possibility of authenticated session timeout and the potential unavailability of authentication tokens in various tiers of the HPC Platform Job Service.
16) The COLL will use execution quota services provided by the HPC Platform.
17) The COLL will use execution estimation services provided by the HPC Platform.

2.3.3 Search (SP6COLL-FR-003)

1) Artefacts uploaded to the COLL will be searchable by diverse metadata.
2) Artefacts produced by the COLL will be searchable by diverse metadata.
3) Artefacts in the COLL will be searchable across the COLL and Neuroinformatics Platforms
4) Tasks and workflows will be searchable by diverse metadata.
5) Parameter contents will be searchable.
6) COLL Project wiki descriptions will be full-text searchable.
7) The search service must provide a common search API for searching Neuroinformatics and COLL Databases for relevant metadata.
8) The search service must return the first page of search results in less than five seconds. Because it will take a series of incrementally refined searches to find a User's desired data, searches taking longer than five seconds to return the first page of 10-25 results will greatly reduce search utility.
9) Search functionality must support optional display of hidden data.

2.3.4 COLL Projects (SP6COLL-FR-004)

1) It will be possible for the Portal User or Service User to attach a wiki to Entities.
2) Permissions are COLL Project-wide.
3) Read permissions are required to read in from the COLL Projects. Some metadata from the COLL Projects will be globally visible to enable public discovery. Read permissions are granted by a COLL Project Administrator.
4) Write permissions are required to add data, link data and modify metadata. COLL Project write permissions are also required to execute Tasks in a particular COLL Project. A COLL Project Administrator grants write permissions.
5) Administration permissions are required to modify User COLL Project permissions. A COLL Project Administrator grants Administration permissions.
6) The COLL must provide a mechanism for COLL Project Administrators to modify COLL User permissions for the COLL Projects that the Administrator owns.
7) The COLL must provide a mechanism for a User to create a link in one COLL Project from a User readable data file in another COLL Project.
8) There is a COLL Project Viewer that embeds content type-specific viewers.
9) The COLL Project Viewer must provide controls for managing the Storage and Data Lifecycle.

2.3.5 Storage and Data Lifecycle Management (SP6COLL-FR-005)

1) Users must be identifiable and have access permissions on all services in the Portal.
2) There must be a metadata service with functionality for marking data as hidden.
3) There should be an obvious mechanism to request deletion of data by System Administrators.
4) The COLL expects to use the HPC Platform Storage for storage of various low-density data files.
5) The COLL will use services in the HPC Platform Storage API for accessing data that are not available in a local storage resource.
6) This storage should be allocated from a per-COLL Project storage quota. The COLL expects to delegate the management of storage quotas to the HPC Platform.
7) There will need to be an HPC Platform service that facilitates the reliable transfer of data between sites using high performance bulk transfer protocols.
8) The transfer service will need to be available to the COLL. This requirement suggests that the transfer service should be JSON REST-enabled. This doesn't mean that the actual transfer is done over REST, but that the transfer is initiated and managed over REST.
9) The HPC Platform Transfer service should enforce quotas of limited transfer resources. The COLL expects to delegate the management of transfer quotas to the HPC Platform.

2.3.6 Common Service Interfaces (SP6COLL-FR-006)

1) All services must use the HBP LDAP as their source of truth for providing User metadata and for authenticating Users. This authentication can be exposed through one of the authentication mechanisms described in Sections 2.3.2 and 2.3.3.

2) All services—web or otherwise—must log their system logs through the system-wide syslog Logging Service.

3) All Services must provide a method to determine service health for the system-wide Monitoring Service.

2.3.7 Web Services (SP6COLL-FR-007)

1) Web Services will offer a REST binding using JSON.

2) REST services must conform to Platform standards.

3) REST services must provide web accessible documentation in a standard location.

4) REST services must have Platform-provided client libraries for Python.

5) Access to REST services must be authenticated via OpenID-Connect that uses HBP LDAP as its source of authentication authority.

6) Web Service Interfaces must log API accesses. Ideally this will provide per-User accounting.

2.3.8 Non-Web Services (SP6COLL-FR-008)

1) Public facing non-Web Service interfaces must be authenticated with one of OpenSSH keys, Kerberos or X509 client certificates.

2.3.9 Brain Atlas Embedding Module (BAEM) (SP6COLL-FR-009)

1) The BAEM must be usable as a viewer of an arbitrary collection of Artefacts. Sources of Artefact collections include COLL Projects and collections of COLL and NIP search results.

2) Each Artefact in a given collection will be displayable in a 3D reference brain viewer, along with its spatial location, if this is available. If the spatial information is not available, the Artefact will be displayable anchored to its most informative semantic spatial location. If neither is available, it will be displayable without a brain viewer anchor.

3) Each Artefact in a given collection will be displayable in a 2D reference brain viewer, along with its spatial information, if available. If the spatial information is not available, the Artefact will be displayable anchored to its most informative semantic spatial location. If neither is available, it will be displayable without a brain viewer anchor.

4) The BAEM must be integrated with the Neuroinformatics search such that search results are treated as a collection of Artefacts.

2.3.10 BAEM Data Handling (SP6COLL-FR-010)

Image and geometry data download from the server to the browser. Compared with other Platform components, the amount of data downloaded to the browser will be large.

1) The download of various data elements from the atlases must be done in an asynchronous manner to ensure that the UI remains responsive.

2) The download process will be visible to the User. It must be possible for the User to cancel the download of a data set.

2.3.11 BAEM Imagery Service (SP6COLL-FR-011)

1) The imagery service must provide bandwidth-efficient query and image download capabilities to allow use of the imagery service by a browser with a broadband internet class connection (10Mbit/s).
2) Contents of imagery services need to be provenance-friendly. This strongly implies identifying URL specifiers for image queries and immutable underlying data.
3) The COLL will need a way to register imagery services.
4) The COLL will need to perform health checks on externally registered imagery services. This could be done at least in part by an interface to the HPC Platform.

2.3.12 BAEM: View Modes (SP6COLL-FR-012)

The BAEM will have the following modes, each of them automatically selected based on application context:

1) List View
2) Tile View
3) 2D View
4) 3D View
5) Neuron selection interface.

2.3.13 Reference Space Transformations (SP6COLL-FR-013)

It is expected that the various atlases will not share a common coordinate space. Consequently, the BAEM will need a mechanism to view data from multiple coordinate spaces.

1) The BAEM is expected to support a limited collection of Spatial Reference Systems. Data providers will be required to provide their data registered in a BAEM supported coordinate space.
2) There exists a Reference implementation of a Spatial Reference transformation system, either in service or library form.

2.3.14 Use Case Mapping

The table below indicates which Functional Requirements are used to satisfy each Use Case. An X in the matrix below indicates that the Functional Requirement in the column is necessary to satisfy the Use Case in the row.

SP6COLL-FR-XXX	001	002	003	004	005	006	007	008	009	010	011	012	013
SP6COLL-UC-001	X			X	X	X	X	X					
SP6COLL-UC-002	X	X		X	X	X	X	X					
SP6COLL-UC-003	X		X	X	X	X	X	X	X	X	X	X	X
SP6COLL-UC-004	X			X	X	X	X	X					
SP6COLL-UC-005	X			X	X	X	X	X					
SP6COLL-UC-006	X	X		X	X	X	X	X					
SP6COLL-UC-007	X					X	X	X					
SP6COLL-UC-008	X	X				X	X	X					
SP6COLL-UC-009	X		X		X	X	X	X					
SP6COLL-UC-010	X		X	X	X	X	X	X	X	X	X	X	X
SP6COLL-UC-011	X	X				X	X	X	X	X			X
SP6COLL-UC-012	X			X	X	X	X	X	X	X			X
SP6COLL-UC-013	X		X	X	X	X	X	X	X	X		X	X
SP6COLL-UC-014	X	X	X	X	X	X	X	X	X	X		X	X

Table 1: COLL Use Case to Requirement Mapping Table

2.4 HBP-COLL: Non-Functional Requirements

2.4.1 Interfaces

Major functions must be accessible from both Web GUI and programmatic Web Service clients.

2.4.2 Efficiency

Data handling needs to be compatible with HPC data management standards to facilitate efficient use of storage and I/O resources.

2.4.3 Reliability

- The COLL is expected to have regularly scheduled (twice monthly) maintenance windows. Each maintenance window will be no more than 30 minutes long.
- Individual non-Portal services that the COLL requires may be more unreliable, but the COLL will provide a service status page that will show the status of the services on which it depends.

2.4.4 Monitoring

The COLL will be monitored by an operations monitoring suite. This suite will report on usage, performance and health metrics.

2.4.5 User Volumes

The COLL web GUI is architected to support 500 concurrent Users. Individual services in the Platforms behind the Portal will have different SLAs depending on the services they offer.

2.4.6 Data Volumes

Data upload through the web GUI will be restricted to an upload limit. The COLL doesn't handle data directly when the files are larger than the upload limit. Larger files will be handled by HPC Platform services.

2.5 HBP-COLL: Architectural Overview

The HBP-COLL is architected to be a best-of-breed Software-as-a-Service (SaaS) offering. In recent years, this has proven to be a robust model for organising large, loosely coupled software platforms in industrial settings, and has also proven to be a viable model for software delivery, through extensive exploration in Blue Brain Project prototypes. Because the COLL is intended to be extended by Users in non-trivial ways, it will provide Users with capabilities similar to a Platform-as-a-Service (PaaS) offering, under certain Use Cases.

As a result of its SaaS and PaaS ambitions, the COLL is made up of components that are accessed through a network-accessible service API. The decision in favour of service orientation has implications for the architecture in terms of development, testing, deployment and monitoring discipline. Failure scenarios must be handled gracefully, and regular service upgrades must proceed with minimal User impact. User authentication must be handled in the most secure manner possible, while still maintaining ease-of-use for non-technical Users. With all of this taken into consideration, the architecture

presented below will provide a foundation that will enable success scenarios for Users in a reliable, secure and user-friendly fashion.

2.5.1 Architectural Principles

- Mostly immutable data – the COLL attempts to preserve as much data as possible from interactions with scientists. This is done primarily for reasons of scientific reproducibility. However, the same data can be used for attribution and citation, which are particularly important in a collaborative scientific environment. For this reason, true deletion of data can only be performed by INFRA Users/System Administrators. Instead, COLL Project Administrators can hide data. Hiding data makes them invisible under the default UI settings. Users who have read access to the data in the parent COLL Project can still make hidden data visible using the Web UI.
- Permissions model – the COLL permissions are allocated in a very coarse-grained manner, and are intended to be uniform across all files in a particular COLL Project. Permission sets available are:
  - Read
  - Read and Write
  - Administer (Read, write, hide and request deletion).
- Sharing groups – the Permissions model also allows for a small number of sharing groups. Sharing groups would imply read permissions on all files owned by the shared COLL Project. Sharing group changes would be carried out at a COLL Project level, and would only be available to a COLL Project Administrator. The small number of sharing groups may change in the future, but generally remain static. These groups would include:
  - Partner – readable by all employees of a particular Partner
  - HBP – readable by all HBP Consortium members
  - Non-HBP Party – readable by a registered Party which is not a Partner of the HBP
  - Universe readable – readable by all registered Users of the COLL.
- Provenance Tracking – Provenance tracking is the practice of tracking where things come from, and where they have gone. In the HBP, this is the tracking of software versions and software execution environments, and their data inputs and outputs. Provenance is used to establish reproducibility of computational Tasks, and to attribute credit for data production and tool development. The HBP COLL implements PROV-DM to track links between Entities, Activities and Agents. In the COLL, Entities are usually files, Activities are usually Task Framework Tasks, and Agents are usually Users. The COLL uses this information to show links between related components of the system. Wherever possible, encouraging immutable data supports the provenance-tracking goal.
- Components – The Collaboratory can be extended with the following components:
  - Apps - web GUIs for certain services integrated into the Collaboratory. In most cases, these components will also use Collaboratory services.
  - Tasks – provenance- and dependency-tracked, asynchronously-executed algorithms.

- Services - web services, network file systems, SSH, source control (git), continuous integration, databases, configuration and deployment services. Software-as-a-Service, Platform-as-a-Service and Infrastructure-as-a-Service offerings.
- Foundation Software - analysis libraries, simulators, and data access libraries, as well as thick client applications (desktop visualisation tools).

2.5.2 High-level Component Block Diagram

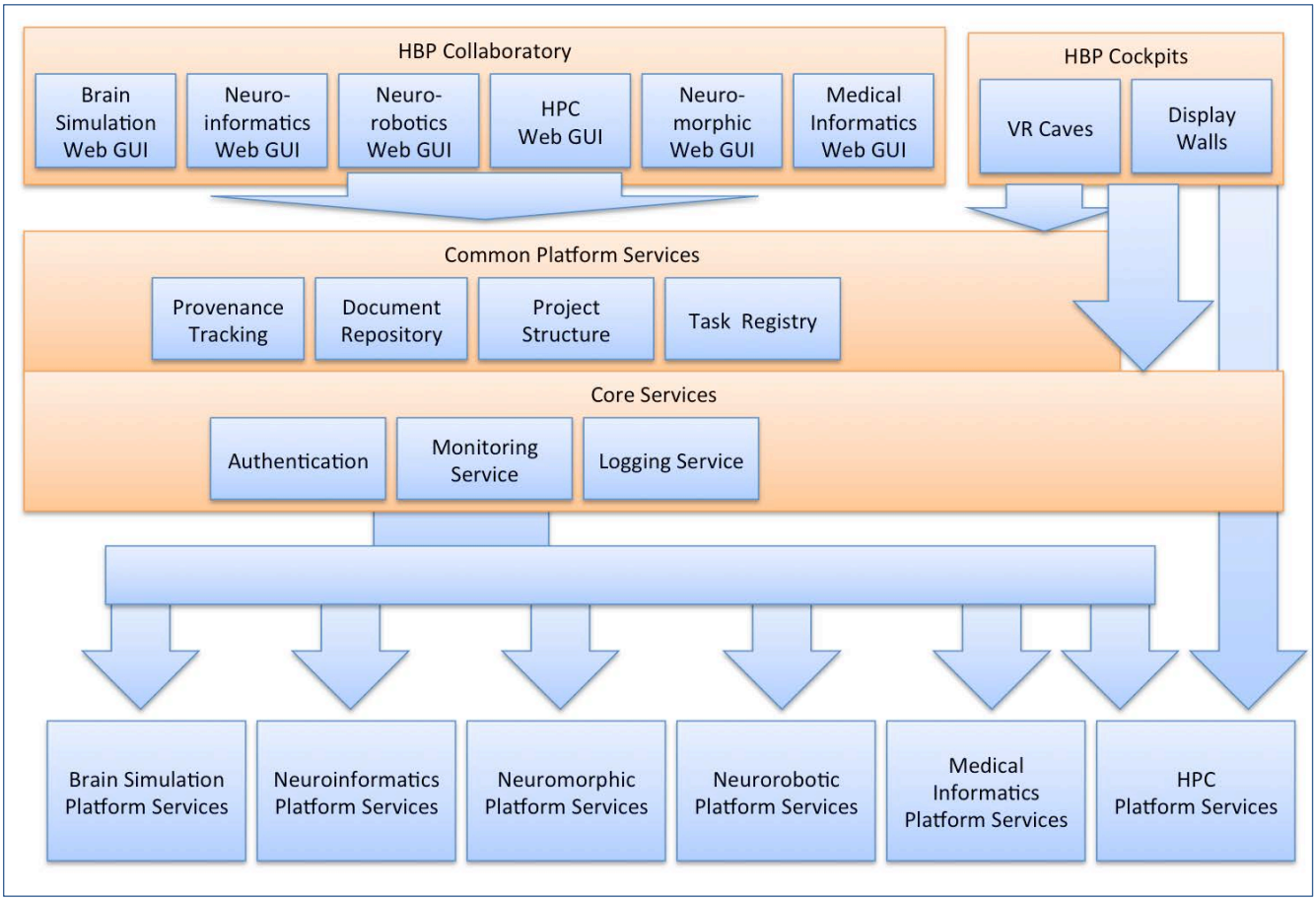

Figure 1: High-level Component Block Diagram

2.5.3 App Components

App components are intended to allow developers who wish to extend the Collaboratory the ability to write standard web applications using any client-side HTML functionality, and any backend service. Typically, these apps would also integrate with several key Collaboratory services, such as Authentication, Monitoring and Logging. Collaboratory Apps would also be integrated with other Collaboratory Services. However, the level of integration with Collaboratory services is entirely the decision of the App developer, and should be decided on the basis of the App's provided functionality.

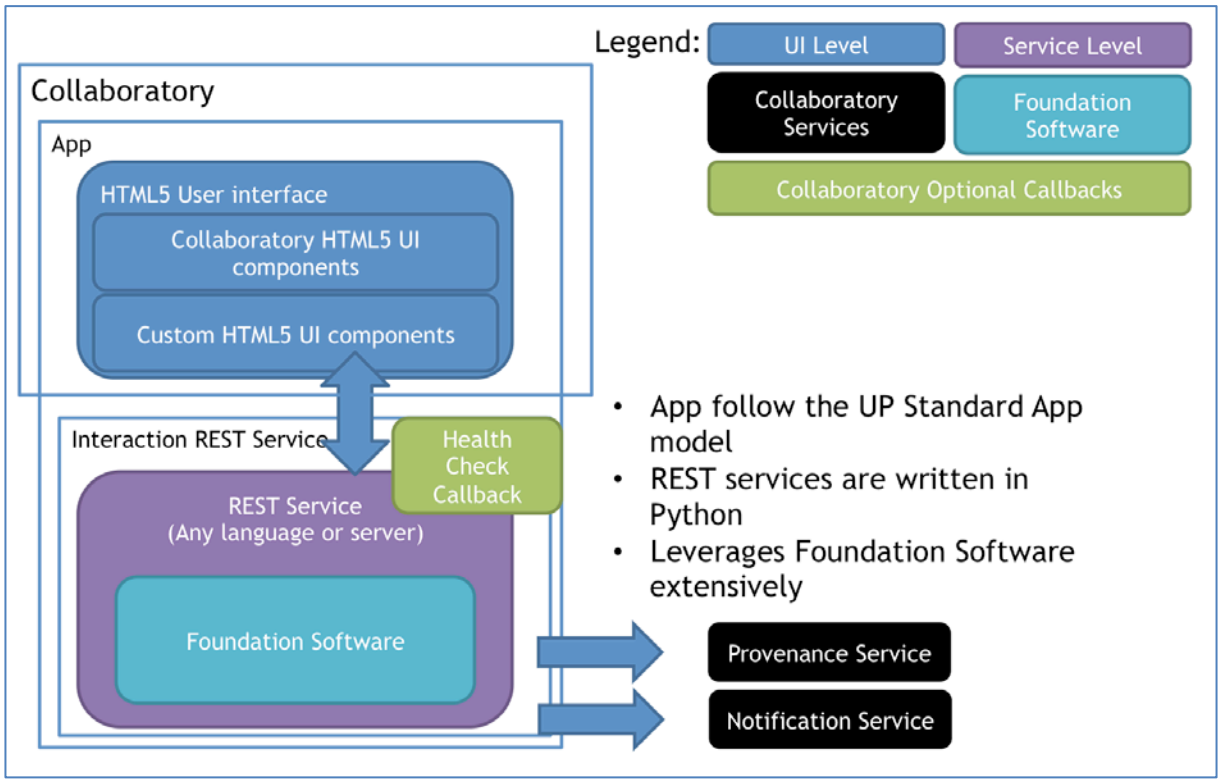

Figure 2: Collaboratory App Architecture

The Collaboratory Authentication system and Service-Oriented Architecture (SOA) allows a loose coupling of HBP and third party apps to the Collaboratory. This same approach also allows the Apps and their supporting web service to be deployed to any Base Infrastructure. This is key to the long-term federation strategy, which will be developed into the Operational Phase.

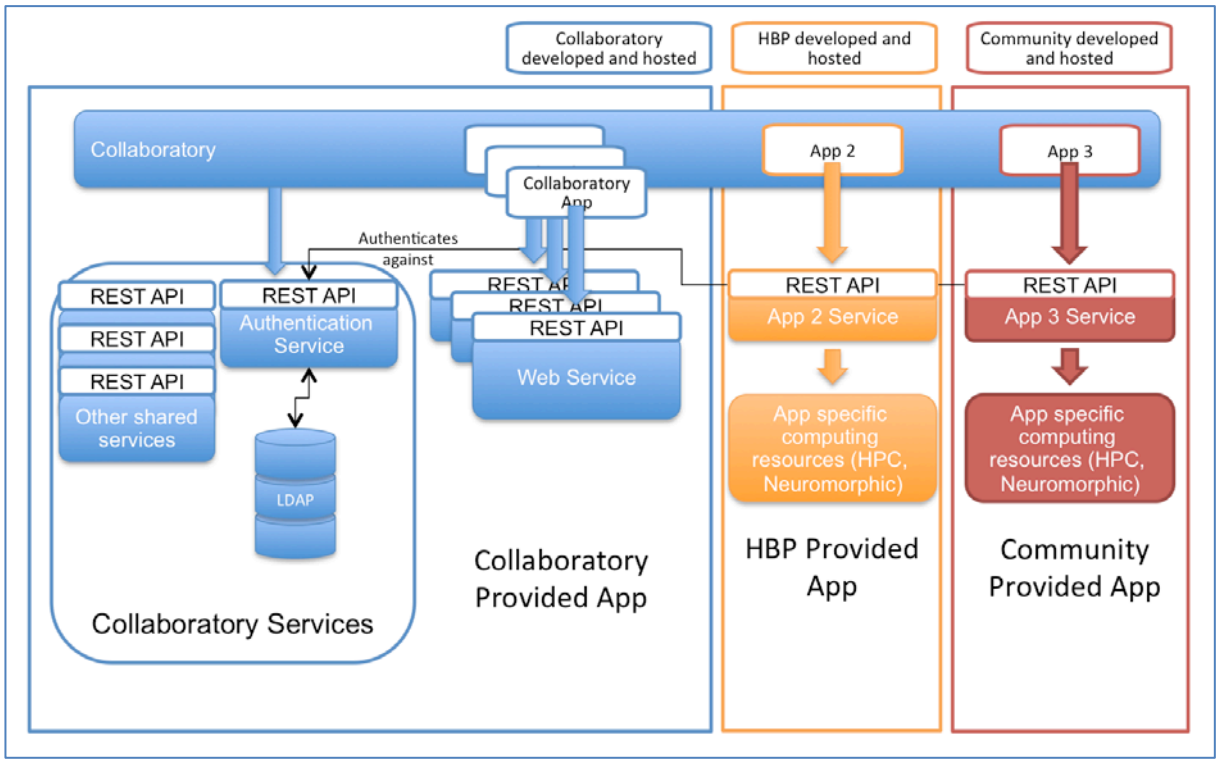

Figure 3: App Infrastructure Relationship

2.5.4 Task Components

The Task Framework is a component model and a collection of software that allows the remote execution of Jobs on different resources, including supercomputers and computer clusters. It is composed of:

- A web service that coordinates the Jobs.
- A set of software packages installed, per Job launch, on the resources for runtime support.
- A Software Development Kit (SDK) that allows collaborators to provide new executable code to be launched as Jobs.

2.5.4.1 Objectives

The system is designed for users to be able to contribute and share algorithm implementations, while keeping track of the provenance of the artefacts generated, and ensuring reproducibility and accountability.

These implementations are named Tasks. Tasks are bundles of code that are kept in a source control repository, which need to be registered with the Task Framework web services. These Tasks must specify their input and output types, as well as their software dependencies, in a manifest that is provided at registration.

The execution of the code of a particular Task, with certain parameters provided by a user, is called a Job. The inputs and outputs of the jobs are registered on the Provenance Tracking service. Because the Task is a particular commit in an immutable Version Control System, this simplifies reproducibility.

The Collaboratory contains a dedicated web UI that allows all users to interact with the service to manage Jobs. This includes an automatically generated Web Graphic User Interface (Web GUI) for Job launching, which is built based on the properties of the Task described in the manifest.

2.5.4.2 Architecture

The general architecture of the execution core of the Task Framework comprises three main parts: the *container_port*, the *container_manager* and the *active_worker*.

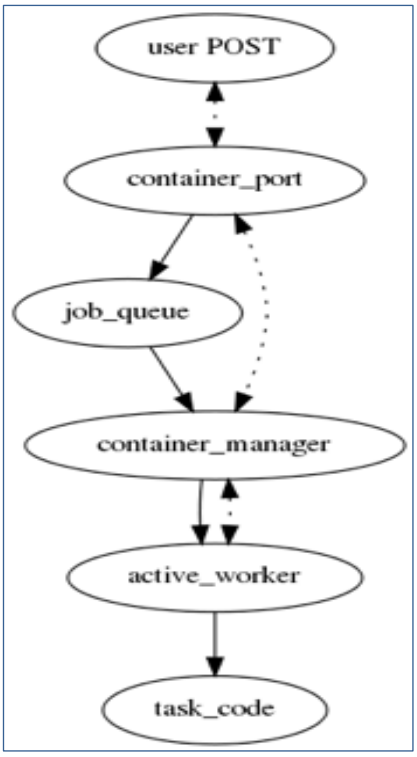

Figure 4: Task Architecture Relationships

The *container_port* is the web service that acts as a Job broker, and is constantly running. Its purpose is to keep track of Tasks and Jobs. Internally to the framework, it initiates Job launching and cancellation. It also responds to requests from Jobs for different events, such as uploading/downloading files, registering information on provenance, etc. From an external point of view, it forms the main interface to Task management functionality through its REST API:

- For Tasks, it holds the registry of Tasks. This allows Tasks to be added and updated, while ensuring that they contain the correct properties. It also enables querying of these properties on any registered Task.
- For Jobs, it can be used to launch new Jobs, query the state of existing Jobs, and cancel any that are running.

The *container_manager* is a monitoring process for an individual Job. The code for the *container_manager* is part of the runtime libraries that must be available on the hardware resources. One instance is run for each Job, and they are spawned by the *container_port*. The *container_manager* handles communications and requests with the *container_port,* such as fetching data, notification of Job status, return values, and exceptional conditions.

It also ensures that the environment and dependencies of the Task code are ready for execution.

The *active_worker* is a very thin wrapper around the Task code. It is part of the runtime libraries installed on the hardware resources. It handles the loading of the User's Task code, and provides an API for the Task code to initiate requests that are to be fulfilled by the *container_manager* and *container_port*.

Figure four is a graphical depiction of the architecture. Dotted lines are communications channels, and solid lines are process execution events.

A general job launch follows these steps:

1) User POST is a REST call initiated by an external client (most likely the web UI).
2) The container port communicates with the *job_queue* of a specific resource (for example Slurm, Unicore, etc.) to instantiate a *container_manager*.
3) The *container_manager* sets up the environment and creates a different *active_worker* process. It also sets up the communication channel with the *container_port*. The separation of the *container_manager* and *active_worker* processes here allows for each process to be executed by different users, which could be used for user impersonation. It can also act as a natural place for containerisation.
4) The *active_worker* loads and executes the Task code, forwarding data requests to the *container_manager* as needed. These then are forwarded to the *container_port*.
5) Once the Job is finished, *active_worker* and *container_manager* notify the *container_port*, clean up the environment, and shut down.

The Task Framework is written in Python 2.6, a widely supported Python distribution.

2.5.4.3 Task SDK requirements

Although the Task Framework is not meant to run arbitrary code, it imposes only a small set of requirements on the contributed code.

Scientific developers are expected to contribute their Tasks in Python 2.6. However, developing Python wrappers for algorithm implementations in other languages is relatively simple. This may be improved with language-specific helper libraries in future versions.

Every Task must be located in a git version control repository, and a specific commit ID must be provided for registration with the system. The system must be able to clone this repository. Therefore, it must be either publicly available, made available to a service account, or another security measure must be agreed and implemented in collaboration with the computing site administrators and the Task contributors.

In the manifest that accompanies each Task, certain properties of the Task must be specified:

- Full Name: a short but human-friendly name for the Task.
- Caption: a one-line description of the Task. Used in the UI for quick Task look-up.
- Description: a human-friendly description of the Task purpose and usage. *Markdown*, a simple markup language, can be used to format this text, which will be properly rendered in the web UI.
- Author: name, email or other identifier of the main point of contact for Task maintenance.

- Categories: a list of text tags for the Task. It is used to group Tasks that perform like functions, to make it easier for users to browse Tasks. These are not limited to a fixed list of possibilities; instead, contributors are allowed to introduce tags they feel better describe their Task. However, a few are predefined, and are expected to be adopted by the majority of contributors: *Simulation*, *Validation*, *Analysis*, *Test* and *Translator*.
- Inputs: every input of the Task must be explicitly typed, with either a base type, or a type from the Mime type service. An input may also provide extra information, such as human-readable descriptions. Ideally, other metadata for each input will be accepted in the future, such as default values, or units.
- Outputs: return values are also expected to be named and typed (with a base type or a Mime type service entry).
- Dependencies: a list of python dependencies can be provided for each Task. The format of each item in this list is exactly the same as the format used when installing Python packages: name==version.

All of these properties are accessible through the REST API and the generic web UI.

2.5.4.3.1 Task SDK Dependency Handling

Every Task dependency will be installed on every Job execution by Python's *pip,* the *de facto* package management system for Python. This is so that the system remains generic, while providing a clean environment for the Job to run. However, if some of those dependencies take a long time to install (usually because of C++ dependencies that need to be built), users of that Task will pay for the installation time on every Job run. In other situations, it may not be possible to install dependencies via *pip*, and will require a manual installation (usually complex C++ libraries with Python bindings).

In either of those cases, adding pre-installed packages to the set of runtime support libraries installed in each resource is a preferable compromise. If a Task has a dependency on any pre-installed package, the package will already be built and available for every Job execution. Because the procedure of installing runtime libraries may vary from one resource to another, the addition of new preinstalled packages will be coordinated between the scientific developers who require them, and the site administrators. A few examples of pre-installed standard packages are *matplotlib*, *numpy* and *h5py*.

2.5.4.4 Data types

The definition of a Task's inputs and outputs can be tagged with base types: long, double, string and bool. It is also possible to specify homogeneous lists of any of these types, e.g. list(long). However, collaborators usually develop Tasks that consume data in a diverse range of file formats. To simplify the integration of these Tasks, the Task Framework extends the concept of Uniform Resource Identifier (URI) types.

All files registered with the storage system, and all file inputs and outputs of Tasks registered in the Task Framework are expected to be tagged with a URI type metadata. This serves multiple purposes:

- It enables dedicated rich viewers and selectors to be associated, in order to interact or explore the inputs and outputs of Tasks.
- It provides Task writers with certain guarantees on the file format their Tasks will deal with, which should make the integration of existing data formats simpler, as well as simplifying migration between versions of code and data formats.
- It makes possible the future linking of Tasks, to compose complex workflows.

A dedicated service with a REST API acts as a registry of these URI types. For each type, a human-readable description, a name, and specifications can be registered. Collaborators can register and maintain these types through the API and web interface, making the system extensible. This service could be integrated with other neuroinformatics systems to simplify the search process, ontological exploration and discovery of data.

2.5.4.4.1 MIME Type specification

The actual implementation and encoding of URI types must be simple and easily serialised to text. For this reason, the Multipurpose Internet Mail Extensions (MIME) type internet standard was used as a template that identifies the format and type of data contained in files.

The adopted basic schema of the MIME types is:

top-level / [tree.] name [+suffix] [; parameters]

The project's needs are largely satisfied with the following top-level types: application, example, image, model, text, video.

The vendor type for custom internal file formats allows quick identification of the origin of a type. For example "vnd.bbp" can be used as the initial tree section for file formats developed within the BBP group. More general "vnd.hbp" MIME-types are expected to appear as new file format specifications emerge across the Consortium.

The suffix is the actual file suffix (e.g.: xml, txt, h5), and can be used to describe the different file formats for the same content. For instance, morphological data can be stored as ASCII in an asc file, or HDF5 in an h5 file.

To version different MIME-types, the *'; parameters'* functionality is used.

As an example, the MIME-type 'application/vnd.bbp.Simulation.BlueConfig; version=2' would identify files containing simulation configuration files developed within the BBP, in the second iteration of the respective specification format.

2.5.4.5 Documentation

The system is highly extensible, via two main interfaces exposed to developers: a REST API and the SDK.

The Task web service exposes a REST API, which external developers can use to build custom clients if the generic web UI is not suitable for their needs. The REST API can be used to create custom rich user interfaces, while still benefiting from the management and tracking offered by the Framework. Other custom clients are also possible. For example, the SDK includes a python library that lets users interact with the service directly, and that is built on top of this REST API. This API is documented in an exposed web UI that is automatically generated using *swagger*. This is a popular framework for API, which ensures that the documentation is always up to date and accessible.

The scientific developers will interact with the SDK to contribute new pieces of code. The SDK is a set of tools and libraries that allows Tasks to be developed and tested. The Python package exposes an API, which during actual remote execution will be implemented by the *active_worker,* but also has a local simplified implementation for testing the Task before execution. The utilities include tools to register new Tasks or update existing ones.

The documentation for the full SDK can easily be generated in an HTML format using Sphinx, a *de facto* standard in the Python programming community. For every minor release of the SDK, it is automatically built and uploaded to a documentation server that is

easily accessible from the Collaboratory site. The Task SDK will contain examples of Tasks and Task development tutorials.

2.5.5 Foundation Software

Having a stable software foundation is critical for the neuroscience community to exchange data and ideas. Foundation Software is the software which supports the rapid development of much of the functionality in the Apps, Tasks and Service of the COLL and the BSP.

Key properties of Foundation Software:

1) Packaged for deployment on HBP Base infrastructure.
2) Available to Tasks using a standard dependency specification scheme.
3) Python packages are *pip* installable using standard Python tools and an HBP specific package repository.
4) C++ modules are available as modules on the HBP Base Infrastructure.
5) Platforms will contribute their own software to the Foundation Software catalogue.

2.5.6 Standard REST Web Service API

A COLL standard REST API has the following properties:

1) Python REST — implemented using BBP Tornado standard worker pooling behind an NGINX frontend.
   a) For database interaction, SQLAlchemy or Django persistence models are preferred.
2) Python, Java or Scala REST services include standard Rest API documentation, and a python API used in service integration and load testing.
3) Official Collaboratory REST APIs are expected to support client authentication using OpenID-Connect access tokens.

2.5.7 Additional Service Components

2.5.7.1 HBP Collaboratory

The goal of the Collaboratory component is to provide a unified web interface for the various components provided by the other HBP Platforms. The Collaboratory component also owns the various user interface libraries used to construct rich HTML5 user interfaces.

2.5.7.2 Authentication Service

Authentication is the act of confirming a User's identity by requesting a piece of information that only they should be able to provide. This can be a password or a number derived from a private key via a cryptographically irreversible process.

The COLL will include an Authentication Service that implements the OpenID-Connect standard to authenticate REST APIs and Web applications.

This standard is used to allow web or native application Single Sign-On (SSO) for applications using REST APIs provided by the various HBP Platforms.

More information on OpenID-connect can be found here: http://openid.net/connect/.

2.5.7.3 Logging Service

The Logging Service provides a central collection and analysis point for logs generated by the various Portal and Platform services.

The Logging Service is supported by standard logging libraries that will be implemented on top of the following libraries:

- Java/Scala: log4j
- Python: standard logging package
- C/C++: log4c / log4cxx
- Matlab: TBD.

The Logging Service supports the following:

- Multiple handlers/appenders
- "Topics" and standard logging levels
- Configuration without recompilation.

Common log levels between the various languages:

- CRITICAL: A serious error, indicating that the program itself may be unable to continue running.
- ERROR: Due to a more serious problem, the software has not been able to perform some function.
- WARNING: An indication that something unexpected happened, or that there will be some problem in the near future (e.g. 'disk space low'). The software is still working as expected.
- INFO: Confirmation that things are working as expected.
- DEBUG: Detailed information, typically of interest only when diagnosing problems.
- TRACE: Only in development environment.

Topic standard:

- Java package-like pattern used in all languages:
  - bbp.viz.orbbp.soa.eg: bbp.soa.restservices.myclass
  - Packagename in [a-z,0-9,-,_] (all lowercase)
  - Between 3-20 characters
  - Depth of no more than 10.

2.5.7.4 Provenance Tracking Service

The Provenance Tracking Service is implemented as a standardised REST API. It follows the PROV-DM model specified here: http://www.w3.org/TR/prov-dm/.

The Provenance Tracking Service supports only the PROV-JSON serialisation format. PROV-JSON has been submitted to the W3C here: http://www.w3.org/Submission/2013/SUBM-prov-json-20130424/. The COLL PROV-JSON serialisation has been extended with necessary attributes, following the extension namespace model in the above submission. The extensions will be documented as part of the final API.

2.5.7.5 Monitoring Service

The monitoring service continuously parses all input to the Logging Service to generate actionable responses. Examples of possible actions:

- Email to a responsible party
- Scripted Tasks
- Service lifecycle management (restart, stop, start, etc.).

2.5.7.6 Document Repository

Provides a standardised REST API.

2.5.7.7 Brain Atlas Embedding Module

The Brain Atlas Embedding Module is a web-based 2D and 3D application component with various roles in the Brain Simulation Platform. It is intended to be a used as a Task-specific Neuroinformatics Platform search interface for the various BSP Use Cases. It is intended to provide a mechanism to interactively explore existing circuit models in various stages of brain building. Finally, the BAEM will be used for selection of subsets of circuit entities (pathways, cells, layers, etc.).

It is important to note that data curators will still primarily use the 3D Brain Atlas building tools in the Neuroinformatics Platform.

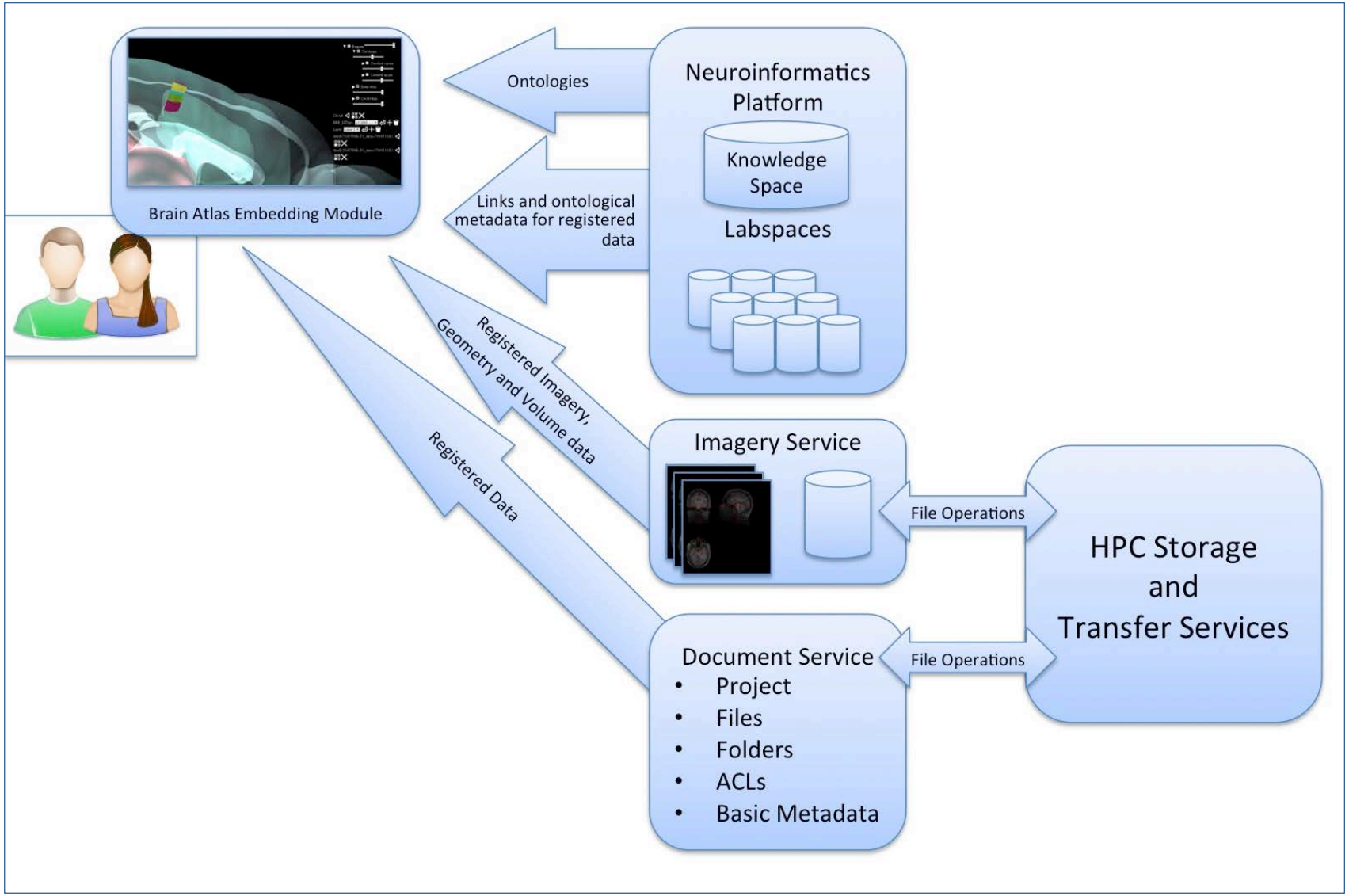

Figure 5: Brain Atlas Embedding Module Interaction with other Platform Services

The 2D and 3D views of BAEM will largely be built on top of WebGL, using a well-developed javascript game engine http://threejs.org/. ThreeJS has been chosen over higher-level toolkits such as XTK (http://www.goxtk.com/), due to the low-level control it offers. Significant client-side development is planned, to allow the BAEM to load large neuroscientific data sets into browsers connected to the internet via standard broadband links.

2.5.7.8 Imagery Service

The BAEM will be a consumer of voxel data sets coming from Magnetic Resonance Imaging (MRI) and Electron Microscopy. The imagery service is a rest service that will use 3D image pyramids to provide data to the BAEM via REST based Level-of-Detail (LOD) image queries. The Imagery Service will provide a COLL Standard REST API.

2.5.7.9 Multi-Search Service

The Multi-search service is a combined effort of the Neuroinformatics and Collaboratory Teams. This service is intended to provide unified search indexes across all services provided by the Collaboratory and Neuroinformatics Platforms. This work will also define a standard API for populating the index. This service will most likely be implemented on top of ElasticSearch. ElasticSearch is an open source horizontally-scalable search index. See http://www.elasticsearch.org/ for more details.

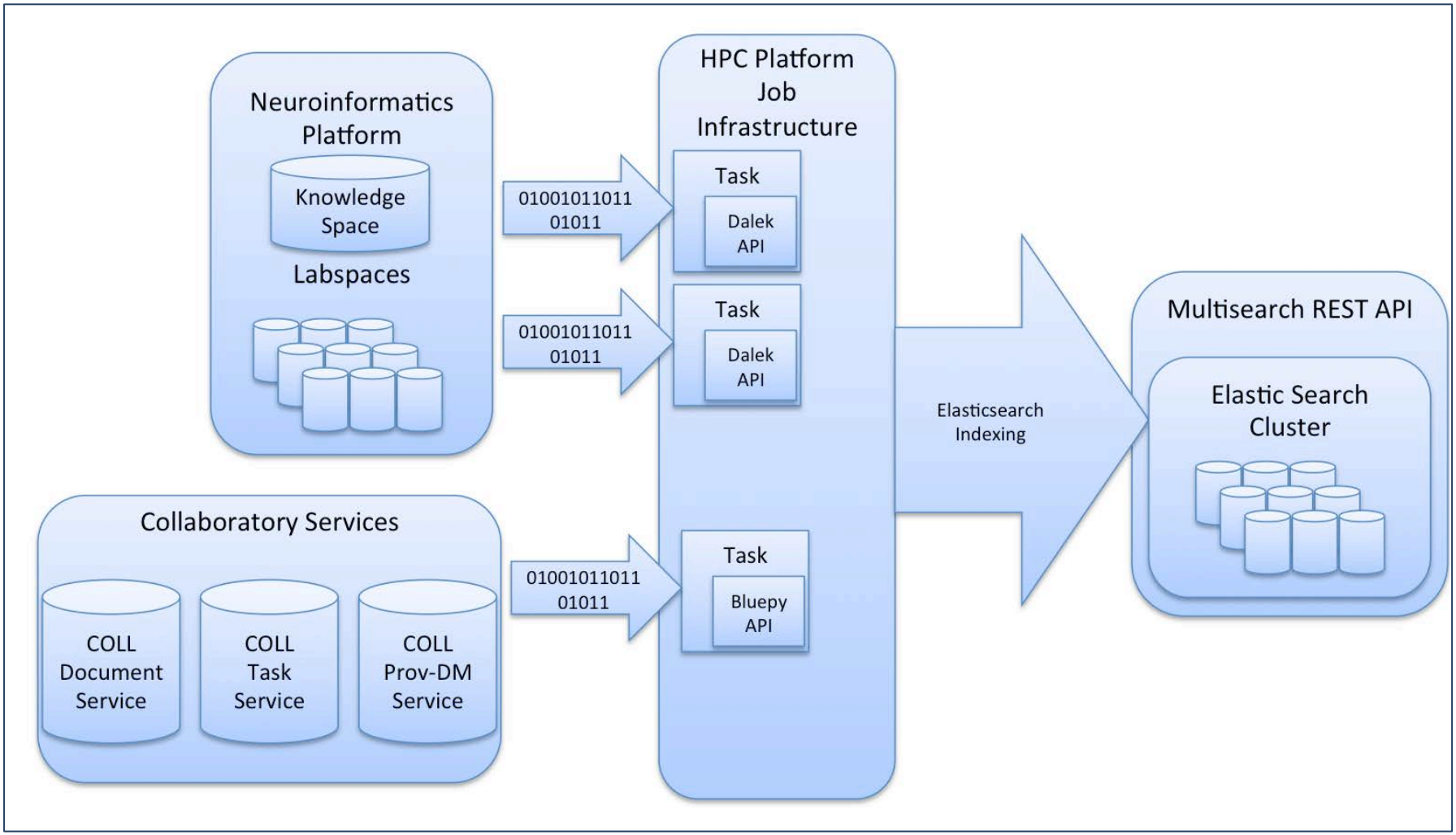

Figure 6: Multi-Search Service interaction with other Platform Services

2.6 HBP-COLL: Physical Architecture

The Physical Architecture of the COLL is largely based on standard best-of-breed infrastructure practices.

- Each REST Service owns one or more VMs.
- External services are accessed primarily through REST.
- Internal services will use direct network filesystem connections (GPFS and NFS) and non-REST web-service connections where necessary.
- Apache TrafficServer acts as a proxy in front of all web traffic to allow enforcement of global traffic management policies.
- Provisioning and configuration of VM resources will be done by a centralised configuration system (puppet) linked to a source controlled configuration repository.

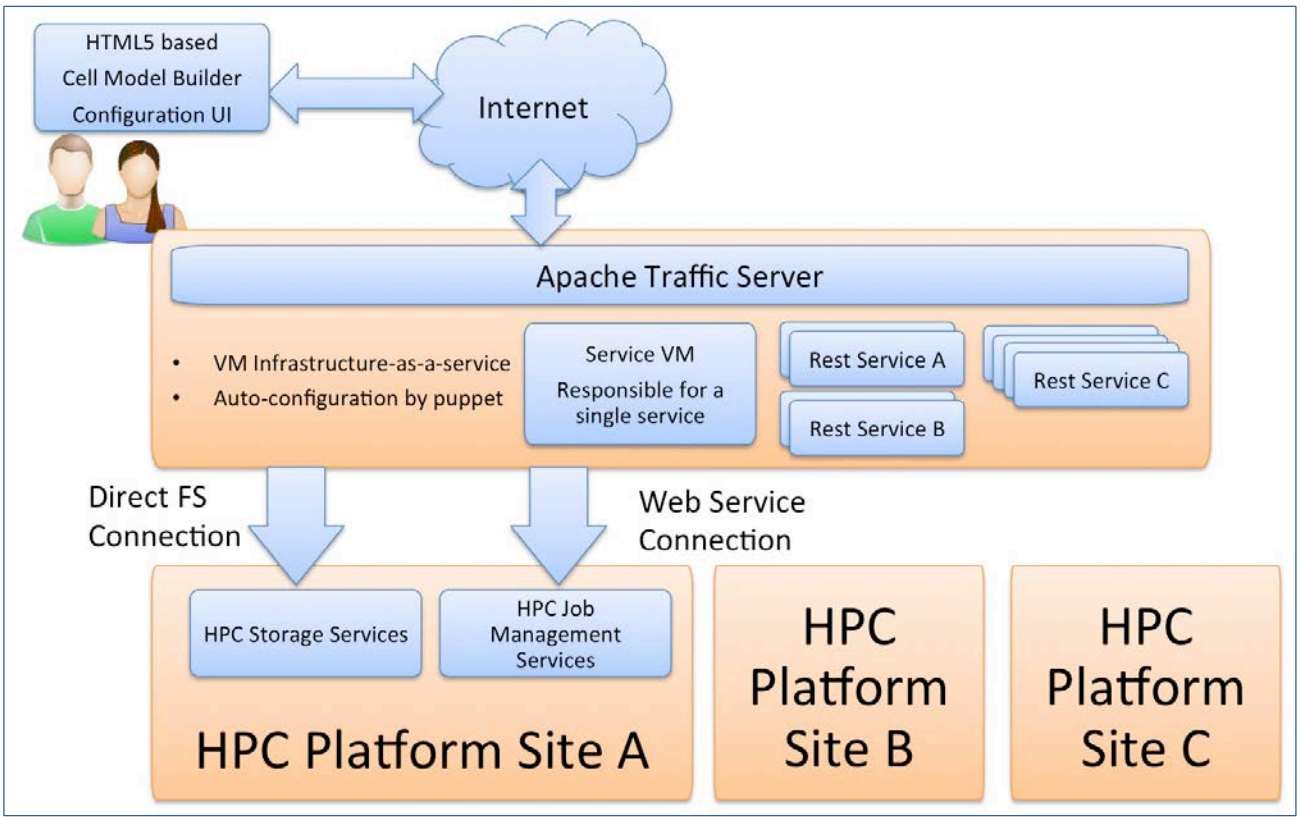

Figure 7: Physical Architecture

The initial target list of HPC Platform sites is to be decided by the HPC Platform team, but the set of services provided by each site is expected to be uniform.

2.7 HBP-COLL: Relations to other Platforms

2.7.1 Functional Requirements on other Platforms

Use Case	External Platform Dependencies
SP6COLL-UC-001	NIP for ontologies NIP Function Requirement 1.3.1 for HBPMIN HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6.
SP6COLL-UC-002	HPC Functional Requirement 1.4.2 and 1.4.6.
SP6COLL-UC-003	NIP for ontologies
SP6COLL-UC-004	HPC Functional Requirement 1.4.2 and 1.4.6.
SP6COLL-UC-005	HPC Functional Requirement 1.4.2 and 1.4.6.
SP6COLL-UC-006	HPC Functional Requirement 1.4.2, 1.4.3, 1.4.4, 1.4.5 and 1.4.6.
SP6COLL-UC-007	HPC Functional Requirement 1.4.2.
SP6COLL-UC-008	HPC Functional Requirement 1.4.4, 1.4.5 and 1.4.6.
SP6COLL-UC-009	HPC Functional Requirement 1.4.2 and 1.4.6.
SP6COLL-UC-010	HPC Functional Requirement 1.4.2 and 1.4.6. Neuroinformatics Platform for ontologies
SP6COLL-UC-011	Neuroinformatics Platform for ontologies
SP6COLL-UC-012	Neuroinformatics Platform for ontologies
SP6COLL-UC-013	Neuroinformatics Platform for ontologies
SP6COLL-UC-014	Neuroinformatics Platform for ontologies

Table 2: COLL Functional Requirements on other Platforms

2.7.2 Services Provided to other Platforms

The COLL will provide the following services to other platforms:

- Provenance tracking REST services
- Document storage REST services
- Project tracking REST services
- Task Registry REST services
- Job Scheduler REST services
- Web-based project visualisation container services
- Web-based data visualisation tools

- BAEM for search and data viewing
- Imagery service

2.8 HBP-COLL: Dependencies

This is an overview of non-platform development dependencies.

2.8.1 Required

The following activities are required to satisfy the minimal Use Cases defined in Section 1.3 of this document. They should be done in consultation with the European Research Programme Office and Intellectual Property and Technology Transfer Manager,

- End User License Agreements development
- Data Use Agreements for User uploaded data
- Code Use Agreements for User uploaded code.

2.8.2 Preferred

The following activities will be necessary to achieve the full potential of the COLL.

- Early release with partial functionality to allow teams throughout the HBP to focus their integration efforts. The COLL development team has adopted an Agile methodology, which encourages project teams to “Release Early, Release Often”. See the section on KPIs for more information.

3. The Brain Simulation Platform (BSP)

3.1 BSP: Overall Goals

During the Operational Phase, one of the HBP's Strategic Flagship Objectives is to achieve a unifying multi-level understanding of the brain. A Specific Project Objective is to simulate the human brain. Simulating the human brain involves *digital reconstruction and simulation* of the mouse brain and ultimately the human brain using experimental data and fundamental principles of brain organisation. The mouse brain provides a source of multi-level data needed to develop the Brain Simulation Platform. The data from other animals and human-specific brain data will be used to configure the brain building process to produce a first draft reconstruction the human brain.

Reconstructing brain tissue as high fidelity digital computer models requires a fundamentally different approach to brain modelling. Previously, the primary objective was to model target phenomena—usually a selected function. Such top-down modelling is a very well established field and is a powerful driver for the discovery of principles and the development of theories of the brain. The bottom-up reconstruction approach is currently missing. The Brain Simulation SP will establish this field of research. The Theory SP will develop the top-down approach further, and also attempt to bring the top-down and bottom-up approaches together during the course of the project.

During the Ramp-Up Phase, the BSP will be built to seamlessly integrate model building, validation, simulation, visualisation and analysis in a cohesive user-friendly web interface. The brain models are composed of model components at a variety of levels, as well as models of their interactions. The components that might eventually be modelled in the BSP include genes, proteins, synapses, cells, microcircuits and brain regions. Because the models are data driven, the BSP is designed to facilitate systematic reconstruction of biologically accurate brain models and their components by using biological data from the Neuroinformatics Platform (NIP). In most cases this biological data needs to be rigorously analysed and algorithmically refined using NIP and BSP tools to extract fundamental parameters of the model components.

Validation of the model's biological properties is a key part of the HBP brain simulation strategy. As a result, the BSP will also integrate a wide variety of repeatable validations against biological data. This will serve to drive the brain models towards convergence with the biological systems that inspired them.

To facilitate deeper investigation and validation of the models, the simulators described in the later MolSim, CellSim and NetSim sections will be integrated into the BSP. The BSP will also integrate analysis tools developed by the Neuroinformatics project, the HPC Visualisation teams and others to observe and analyse chemical, biophysical, biomechanical, electrical and electromagnetic activity states of interest.

Computational tools used at different stages of the BSP workflows make use of diverse computational resources – particularly HPC resources. To do this efficiently and effectively normally requires a high level of supercomputing expertise. To make these resources accessible to the *non-technical User*, the BSP will provide a user-friendly facade behind which the HPC subproject of the HBP will advance best-of-breed HPC hardware and software development practices.

Finally, the integration of the BSP into the HBP-COLL will allow for the exchange of data and models between the Neuroinformatics, Brain Simulation, Neurorobotics, HPC, Neuromorphic and ultimately the Medical Informatics Platforms. The same analysis tools

that will be used to compare reconstructed models and hardware models will be used to compare high-level models in the Theory subproject. This will enable more rapid evolution of the software models, neurorobotics applications, and neuromorphic hardware models than would otherwise be possible.

3.2 BSP: Use Cases

To deliver iterative refinement of state-of-the-art brain models, the Brain Simulation Platform needs to be built from modular interchangeable components with well-defined workflows.

The figure below shows the high-level interaction between the various Brain Simulation Platform components.

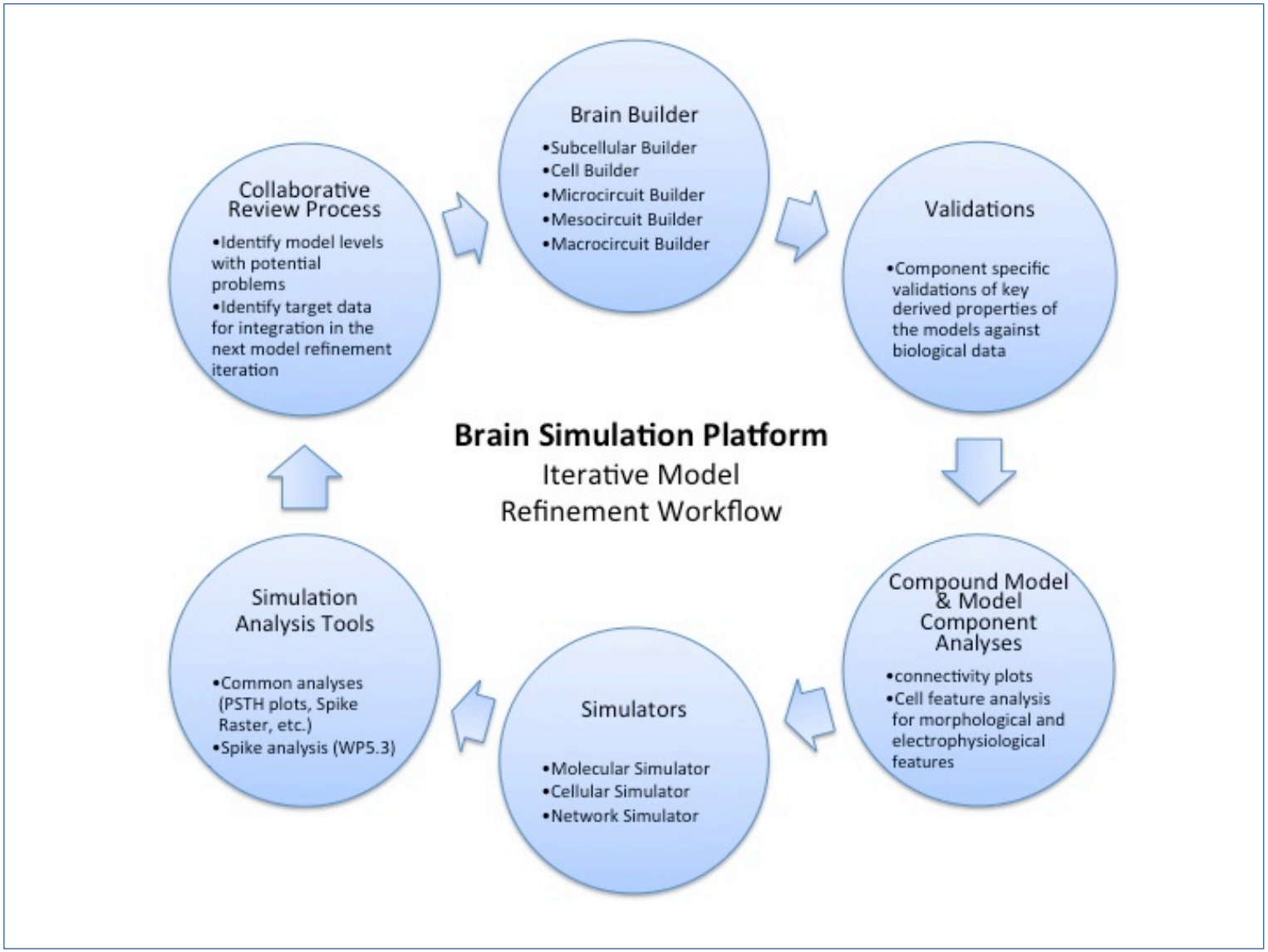

Figure 8: BSP Iterative Model Refinement Workflow

The models are first built by allowing the User to configure and run the model building tools of interest. Some model builders will execute in minutes. Others will take days of supercomputer time. Throughout the build process, the BSP analysis and validation tools are available at researchers' fingertips.

After the various levels of models have been built, they can be exported to run on their respective simulators. The BSP will provide infrastructure for the configuration and deployment of simulations on supercomputers and the launching of those simulations from the Collaboratory Web GUI and service APIs. The results will be visible in the Collaboratory and can then be analysed using the analyses integrated into the Collaboratory from the Neuroinformatics, Brain Simulation and HPC platforms. The analysis results can then be used for publishing new papers or to provide for the next Iteration of the modelling refinement process.

The Use Cases below describe success scenarios for small groups of scientific Users. The scenarios describe high-level interaction with the Brain Simulation Platform and are dependent on functionality provided by many of the BSP components and the BSP dependencies. The process below frequently specifies the circuit model in the Use Case. This is intended to be illustrative. The Use Cases below apply to other brain component models as well.

3.2.1 Brain Builder (SP6BSP-UC-001)

Abigail wants build a Compound model or Model component.

Primary Actors: A Scientific User, Abigail.

Success Scenario:

1) Abigail selects an existing COLL Project to work within or creates a new COLL Project.
2) Abigail selects the build for the model type she wants to build.
3) The builder Task guides her through the input selection steps using content typed inputs to filter the list of valid inputs, using previous configurations as initial defaults.
4) Abigail executes the Model Build Task as a Job and is informed when her Job is complete.
5) A minimal set of validation Tasks is run.

3.2.2 Validations (SP6BSP-UC-002)

Abigail wants to validate a circuit model she finds in the COLL.

Primary Actors: Two Scientific Users, Abigail and Bill.

Preconditions:

- A circuit model has been built as in SPBSP-UC-001.

Success Scenario:

1) Abigail wants to validate the circuit she has found in the COLL or has built herself.
2) A minimal set of validations has already been run as part of the build process. She reviews these and decides that several more computationally costly validations are required.

3) Abigail selects and runs validations the have been filtered for applicability to the model content type she has selected.
4) Validation Tasks output standardised validation reports, which are added to the list of all validations that have been run on the circuit.

3.2.3 Compound Model and Model Component Analysis (SP6BSP-UC-003)

Abigail wants to perform a simulation on a circuit model that she finds in the COLL.

Primary Actors: Two Scientific Users, Abigail and Bill.

Preconditions:

- A circuit model has been built as in SPBSP-UC-001.

Success Scenario:

1) Abigail finds a released brain or brain component model in the COLL using the integrated search functionality.
2) She would like to review some of the analyses that have been run on this circuit model.
3) She uses the COLL Project Viewer to show her which analyses and have been run and their results.
4) Abigail would like to see a pathway connectivity report, but one has not yet been generated. She can select the Pathway Connectivity Report Task and launch it as a Job. She will be notified when the analysis completes.

3.2.4 Simulation Configure and Launch (SP6BSP-UC-004)

Abigail wants to perform a simulation on a model she finds in the COLL.

Primary Actors: Two Scientific Users, Abigail and Bill.

Preconditions:

- A circuit model has been built as in SPBSP-UC-001.

Success Scenario:

1) Abigail finds a released brain or brain component model in the COLL using the integrated search functionality.
2) Abigail adds the model to a new COLL Project, COLL Project 5.
3) Abigail selects which type of simulation she would like to perform on the model from three options: Molecular, Cellular or Network. In this particular case she chooses the Cellular simulator as her target.
4) The selection causes the Platform to open a Cell Experiment builder that allows Abigail to configure whichever parameters the Experiment builder supports.
5) Abigail then launches the simulation Task as a Job to run an HPC Platform compute resource. During the launch process, she may be asked (depending on the Task definition) to make a decision about which computing resource to run on and where to store final output data. The HPC Platform could provide recommendations based on data locality, data transfer volume and compute resource availability.

6) Once it is complete, the Job registers some or all of its output (as determined by the Task definition) from COLL Project 5 in the COLL. The output may be marshalled to its final storage location by the HPC Platform Data Transfer service.

3.2.5 Simulation Analysis Tools (SP6BSP-UC-005)

Abigail wants to analyse the results of a simulation that she previously ran in the COLL.

Primary Actors: Two Scientific Users, Abigail and Bill.

Preconditions:

- A circuit model has been built as in SPBSP-UC-001.

Success Scenario:

1) Abigail adds Bill to the COLL Project with write permissions to help her analyse the output data using analysis Tasks available for simulations in the COLL.
2) Bill can now run analysis Jobs on the simulation.
3) Analysis output will be associated with the analysed simulation in the COLL Project view.

3.2.6 Collaborative Review Process (SP6BSP-UC-006)

Abigail wants to review the results of her model building and simulations with additional experts prior to releasing.

Primary Actors: Three Scientific Users, Abigail, Bill and Chris.

Preconditions:

- A model has been built as in SPBSP-UC-001, validated as in SPBSP-UC-002, and analysed as in SPBSP-UC-003. The model has had simulations run on it and the simulation results have been analysed.
- The model and simulations have all been done in the same COLL Project.

Success Scenario:

1) Abigail and Bill have a teleconference to discuss the results of the latest model refinement. Abigail and Bill have already run their analysis and both visit the Portal to review the results together.
2) Abigail and Bill notice something in the simulation activity that they can't explain. They trace through the validations that were run on the model components and can't find a clear explanation.
3) They invite Chris into the COLL Project because they think he might have insight on the phenomenon they have observed *in silico*.
4) Chris joins the COLL Project and runs a couple of additional analyses. He explains that the observed behaviour makes sense in light of a recent paper he read and the results of the analysis.
5) Abigail is happy with the state of the model and releases it for the community to analyse and review further.

3.2.7 Additional Use Cases

The Use Cases defined in this section are only part of the Use Cases that the BSP is meant to address. For additional Use Cases see the respective BSP component sections:

- Brain Builder: Use Cases
- MolSim: Use Cases
- CellSim: Use Cases
- NetSim: Use Cases.

3.3 BSP: Functional Requirements

The BSP inherits all Functional Requirements of the Collaboratory. In addition, the following Functional Requirements apply:

3.3.1 Brain Builder (SP6BSP-FR-001)

1) The COLL allows configuration and execution of Builders for all relevant model types through the standard COLL Task model. Target builders will be driven by interaction with the Initial Brain Modelling teams.

3.3.2 Analysis and Validation Tasks (SP6BSP-FR-002)

1) Common Analysis and Validation Tasks will be integrated into the COLL. This list (which will grow continuously over the lifetime of the Portal) includes:
   a) Spike Raster plots
   b) Peristimulus Time Histogram or PSTH Plot
   c) Connectivity analysis
   d) Collage plots of randomly selected cell morphology placements
   e) Collage plots of randomly selected cell electrical behaviour.
2) The Collaboratory has content type-specific editor modules to allow the generic Task Web GUI to be sufficiently user friendly.

3.3.3 Simulators – MolSim, CellSim and NetSim (SP6BSP-FR-003)

1) Simulators must be integrated with the Task and Workflow Model of the COLL. Each simulator will have at least one COLL Task that makes it possible to launch the simulator from the Web GUI or the programmatic REST Service API.
4) The BSP must provide at least one specialised BSP Configuration Web GUI per simulator for configuring the most relevant simulation parameters. These are called the Experiment Builders. There will likely be more than one per simulator as the simulators evolve and the configuration options evolve:
   a) Molecular Experiment Builder: Configuration system for MolSim
   b) Cellular Experiment Builder: Configuration system for CellSim
   c) Network Experiment Builder: Configuration system for NetSim.

5) See the Simulator Component Requirements for more details:
   d) MolSim: Functional Requirements
   e) CellSim: Functional Requirements
   f) NetSim: Functional Requirements.

3.3.4 COLL Projects and COLL Interaction (SP6BSP-FR-004)

- All outputs of Brain Simulation Platform Tasks and Workflows are saved into the User's current COLL Projects or a User specified location.
- All inputs to the Brain Simulation Platform Tasks and Workflows are loaded from URLs residing in COLL Projects or a User specified location.

3.3.5 Use Case Mapping

The table below indicates which Functional Requirements are used to satisfy each Use Case. An X in a (row, column) below indicates that the Functional Requirement in the column is necessary to satisfy the Use Case in the row.

SP6BSP-FR-XXX	001	002	003	004
SP6BSP-UC-001	X			X
SP6BSP-UC-002		X		X
SP6BSP-UC-003		X		X
SP6BSP-UC-004			X	X
SP6BSP-UC-005		X		X
SP6BSP-UC-006				X

Table 3: BSP Use Case to Requirement Mapping Table

3.4 BSP: Non-Functional Requirements

The BSP inherits all Non-Functional Requirements of the Collaboratory.

3.5 BSP: Architectural Overview

This section describes the components that make up the Brain Simulation Platform. The components are implemented as Collaboratory App, Task or Foundation Software components, and make extensive use of the HPC Platform (HPCP), and NIP service offerings. In some cases below, the complexity of the listed components requires that they be described in detail in their respective Component sections.

- BSP Tasks — a collection of BSP software components that can be invoked through a Web API, and provide model building, analysis and validation capabilities.

- BSP Apps — a collection of App components provided as a Web based GUI for BSP Tasks.
- BSP Foundation Software — a collection of BSP libraries for working with model and simulation data.
- Brain Builder.
- Molecular Simulator — specified in the Molecular Simulator section of this document.
- Cellular Simulator — specified in the Cellular Simulator section of this document.
- Network Simulator — specified in the Network Simulator section of this document.
- Initial Brain Models — specified in the Initial Brain Models section of this document.

3.5.1 BSP Tasks

Central to the task of Brain Simulation is one series of tools that are used to create models, and another set of tools that perform the actual simulation. At each stage of the process, the idea is to output files that can be read by the next stage of the process, to create a decoupled system where any piece can independently be re-implemented.

Within the context of the COLL Task Framework, an initial set of Tasks are provided to perform common analysis, circuit building and validation operations. The groups and specific Tasks presented here are expected to be extended as collaboration with scientific contributors within the Consortium develops.

3.5.1.1 Post Circuit Building Tasks

Once a circuit is built, a set of validation and analysis tasks is run against it.

3.5.1.1.1 Circuit Validation Tasks

A validation Task checks that the circuit is consistent with a set of biological data.

The development of a validation Task is facilitated by the use of a validation toolkit library. This is intended to make it easy for the developer to provide the data to be validated against, and the validation logic.

The result of a validation is a boolean value. This states whether the circuit passed or failed the validation, and gives a validation report that provides more details. A default viewer is provided along with the COLLUP, and displays the report in a comprehensive manner for the end users.

The following circuit validation Tasks are provided:

- Cell density per layer validation
- Intrinsic + extrinsic synapse densities per layer validation
- Inhibitory synapse densities validation
- Spine length distribution validation
- Synapse count validation
- Synapses overall density validation
- Somata volume fraction validation
- GABAergic cell fraction per layer validation.

3.5.1.1.2 Circuit Analysis Tasks

A circuit analysis Task provides insights to the reconstructed circuit.

The Brain Simulation Platform provides the following circuit analysis Tasks:

- Layer analysis — generates a plot of the layers of the circuit. Only relevant for cortex circuits.
- Geometry analysis — generates a plot of the mosaic geometry of the circuit.
- Morphology collage tasks — generate a projection plot of random morphologies for each layer and morphological type.

3.5.1.2 Post Simulation Tasks

Executing a simulation against a circuit generates different reports that capture the value of some of the model variables, such as voltages and currents. Post simulation Tasks help to analyse and summarise these reports. The results of these analyses are visible through the COLL.

- Spike raster analysis — generates a plot that graphically shows a tick at the time a spike was present.
- Peristimulus Time Histogram analysis — generates a histogram of the times at which a neuron fired.
- Voltage Collage analysis — generates a plot showing the times between spikes.

For the convenience of scientific users, these Tasks, and other analysis Tasks, can be bundled together and run semi-automatically once the results of a simulation have been registered with the Platform.

3.5.1.3 Other BSP Tasks

- Axon splicing Task — reconstructs a set of neurons by replacing their axon with the axon provided in input.
- Axon splicing analysis Task — generates a projection view of the neuron reconstructed by the axon splicing Task.

3.5.2 BSP Apps

In most cases, BSP Apps will use the Tasks' automatic Web GUI functionality. As scientific Use Case priorities are more clearly identified, custom GUIs will be used to control the underlying BSP Task components. The Collaboratory section on the Task Framework component contains further information on COLL Task Framework automatic GUI generation.

3.5.3 BSP Foundation Software

Having a stable software foundation is critical for the neuroscience community to exchange data and ideas. Thus, the following section will outline two important workflows for model building and simulation, and provide an outline of the software needed to run from start to finish.

The overview of the particular workflows presented here corresponds to implementations developed within the BBP group. A number of these workflows are described in more detail in the Brain Builder section of this document. Collaboration within the Consortium and the

larger community will allow other members to provide and use their own their own tools and workflows in place of the current versions.

This section also presents a sample from the BSP Software Foundation (BluePy) and presents its high-level data dependencies. Further, it specifies a single selected data dependency at the file format level. This is meant as an example of ongoing work to bring the BSP Foundation Software up to a sufficient quality for it to readily adopted by the larger neuroscience community. In this way, the tools and formats of the Brain Simulation Platform will serve as emerging standards for high-performance, HPC-ready, data-driven model and simulation formats.

3.5.3.1 Major Workflows

3.5.3.1.1 Brain Building

Before a model can be simulated, it has to be built. This means, for instance, putting reconstructed or synthetically grown morphologies in place, and recreating potential synapses that would appear in nature. The following high-level workflow classifies the different tools used by the BBP to build a detailed reconstructed circuit, according to their primary use (many are used in multiple parts of the workflow):

1) Morphology reconstruction, electrical model parameterisation and population construction:
    a) MorphologyRepair — collection of workflows to turn populations of reconstructed neuron morphologies into larger populations of statistically accurate morphologies.
    b) BlueRepairSDK — collection of tools to repair artefacts of the reconstruction.
    c) MUK — a collection of tools to perform cleanup and analysis of reconstructed neuron morphologies.
    d) Pneumatk — Python morphology analysis library.
    e) eFEL — a BBP open source library for extracting electrophysiological features from electrophysiological recordings.
    f) BGLibPy — a Python productivity interface on top of Neuron.
    g) ModelManagement — a workflow for extending etype channel distributions to large morphology libraries and validating the resulting metype models.
    h) OptimizerFramework — C++ multi-objective optimisation (currently used for etype channel fitting).
2) Cell Placement
    a) AxonSplicing & AxonSynthesis — a workflow for synthesising statistical axon populations and splicing them onto metype models in an existing cellular-level microcircuit model.
    b) Mesobuilder — a command-line automation system for managing the cellular-level model reconstruction workflow.
    c) Connectome prediction.
    d) Touchdetector — large-scale structural synapse creation.
    e) Functionaliser — large-scale conversion of structural synapses to functional synapses.

3) Validation and Analysis
   a) Brion — C++ IO library for common circuit and simulation data objects.
   b) BBPSDK — a C++ object model for common circuit and simulation data objects.
   c) BluePy — a Python productivity suite and object model for common circuit and simulation data objects.
   d) Validation Toolkit — object model for structured statistical validation and standardised reporting.

3.5.3.1.2 Simulation

After a model has been built, it can be simulated with the aim of producing relevant predictions about its behaviour. The workflow proceeds in two phases, described below:

1) Simulation run:
   a) STEPS — opensource molecular-level simulator.
   b) NEURON — opensource cellular-level simulator.
   c) NEST — opensource network-level simulator.
   d) Reportinglib/pybinreports — C++ and Python access libraries for parallel electrical reporting from large scale Neuron simulations.
   e) CoreNeuron/Bluron/Neurodamus — variants of the Neuron simulator for specific HBP simulation use cases.
2) **Simulation analysis**: Once a simulation has been run, scientists perform validation and analysis by using tools such as:
   a) reportinglib/pybinreports — C++ and Python access libraries for parallel electrical reporting from large scale Neuron simulations.
   b) *BluePy* — described in a previous section is described in more detail below.

3.5.3.2 BluePy Data Specification

BluePy is a library for accessing various objects in the circuit models and simulation reports. It contains parsers for various data formats used within the Project, as well as tools for computational neuroscientists who wish to analyse and validate circuit properties and simulation results.

It is expected that BluePy will be open-sourced for open development in the near future. It is considered to be one of the cornerstones of analysis and validation of cellular-level models produced by the Brain Simulation Platform.

Note: the name BluePy is likely to change prior to release as an opensource software package.

BluePy interacts with the following data formats, which are produced and consumed in various parts of the cellular model building and simulation pipeline:

- BlueConfig — simulation configuration file.
- CircuitConfig — circuit configuration file, a strict subset of the BlueConfig.

- start.target / default_user.target / start_meso.target — definitions of groups of cells used for analysis and circuit building software components.
- circuit.mvd2 / circuit_mvd2.sqlite — metadata regarding cells in a circuit, ASCII and SQLite DB form respectively.
- start.ncs — legacy simulation output data.
- Morphologies — H5, SWC and ASC files describing the points and connections of an individual neuron.
- nrn.h5 files — final synapse parameters, for pre- and post-synapses.
- Spatial Index — "novel spatial access methods ... [exploiting] connected spatial objects [to] enable an additional means of accessing data in spatial proximity"[1].
  - SYNAPSE_payload.dat / SYNAPSE_index.dat / SYNAPSE_index.idx
  - SEGMENT_payload.dat / SEGMENT_index.dat / SEGMENT_index.idx
- XML recipe files — method of describing certain parameters of a circuit design.
- miniHints.mhf — file containing minicolumn positions to be avoided from other meso-scale columns.
- out.dat — output format describing the spikes that have been generated, and the time at which they were generated.

All of these formats have conventions surrounding their locations in a given file hierarchy. These conventions will be formalised as standards in later versions of BluePy.

3.5.3.2.1 BluePy Example File Format

One example of the file format used within the Project is mvd2. It is used to collect and transmit metadata regarding cells in a circuit to the various circuit building, simulation and analysis portions of the workflow. As the Project continues, more file formats will be documented, leading to easier collaboration between different groups.

The MVD file format is an ASCII file containing a series of sections. Within each section, a series of section dependent rows describe the necessary metadata for the smooth interchange of data between software tools.

Invariants:

- Only a single section of each type can exist.
- Lines starting with # are assumed to be comments.

The following sections exist:

- **Neurons Loaded**: one neuron per line, containing:
  - Morphology name (string without spaces)
  - Database type [not used] (int)
  - HyperColumn (int)
  - MiniColumn (int)
  - Layer [note that 0 is layer 1, 1 is layer 2, etc.] (int)

- Morphology type [index into MorphTypes below] (int)
- Electrophysiology type [index into ElectroTypes below] (int)
- NeuronCenter X (float)
- NeuronCenter Y (float)
- NeuronCenter Z (float)
- NeuronRotation (float) rotation in the Y plane
- ME-type (string).

```
EX: dend-tkb071119a1 0 0 1 5 1 0 13.624802 1266.300928 1265.016762 0.0
cADpyr231_L6_BPC_6_dend-tkb071119a1
```

1) **MicroBox Data**
   - Column Size x/y/z coordinates (all float)
   - Layer 6 percentage in the column (float)
   - Layer 5 percentage in the column (float)
   - Layer 4 percentage in the column (float)
   - Layer 3 percentage in the column (float)
   - Layer 2 percentage in the column (float).

EX: 461.840 2006.348 399.960 33.627 25.230 9.109 16.958 7.151

1) **MiniColumnsPosition**
   - X coordinates (float)
   - Y coordinates (float)
   - Z coordinates (float).

EX: 233.161 1003.174 204.186

1) **CircuitSeeds Seeds**

Needed for pseudo-random number generation.

EX: 5853043.000000 3486991.000000 8465746.000000

2) MorphTypes: one row per morphology type, containing:
   - -MorphologyName (string)
   - -PYR or INT (string)
   - -EXC or INH (string).

EX: L1_DAC INT INH

3) ElectroTypes: one row per electrical type:
   - ElectrophysiologyName (string).

3.5.4 Functional Requirements on other Platforms

Use Case	External Platform Dependencies
SP6BSP-UC-001	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements
SP6BSP-UC-002	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements
SP6BSP-UC-003	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements
SP6BSP-UC-004	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6.
SP6BSP-UC-005	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements
SP6BSP-UC-006	COLL Functional Requirements 1.3.1 and 1.3.4

Table 4: BSP Functional Requirements on other Platforms

3.6 BSP: Relations to other Platforms

3.6.1 Services Provided to other Platforms

The services below are actually provided by the subcomponents of the BSP, but they are listed here for the sake of completeness:

- Simulation configuration
- Compound model and model component configuration and building
- Model analysis and validation
- Simulation analysis and validation.

3.7 BSP: Dependencies

The Biological Reconstruction Process yields instance models of the brain given a partial and flexible data set as the input (i.e. a snapshot of a brain at an instant). To reconstruct the brain at different ages, genders, genetic variations, conditions and even different species, the specific and relevant data sets are used as the input (akin to a "configuration file"). The Neuroinformatics Platform will enable access to data on genetic variations, ages, genders, and under different conditions, and specific data sets will be used to reconstruct specific instances of the brain. By using disease signatures obtained in the Medical Informatics Platform, brain models of disease, fully consistent across the levels of biology, will be reconstructed for simulations of brain diseases.

The Theory subproject will ensure the mathematical and theoretical foundations of the reconstruction and validation process and the High Performance Computing Platform will develop the data-intensive, multi-scale and interactive capability and capacity requirements of simulating the human brain. The Neurorobotics Platform will develop the interface technologies to couple brain models to "bodies" (virtual agents) using closed loop technologies and the technologies for studying cognition and behaviour in virtual environments. The Brain Simulation SP will yield circuits of the brain with unprecedented fidelity and the Theory SP will carry out the research to simplify these models and develop a multi-scale theory for the brain. The multi-scale theory will allow multi-scale simulations to pragmatically simulate the brain across all levels, but with greater fidelity as supercomputing capability increases. The simplified models will be implemented into neuromorphic architectures in SP9 to explore parameters spaces, learning and memory and develop products and services for a new generation of devices.

3.7.1 Required

For a list of non-service dependencies, see the dependencies of the BSP components listed in the previous components section, 3.5.1. Additionally the following developments are required to meet the objectives of the Brain Simulation Platform:

- User-contributed analysis and validations
- Validation-ready data have been contributed to the NIP
- Ontology development in the NIP has produced usable ontologies which cover all terminology needed in the BSP.

3.7.2 Preferred

For a list of non-service activities that will be necessary to achieve the full potential of the Brain Simulation Platform, see the dependencies of the BSP components listed in the previous components section, 3.5.1.

4. BSP: Brain Builder (BB)

One strategy to obtain a multi-level understanding of the brain is to exhaustively study every biological parameter (component) and variable (interactions, functions) at every level of brain organisation (genes, proteins, biochemical, metabolites, ions, subcellular organelles, cells, synapses, microcircuits, brain regions, whole brain, cognition, behaviour) and then to obtain such maps for all ages, genders, genetic and epigenetic variations of multiple species. Then, one would develop a model and/or theory of how all the known pieces fit together to explain the detailed workings of the brain. Even without considering contextual, environmental and social influences, it is obvious that experimentally mapping the brain in this manner is impossible for the foreseeable future. Even if one brain could be fully mapped, it would not provide the biological ranges for parameters and variables even for that instant.

This intractable barrier raises the question: is it necessary to map it all? One argument is that we do not need to map it all; rather, we should focus on searching for principles and developing theories of brain function. However, this approach cannot fully explain how the detailed multi-level structure of the brain (the machinery) gives rise to function and has limited value in explaining how diseases may arise and how drugs exert their effects by acting at the molecular level. Another question then arises: can we obtain complete biological maps of the brain under any condition and, in principle, for any species, given that the experimental data and knowledge are very sparse and fragmented?

The Brain Simulation SP is building on the work of the Blue Brain Project, which pioneered predictive neuroscience. Predictive neuroscience identifies and leverages highly interdependent data within and across levels of organisation. Interdependencies imply that the knowledge of any one biological parameter can be exploited to predict other data parameters, rather than just to measure them. Interdependencies are described as principles of organisation and implemented in algorithms that computationally reconstruct structurally and functionally accurate digital computer models of the brain. Each reconstruction step creates a hypothesis of the brain, pragmatically integrating available data and knowledge. The hypothesis is tested. Verifying or falsifying any aspect of the model (the hypothesis) guides new experiments, and challenges and refines our understanding of the principles of organisation. In this way, each reconstruction step generates new scientific knowledge. This approach yields a quantitative measure of the value of any biological parameter, hence the importance of experimentally obtaining a specific data set. The value is directly proportional to the number of interdependencies that can be identified. Assessing the value of data sets helps focus limited experimental resources to fill strategic aspects of the sparse map of the brain, thus allowing completion of the map via predictions.

To implement the data-driven algorithms and workflows needed to reconstruct multi-level brain models, the BSP will develop and deploy a Brain Builder. The Brain Builder implements the bottom-up predictive strategy of reconstructing brain models, based on data and principles for any instance. This section presents a specification of the Brain Builder Component of the BSP.

4.1 Brain Builder: Overall Goals

The approach adopted by the Brain Simulation SP is fundamentally different from a hypothetical approach of blindly putting the pieces together and expecting to understand

what has been constructed. It is also fundamentally different from modelling the brain top-down, based on an over-arching theory of the brain. Each organising principle is rather a "micro" theory of the brain and a bottom-up strategy is followed. This bottom-up approach is completely missing in neuroscience, while theory-based top-down modelling is well established. In adding this new strategy, the Brain Simulation SP will play a unique role in the HBP by connecting the levels of biology to achieve a multi-level understanding of the brain.

The core strategy pursued by the Brain Simulation SP is defined as a data-driven, algorithmic reconstruction of the brain. We call this a Biological Reconstruction Process. We have developed a theoretical framework that provides guidelines for rigorous validation and systematic refinement of reconstructed brain models towards biological accuracy. The Brain Simulation SP will extend this strategy from reconstructing neurons and microcircuits to reconstructing brain regions, brain systems and the entire brain. The multi-level structural data on the mouse brain provided by the Core Projects, and the functional data on the mouse provided by the Partnering Projects in SP1 will provide the input data set to fully develop the Biological Reconstruction Process for whole brains down to the molecular level. Looking forward to the Operational Phase of the HBP shows how the BSP must evolve. The main Deliverable of the first five years of the Operational Phase is a first draft reconstruction of the mouse brain (i.e. by 2018). The Platform will enable continual community-driven refinement. The multi-level structural data on the human brain provided by the Core Projects and the functional data provided by the Partnering Projects in SP2 will provide the human-specific data set to apply this strategy to reconstructing the human brain. The main Deliverable in the second five years of the Operational Phase is a first draft reconstruction of the human brain (i.e. by 2023).

The Brain Builder (BB) is the umbrella component for the model building functionality in the BSP. The BB is intended to facilitate collaborative model building by integrating model building workflow requirements from around the HBP, enhancing them for reproducibility, maintainability and usability. In this way, the BSP will evolve as an ecosystem for software and tool development, data sharing, and algorithmic and model development throughout the HBP. As this ecosystem is opened up to the community, data sharing and algorithmic development by experimentalists can be accelerated to reconstruct more accurate models, enrich the models with new data, refine the models as they are collaboratively challenged and validated by biologists, simulate the models and analyse the results, and develop models that can be exploited for the development of computing and medical technologies.

4.2 Brain Builder: Use Cases

The Brain Builder Use Cases revolve around the building of components in the Initial Brain Models. More specifically, they are based on the generic modelling strategies presented in Section 7.2 and Section 7.3:

- Molecular – extraction of electrical behaviour and geometry from an existing cellular model and populating it with synthesised molecular level models.
- Cellular – integration of reconstructed neurons into large populations of me-types instances. See 7.3.2.3 items i)-v).
- Microcircuit – construction of a microcircuit from populations of me-types. See 7.3.2.3 items vi)-viii).

- Mesocircuit – construction of a mesocircuit, on the scale of a single brain region, from populations of me-types. See 7.3.2.3 items vi)-viii).
- Macrocircuit - connecting brain regions, an active research topic. Availability depends on the progress of model refinement and data availability for multi-region models.

It is expected that the builders will evolve rapidly as the models incorporate new data.

4.2.1 Repair and Diversification of Reconstructed Morphologies (SP6BSP-UC-007)

Primary Actors: Scientific User, Abigail.

Preconditions:

- A reconstructed morphology library of sufficient diversity is available for the species, age and brain region of interest.
- Validation Tasks for reconstructed morphologies have already been written to validate that an arbitrary morphology instance in a predefined morphology class is correct.

Success Scenario:

1) Abigail wants to generate a large collection of morphologies based on repaired and diversified reconstructed morphologies. These morphologies will populate a detailed circuit model in a particular region.
2) She selects the Morphology Release Task and builds her set of input morphologies from those available in the Neuroinformatics Platform.
3) Abigail then launches the Morphology Release Task as a Job.
4) Once the portal notifies Abigail that the Job is complete, Abigail can run the Morphology Validation suite to verify that the morphologies match all expected biological characteristics.

4.2.2 Synthesise Full Cell Morphologies (SP6BSP-UC-008)

Primary Actors: Scientific User, Abigail.

Preconditions:

- A reconstructed morphology library of sufficient diversity is available for the species, age and brain region of interest.
- Validation Tasks for reconstructed morphologies have already been written to validate that an arbitrary morphology instance in a predefined morphology class is correct.
- The reconstructed morphology library for the area of interest has been analysed and the cell synthesis parameter sets have been extracted.

Success Scenario:

3) Abigail wants to synthesise cell morphologies to test that synthesis is working properly.
4) She selects the cell synthesis configuration tool and searches the Neuroinformatics Platform for synthesis configurations for the appropriate species, brain region, age, etc.
5) Abigail then launches the Cell Synthesis Job.

6) Once the Portal notifies Abigail that cell synthesis is complete, Abigail can run the Morphology Validation suite to verify that the morphologies match all expected biological characteristics.

4.2.3 Create a Complete Cell Model Using Automated Fitting of Conductance Densities (SP6BSP-UC-009)

Primary Actors: Scientific User, Bill.

Preconditions:

- A morphology that has passed the Morphology Validation Suite.

Success Scenario:

1) Bill wants to create a complete Cell model from a validated morphology.
2) Bill selects an exemplar morphology from a list of previously validated neuron morphologies. The exemplar is used as input to the automated conductance density-fitting tool, and Bill configures the tool with the correct optimisation settings.
3) Bill submits the configuration for execution as a Job.
4) When the Job is complete, an electrical-type template is produced. The electrical-type template can be considered validated, if the automated conductance density fitting tool functions. As a result, no further validation of the electrical type template is necessary.
5) Bill applies his electrical type template to candidate morphologies to validate that the morphological and electrical type combinations are valid. This process produces combined morphological and electrophysiological models known as METypes. Validated METypes can now be used standalone or in cellular level model building.

4.2.4 Distribute Cells and Use this to Create a Point Neuron Model of a Brain Region (SP6BSP-UC-0010)

Primary Actors: Scientific User, Bill.

Preconditions:

- Volumetric data sets in the correct reference space for the brain region of interest have the required data components:
  - Excitatory-inhibitory ratio
  - Cell density
  - M-type ratio.
- Intraregional connectivity density maps are available for the region in question.

Success Scenario:

1) Bill would like to build a point neuron model of a brain region.
2) Bill selects volumetric inputs for Excitatory-inhibitory ratio, Cell density, M-type ratio and connectivity ratios.
3) Bill executes the point neuron region builder workflow Task as a Job.

4) When the Job completes, Bill can use BSP Tasks to analyse and validate the completed circuit.
5) Bill can also use the BSP Network-level simulation configuration system to configure and launch simulations.

4.2.5 Distribute Cells and Use this to Create a Point Neuron Model of a Whole Rodent Brain (SP6BSP-UC-011)

Primary Actors: Scientific User, Bill.

Preconditions:

- Volumetric data sets in the correct reference space for the brain type of interest have the required data components:
  - Excitatory-inhibitory ratio
  - Cell density
  - M-type ratio.
- Intraregional connectivity density maps are available for the whole brain.

Success Scenario:

1) Bill would like to build a point neuron model of a whole brain.
2) Bill selects volumetric inputs for Excitatory-inhibitory ratio, Cell density, M-type ratio and connectivity ratios.
3) Bill executes the point neuron whole Brain Builder workflow Task as a Job.
4) When the Job completes, Bill can use BSP Tasks to analyse and validate the completed circuit.
5) Bill can also use the BSP Network Experiment Builder to configure and launch simulations.

4.2.6 Distribute Cells and use this to Create a Detailed Neuron Model of a Rodent Neuronal Microcircuit (SP6BSP-UC-012)

Primary Actors: Scientific User, Bill.

Preconditions:

- A detailed Cellular-level microcircuit recipe is available for a rodent in the brain region of interest. The data required here is discussed in more detail in the Cellular Model portion of this document.

Success Scenario:

1) Bill would like to build a detailed neuron microcircuit model in a particular rodent brain region. The microcircuit would represent a small portion of the neuron population of the full region.
2) Bill selects the species, age, region, etc. where the microcircuit will be built. This defines the ontological model context. In this case he will use a preferred species, but the other pieces of model context could be selected based on availability of data.

3) Bill selects a previous microcircuit recipe for the selected model context.
4) Bill executes the microcircuit builder workflow Task as a Job.
5) When the Job completes, Bill can use BSP Tasks to analyse and validate the completed microcircuit.
6) Bill can also use the BSP Cellular level simulation configuration system to configure and launch simulations using the microcircuit model.

4.2.7 Simplify the Cellular Level Model to a Network Level Model (SP6BSP-UC-013)

Primary Actors: Scientific User, Bill.

Preconditions:

- A detailed Cellular-level microcircuit, brain region circuit or multi-region circuit.

Success Scenario:

1) Bill selects a Cellular-level microcircuit, brain region circuit or multi-region circuit of interest.
2) Bill selects the Network-level model export Task.
3) Bill configures the Network-level model export Task.
4) Bill runs the Network-level model export Task as a Job.
5) Upon completion, the Job deposits the exported Network-level simulation in the same COLL Project as the source Cellular-level model.

4.2.8 Export a Volume Region of the Cellular Level Model and Add Molecular Level Detail to Produce a Molecular Level Model (SP6BSP-UC-014)

Primary Actors: Scientific User, Bill.

Preconditions:

- A detailed Cellular-level microcircuit, brain region circuit or multi-region circuit.

Success Scenario:

1) Bill selects a Cellular-level microcircuit, brain region circuit or multi-region circuit of interest.
2) Bill selects the Molecular-level model export Task.
3) Bill selects a sub-region of space in the Cellular-level model to export.
4) Bill configures the Molecular-level model export Task.
5) Bill runs the Molecular-level model export Task as a Job.
6) Upon completion, the Job deposits the exported Molecular-level simulation in the same COLL Project as the source Molecular-level model.

4.3 Brain Builder: Functional Requirements

The Brain Builder inherits all Functional Requirements of the Collaboratory.

4.3.1 Tasks (SP6BSP-FR-005)

The Brain Builder will provide at least one version of the following Tasks in the Collaboratory Task Repository:

1) Repair and diversification of reconstructed morphologies.
2) Synthesise full cell morphologies.
3) Create a complete Cell model using automated fitting of conductance densities.
4) Distribute cells and use this to create a point neuron model of a brain region.
5) Distribute cells and use this to create a detailed neuron model of a rodent neuronal microcircuit.
6) Simplify the Cellular level model to a Network level model.
7) Export a volume region of the Cellular level model and add molecular level detail to produce a Molecular level model.

4.3.2 Configuration GUIs (SP6BSP-FR-006)

The Brain Builder has configuration GUIs for all of the Tasks defined in 3.3.1.

4.3.3 Neuroscientific Semantic Database Integration (SP6BSP-FR-007)

The Brain Builder will rely on data, vocabularies, and classifications from the Neuroinformatics Platform wherever possible. Integration work will be required to make this a seamless process for COLL Developers and Users.

4.3.4 Use Case Mapping

The table below indicates which Functional Requirements are used to satisfy each Use Case. An X in the matrix below indicates that the Functional Requirement in the column is necessary to satisfy the Use Case in the row.

SP6BSP-FR-XXX	005	006	007
SP6BSP-UC-007	X	X	X
SP6BSP-UC-008	X	X	X
SP6BSP-UC-009	X	X	X
SP6BSP-UC-010	X	X	X
SP6BSP-UC-011	X	X	X
SP6BSP-UC-012	X	X	X
SP6BSP-UC-013	X	X	X
SP6BSP-UC-014	X	X	X

Table 5: BB Use Case to Requirement Mapping Table

4.4 Brain Builder: Non-functional Requirements

The Brain Builder inherits all Non-functional Requirements of the Collaboratory.

4.5 Brain Builder: Architectural Overview

This section describes the components that make up the Brain Builder and how it interacts with the other pieces of the BSP and COLL.

All of the components are accessed through a network accessible service API. However, not all components will be exposed to end-users of the Brain Simulation Portal. Some may be reserved for internal use.

4.5.1 Architectural Principles

The architecture of the Brain Builder is one that builds on the foundation of the COLL. As a result, the Brain Builder is best described as a set of extensions to the COLL to enable the various model-building workflows. A large number of the model building workflows rely on HPC Platform services to provide execution and storage services of sufficient scale. The diagram below outlines the interactions between the Brain Builder, Collaboratory and the HPC Platform.

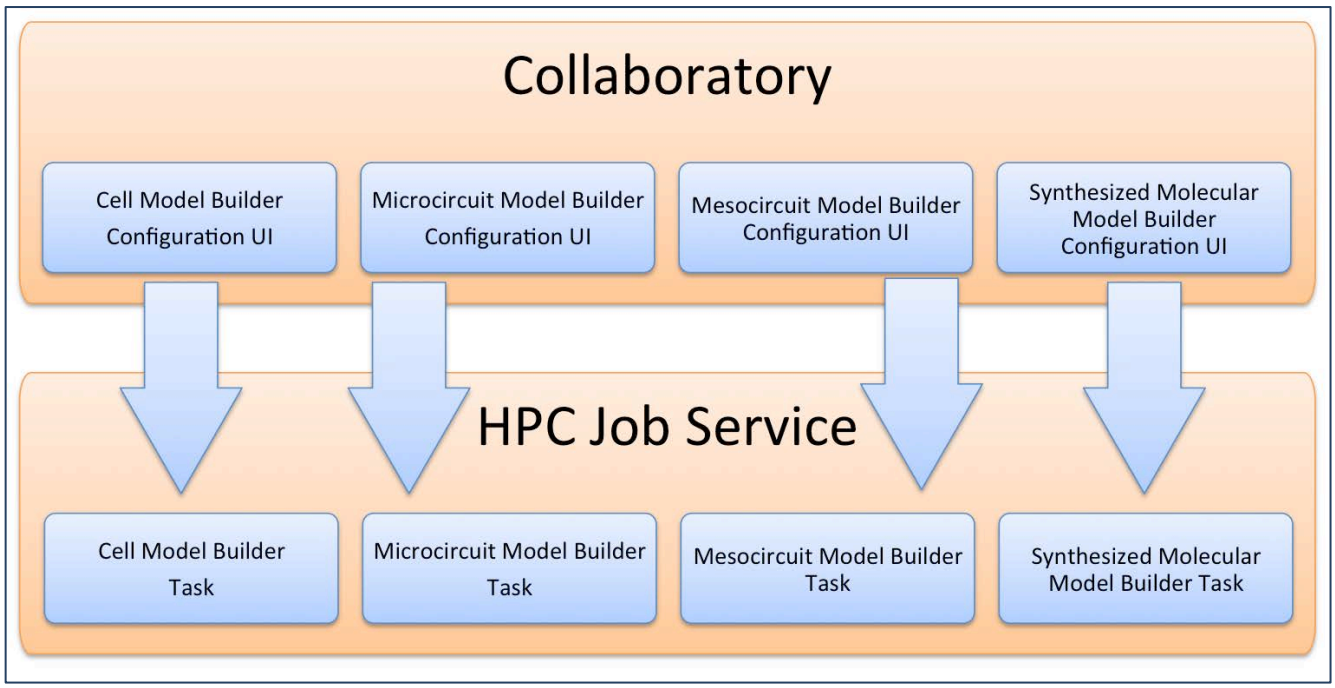

Figure 9: BB Interaction with the COLL and the HPC Platform

4.5.2 Components

The Brain Builder components will initially be based on workflows already used in the HBP.

4.5.2.1 Molecular Model Generation from Cellular Model Definitions

The existing workflow from the BBP sub-cellular simulation group extracts a region of space from the Cellular circuit model and produces a model capable of being run in the Steps simulator.

4.5.2.2 Cell Model Generation

Sub-workflows for:

1) Reconstructed morphology repair and diversification
2) E-type fitting and template generation
3) Me-type template validation.

4.5.2.3 Reconstructed Neuron Somatosensory Mesocircuit Cellular Model Building Workflow

Integration of an updated version of the somatosensory cellular model building workflow must pull reconstructed neuron data from the NIP.

4.5.2.4 Somatosensory Cellular Model Building Workflow

Building a full somatosensory cortex at the cellular model level of detail.

4.5.2.5 Neocortical Cellular Model Building Workflow

Building a full Neocortex at the cellular model level of detail. See the Cellular Model section of this document for further details on the model refinement plan.

4.5.2.6 Hippocampal CA1 Cellular Model Building Workflow

Building a full Hippocampus CA1 at the cellular model level of detail. See the Cellular Model section of this document for further details on the model refinement plan.

4.6 Brain Builder: Relations to other Platforms

4.6.1 Functional Requirements on other Platforms

Use Case	External Platform Dependencies
SP6BSP-UC-007	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements
SP6BSP-UC-008	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements
SP6BSP-UC-009	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements
SP6BSP-UC-010	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements
SP6BSP-UC-011	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements
SP6BSP-UC-012	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements
SP6BSP-UC-013	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements
SP6BSP-UC-014	Neuroinformatics Platform for semantic metadata handling and ontologies HPC Functional Requirement 1.4.2, 1.4.3, 1.4.5 and 1.4.6. All COLL Functional Requirements

Table 6: BB Functional Requirements on other Platforms

4.6.2 Services provided to other Platforms

Molecular, Circuit, and Network model configuration and building.

4.7 Brain Builder: Dependencies

4.7.1 Required

The following non-service activities are required to satisfy the minimal Use Cases of the Brain Builder.

- Brain model development has progressed to the point where a Cellular model building workflow can be built for the Cerebellum, Neocortex, and Hippocampus CA1.

4.7.2 Preferred

The following non-service activities will be necessary to achieve the full potential of the Brain Builder.

- Registration of large quantities of reconstruction source and validation data in the NIP.

5. BSP: Molecular Simulator (MolSim)

5.1 MolSim: Overall Goals

The molecular simulator is the simulation engine of the HBP for neuron and synapse models at the molecular level of resolution. The major requirement informing the design process is the ability to represent, at the given level of resolution, molecular processes occurring in neural tissue. The focus of the simulator is on detailed geometrical models of synapses and compartmental models of entire neurons and regions of surrounding space. Each of these levels of detail has a different simulation strategy, e.g. reaction-diffusion mathematics on a tetrahedral volume mesh or more abstract reaction-only mathematics for modelling well-mixed compartments. Complementary to these modelling capabilities, the MolSim component will have to meet performance requirements that allow reasonable turn-around times, and thus must be able to run on parallel architectures.

5.2 MolSim: Use Cases

This section describes Use Cases that are specific to the MolSim and the components it depends on.

5.2.1 Geometrically Accurate Synapse Model with Molecular Reactions and Diffusion (SPBSP-UC-015)

This is a more capability specific version of the Cellular level to Molecular level export workflow descripted in SP6BSP-UC-014.

Primary Actors: Scientific User, Abigail.

Preconditions:

- A neuron pair model from the Brain Simulation Platform Brain Builder with morphologically detailed neurons and synaptic locations.

- An Exporter module to extract the geometry of portions of the brain tissue model with subcellular detail.
- A database of cell-type specific molecule concentrations and reactions.

Success Scenario:

1) Abigail wants to conduct an *in silico* experiment on a molecular-level synapse between a bi-tufted cell and a pyramidal cell.
2) She chooses the appropriate synapse from the neuron pair model in the Brain Simulation Portal.
3) Abigail then opens the Cellular Experiment Builder and configures the stimulation protocol for the presynaptic neuron of interest. She furthermore selects the duration of the experiment.
4) She then launches the cellular simulation from the Brain Simulation Portal, selecting the CellSim. The voltage traces of the presynaptic neuron spiking and the post-synaptic neuron's synaptic potentials are recorded.
5) Once the Portal notifies Abigail that the cellular simulation is complete, Abigail uses the Exporter Module to extract the detailed geometry of the chosen synapses together. She is able to curate and adjust the mesh generation properties for the geometry at this stage to match electron microscopic data previously identified in the Neuroinformatics Platform.
6) Abigail configures the model with molecules; selects a subset of proteins, sets initial concentrations for each, defines the distribution of each protein, and sets the reaction kinetics between each protein and another reaction partner.
7) In the BSP, Abigail can now choose the MolSim as the target simulator for her molecular synapse and uses a subset of the presynaptic traces to stimulate the synapse and the cellular-level synaptic response as a target result.
8) A BSP application will analyse the target synaptic responses and use the features extracted (amplitudes, latencies, etc.) as a target in a multi-objective feature optimisation fitting that adjusts only the relative concentrations of the proteins until the target synaptic response is recreated.
9) Abigail now stimulates the synapse with a different subset of voltage traces from the presynaptic neuron to validate the synapse model.
10) She chooses the molecule species she would like to track and launches the MolSim from the Portal; she also chooses how many random seeds she would like to simulate for the diffusion process.
11) The Portal notifies Abigail by email/SMS when the simulations have finished. Abigail now has the opportunity to analyse the concentrations of molecules in the presynaptic terminal, in the cleft and in the synaptic spine.
12) Abigail now invites an expert in synaptic biophysics and physiology to review the results and provide further validation data if necessary.
13) Abigail releases the model for others to use and then publishes a multi-author paper describing the role of the chosen molecule in synaptic transmission.

5.2.2 Molecular Neuron Simulation using MolSim (SPBSP-UC-016)

Primary Actors: Scientific User, Bill.

Preconditions:

- A reconstructed microcircuit model from the Brain Simulation Platform Brain Builder with morphologically detailed neurons and synaptic locations.
- A database of cell-type specific molecule concentrations and reactions

Success Scenario:

1) Bill wants to study a molecular-level model of an entire neuron, stimulated with network activity drawn from a circuit simulation.
2) He chooses an appropriate neuron within a previously built circuit from the BSP. In the Molecular Model Builder he chooses to export the entire neuron volume mesh to MolSim. He has the opportunity to curate and adjust the mesh generation properties at this stage.
3) Bill then opens the Molecular Experiment Builder. He chooses a pre-run Cellular simulation run in NEURON that includes all of the other neurons in the circuit. The simulations available for selection will be only those that were run on the circuit selected in 2). This step sets the synaptic activity for the Molecular Experiment.
4) Bill selects the subsets of proteins and synapse configuration parameters to include in the simulation. These are drawn from protein libraries stored in the NIP.
5) Once the simulations are complete, the portal notifies Bill. He can then apply a range of other stimuli to study the resulting molecular interactions in an interactive analysis session on a high-fidelity cockpit.

5.3 MolSim: Functional Requirements

5.3.1 MolSim models (SPBSP-FR-008)

The Cellular-to-Molecular export workflow must be able to:

- Import compartments or volume meshes describing volumes and surfaces of the geometry on which reaction and diffusions are to take place.
- Define chemical species and reaction rates according to the chemical master equation.
- Distinguish different compartments separated by membranes.

The Molecular Simulator must be able to:

- Precisely solve the chemical master equation for low species counts.
- Accurately track volume and surface diffusion of species.

5.3.2 Recording Devices (SPBSP-FR-009)

The simulator must permit the measurement of the simulated molecular concentrations in a flexible manner. Users should be able to specify the regions in which the concentrations are recorded. The regions and identity of the molecular species will be selected.

5.3.3 Stimulation Devices (SPBSP-FR-010)

The experimental situation of providing stimulation to the molecular simulation via influx or clamping of molecules can be designated in the configuration of the initial experiment.

5.3.4 Voltage-Gated Transitions (SPBSP-FR-011)

The Molecular Simulator must have the ability to compute voltage-gated transitions of the states of membrane channels.

5.3.5 Use Case Mapping

The table below indicates which Functional Requirements are used to satisfy each Use Case. An X in the matrix below indicates that the Functional Requirement in the column is necessary to satisfy the Use Case in the row.

SP6BSP-FR-XXX	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	016
SP6BSP-UC-015		X	X	X	X	X	X	X	X	X	X	X	X	X		X
SP6BSP-UC-016		X	X	X	X	X	X	X	X	X	X	X	X	X		

Table 7: MolSim Use Case to Requirement Mapping Table

5.4 MolSim: Non-Functional Requirements

5.4.1 Parallelisation

The simulator should be capable of running simulations of large volumes of neuropil with many molecular species. For this, strong scaling is important and parallelisation of the solver is necessary.

In order to run efficiently, the simulator should be capable of balancing the simulation domain size with the communication overhead associated with a particular domain decomposition.

Parallelisation will require approximations of diffusion processes; the accuracy of this approximation should be monitored dynamically.

5.4.2 Interfaces

The full functionality of the simulator is available through the programmatic API. The interface provides access to the simulator through a Python-based API to allow simple integration into the web-based and GUI frameworks. A second Python-independent interface ensures that the simulation engine is usable on platforms that do not provide Python. This interface is compiled along with the simulation kernel and only depends on standard libraries available on today's supercomputers.

5.4.3 Memory Efficiency

Basic memory requirements can be estimated. The simulator tracks the number of molecules per tetrahedral element. We anticipate that more complicated simulations will have on the order of a million tetrahedral elements, each with 10,000 types of molecules, including their associated sub-states. Therefore, 10 billion concentration values might need to be tracked for an entire simulation. Since the entire simulation is divided among the cores of a supercomputer, which might have on the order of 10,000-100,000 cores, this works out to be only one hundred thousand to one million values tracked per core—well within the capacity of most supercomputer configurations.

Of course, a certain overhead is also required to manage the reactions as well as diffusion between tetrahedral elements. Molecules will each have a several reaction partners, and each reaction has kinetic constants associated with it. Diffusion constants per sub-state will need to be tracked.

5.5 MolSim: Architectural Overview

The MolSim component is based on the molecular simulator STEPS [2], which will be developed further in the context of the HBP to meet the overall goals of the MolSim component.

The general process of exporting and running a simulation is outlined in Figure 9. A biochemical model of chemical species, reactions, and region-specific concentrations is combined with the geometrical model. The simulation is then run using a standard simulator [3] [4].

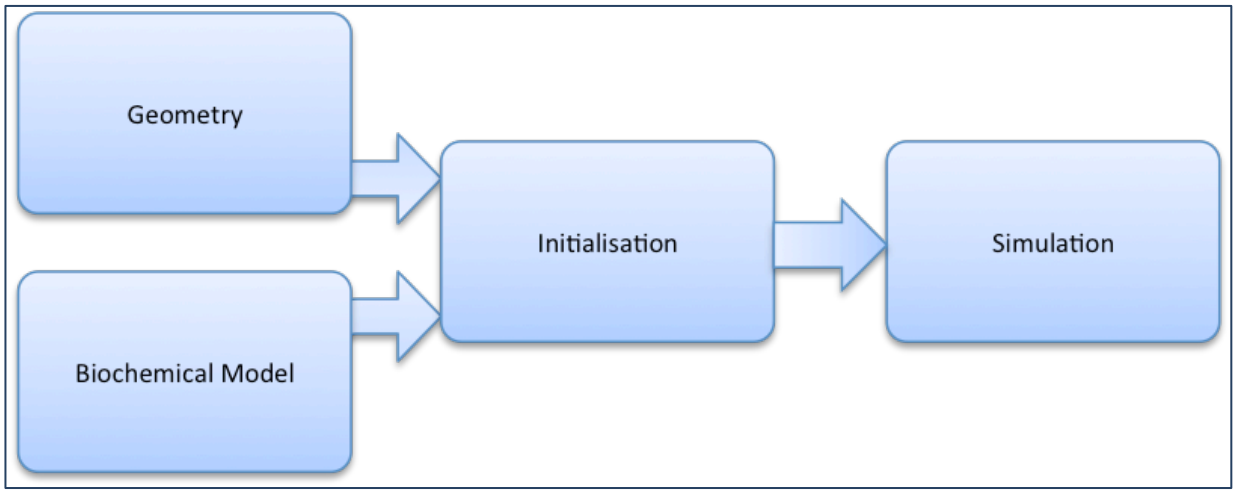

Figure 10: MolSim Configuration and Simulation Process

The molecular model is represented in a database and components of the database are selected according to the cell type and region being simulated. Reactions in the database are drawn from the literature and tagged with the source of the settings. The molecular model can be imported from SBML to facilitate input to the simulator. Geometry can either be abstract compartments or 3D mesh models based upon neuropil generated from the actual circuit.

An object-oriented architecture, with detailed subcomponents for the model, geometry, and solver objects, exists in the STEPS simulator. This architecture includes support for

reactions and diffusion on both volumes and surfaces. The geometry can accommodate compartments and tetrahedral meshes as well as membrane interfaces (not shown). The solver can handle well-mixed and tetrahedral systems and includes an Efield object for computing electrical fields in the tetrahedral system (not shown).

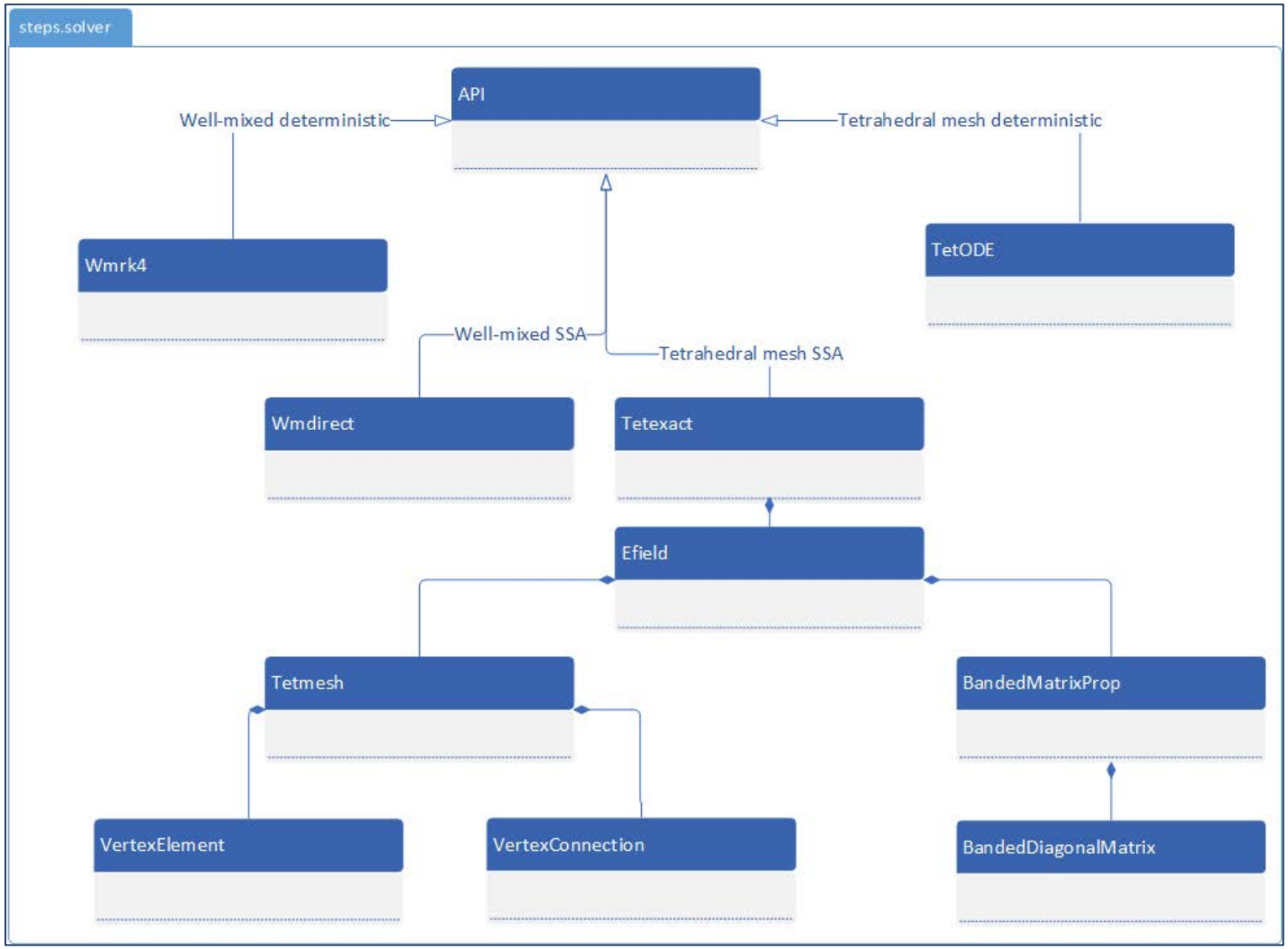

Figure 11: Detailed Components of the STEPS Simulator

Simulation will be either in serial or parallel mode. In serial mode, the simulator will run on a single processor and multiple runs could be farmed out to a compute cluster or a distributed computing system. As reactions with low numbers of molecules may exhibit large stochastic effects, multiple seeds are often necessary to characterise a model. Serial mode simulations are best suited for smaller systems due to the large amounts of time required to model large systems. Parallelisation will allow convenient modelling of larger systems such as portions of dendrites or entire neurons. Taking advantage of the increasing number of cores/threads and the available amount of memory per node, STEPS will first be made parallel using OpenMP. This will speed up the calculation of the application without introducing synchronisation complexity. Building on this experience and preliminarily prototypes, we will implement a distributed version of STEPS using MPI and target both cluster and supercomputing architectures. As particles will frequently move from one parallel region to another, utilisation of frameworks, such as Zoltan[5] or Parmetis[6], will be investigated to balance both particles and tetrahedrons dynamically.

5.6 MolSim: Relations to other Platforms

5.6.1 Functional Requirements on other Platforms

Use Case	External Platform Dependencies
SP6BSP-UC-015	Neuroinformatics Platform for semantic metadata handling and ontologies Neuroinformatics Platform for molecular and reaction properties database.
SP6BSP-UC-016	HPC Functional Requirement 1.4.2, 1.4.3, and 1.4.6. Neuroinformatics Platform for molecular and reaction properties database. HPC Functional Requirement 1.4.2, 1.4.3, and 1.4.6. HPC Function 7.3.4.1 and 7.3.4.4 (no functional requirement to refer to for these Task Functions)

Table 8: COLL Functional Requirements on other Platforms

5.6.2 Services provided to other Platforms

- Serial and Parallel Molecular Simulation Execution Services.

5.7 MolSim: Dependencies

5.7.1 Required

The following activities are required to satisfy the minimal Use Cases defined in Section 4.2 of this document.

- Development of database of cell-type specific molecule concentrations and reactions.
- Inclusion and analysis of EM data in the Neuroinformatics Platform to allow volume correction of meshes exported with the Molecular Experiment Builder.

5.7.2 Preferred

The following activities will be necessary to achieve the full potential of the MolSim component.

- Development of molecular dynamics models in 6.3.

6. BSP: Cellular Simulator (CellSim)

6.1 CellSim: Overall Goals

The CellSim component is the simulator engine in the HBP capable of simulating large-scale, biophysically realistic neural tissue models and providing observables that relate directly to experimental measurements. As with the other simulators in the BSP, the CellSim is subject to two competing requirements. On one hand, the simulator has to provide enough flexibility to allow modelling of current and future biophysical detail used to describe the various processes and objects in the neural tissue. The complexity of the model will depend on the detail of biological configuration data available, and this will increase over the duration of the project. On the other hand, the CellSim must keep execution times as low as possible while implementing increasingly complex large-scale tissue models. Otherwise, the CellSim will not maintain its utility as a scientific tool. This requires that it be optimised for memory consumption and for scalable performance on massively parallel supercomputing architectures.

6.2 CellSim: Use Cases

6.2.1 Simulation of a Microcircuit with Biophysically Realistic Neurons (SPBSP-UC-017)

Primary Actors: Scientific User, Abigail.

Preconditions:

- A reconstructed microcircuit model from the Brain Simulation Platform Brain Builder with morphologically detailed neurons and synaptic locations.
- Cell models describing the biophysical properties of the neurons contained in the microcircuit.

Success Scenario:

1) Abigail wants to conduct an *in silico* experiment on the cortical microcircuit in order to test the effect of thalamic stimulation; the microcircuit is too big and too detailed to be simulated on a workstation and requires a supercomputer.
2) She chooses the appropriate microcircuit model from the Brain Simulation Portal that was previously built.
3) Abigail opens the Cellular Experiment Builder and configures thalamic inputs and a stimulation protocol using data identified in the NIP. She selects the duration of the experiment and selects a set of neurons from which she wants to have current clamp traces recorded.
4) She then launches the simulation from the Brain Simulation Portal.
5) Once the Portal notifies Abigail that the Job is complete, Abigail runs an analysis on the completed simulation to produce a movie of the voltage activity of the microcircuit.

6.2.2 Multi-Parameter Exploration of Medium sized Networks with Biophysically Detailed Neurons (SPBSP-UC-018)

Primary Actors: Scientific User, Abigail; Biological Electrophysiologist, Bill; HPC specialist, Chris.

Preconditions:

- A reconstructed mesocircuit model from the Brain Simulation Platform Brain Builder with morphologically detailed neurons and synaptic locations.
- Cell models describing the biophysical properties of all neurons contained in the mesocircuit model.
- Performance analysis and benchmarking, and optimised version of the CellSim software.

Success Scenario:

1) Bill is scheduled to do an experiment in the wetlab the following day; he has a hypothesis of the effect of a particular multi-electrode array stimulation pattern. What he doesn't know is the exact pattern and stimulus strength.
2) Abigail has a mesocircuit model of the area of interest in the Brain Simulation Platform.
3) Bill asks Abigail whether she can run a parameter exploration on a mesocircuit model in order to help him reduce the amount of experiment time; however, they only have one day to come up with the computational answer because of the lab schedule.
4) She chooses the appropriate brain region model from the Brain Simulation Portal that was previously built.
5) Abigail opens the Cellular Experiment Builder and configures a parameter sweep on for the multi-electrode array stimulation. She selects the duration of the experiment and the recording channels.
6) She queries the Brain Simulation Portal for the duration of the simulation and realises that the sweep will not finish in time with the default CellSim.
7) Chris has previously worked on the performance characterisation of the CellSim code and has developed an optimised version of it; the Portal indicates to Abigail that a number of different CellSim versions are available and shows what changes have been made in those versions. Chris' version is described as optimised, so Abigail decides to directly contact Chris.
8) Abigail opens a chat with Chris and discusses her simulations; Chris informs Abigail that the simulation she is about to run is covered by his optimised version of CellSim.
9) Abigail chooses to go ahead with the optimised version of CellSim and launches the parameter sweep; she subscribes Bill and Chris to the notification
10) Once the job is complete, Abigail, Bill and Chris are informed; Abigail performs further analysis; Bill goes about launching his experiment; Chris analyses the performance data of the parameter sweep.

6.2.3 Full-Scale Simulation of an Entire Brain Region with Biophysically Realistic Neurons (SPBSP-UC-019)

Primary Actors: Scientific User, Abigail; Experimentalist, Bill.

Preconditions:

- A reconstructed brain region model from the Brain Simulation Platform Brain Builder with morphologically detailed neurons and synaptic locations.
- Cell models describing the biophysical properties of all neurons contained in the brain region model.

Success Scenario:

1) Abigail wants to conduct an *in silico* experiment on an entire brain region model of 10 million detailed neurons; due to all the biophysical detail, the brain model is too big to be simulated on her regular supercomputer allocation.
2) She chooses the appropriate brain region model from the Brain Simulation Portal that was previously built.
3) Abigail then opens the Experiment Builder and configures a bath experiment protocol. She selects the duration of the experiment, and then selects the option to have Local Field Potential recorded.
4) She then launches the simulation from the Brain Simulation Portal.
5) The portal informs Abigail that the brain model of her choice cannot be simulated on the currently available resources. It informs Abigail that she can a) wait about two days before a bigger partition frees up, or b) use a memory-optimised version of the CellSim, which is possible since the chosen model ingredients are supported by this simulator.
6) Abigail chooses to go ahead with the memory-optimised version of CellSim.
7) The Portal invokes a special process of writing a cache-efficient memory configuration for the memory-optimised CellSim and launches the simulation; it sets up the data and post-processing pipeline to calculate the Local Field Potential on the fly.
8) Once the Portal notifies Abigail that the Job is complete, Abigail can invoke frequency analysis on the recorded local field potential.
9) She shares the local field potential recordings with Bill, who has recorded LFP from the same brain region in the wetlab. Together they discuss the results.

6.3 CellSim: Functional Requirements

6.3.1 Neuron Models (SPBSP-FR-012)

The main requirement for the CellSim component is to provide a cell-based abstraction faithfully representing the 3D morphology of neurons. In order to accurately capture the electrical properties of these dendritic and axonal trees, a representation has to be provided that allows diameter, channel density, synaptic locations, etc. to be described for the entire neuronal tree.

For extensibility and configurability of the biophysical properties along the neuronal tree (e.g. ion channels, synapses, gap junctions, etc.), a mechanism is required by virtue of which novel mathematics can be integrated efficiently into the integration loop of the simulator.

In order to accommodate inclusion of physical phenomena beyond cell-based abstraction, such as local field potentials, additional solver methods for linear networks are required

6.3.2 Recording Devices (SPBSP-FR-013)

Any set of variables can be recorded during simulation and dynamically stored or sent out of the simulation engine. This allows the definition of virtual biophysical recording devices (e.g. *in silico* patch clamp recording) or direct tracking of any variable accessible to the simulator for post-processing.

6.3.3 Stimulation Devices (SPBSP-FR-014)

In order to conduct *in silico* experiments, virtual stimulation devices need to be provided that model (among others):

- Electrical stimulation at any portion of an *in silico* neuron (e.g. *in silico* patch clamp stimulation).
- Electrical field stimulation linking to multiple portions of a neuron and multiple neurons (e.g. multi-electrode array stimulation).
- Synaptic stimulation from external sources (e.g. projecting fibres from other brain regions).
- Bath manipulation in the entire simulation.

6.3.4 Synapse and Plasticity Models (SPBSP-FR-015)

Tightly related to the requirement of CellSim to provide a mechanism for defining a cell's biophysics, CellSim is required to provide a mechanism to flexibly configure any type of phenomenological model of synapses and their plastic behaviour.

6.3.5 Network Connection Routines (SPBSP-FR-016)

CellSim is required to provide functionality to morphologically and geometrically represent networks of neurons. It needs to be possible to either generate these network representations programmatically or instantiate them from predefined data.

For efficiency the network connection routines needs to be computer architecture aware (load balancing) and able to operate on distributed memory machines.

6.3.6 Use Case Mapping

The table below indicates which Functional Requirements are used to satisfy each Use Case. An X in the matrix below indicates that the Functional Requirement in the column is necessary to satisfy the Use Case in the row.

SP6BSP-FR-XXX	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	016
SP6BSP-UC-017		X	X	X	X	X	X					X	X	X	X	X
SP6BSP-UC-018		X	X	X	X	X	X					X	X	X	X	X
SP6BSP-UC-019		X	X	X	X	X	X					X	X	X	X	X

Table 9: BB Use Case to Requirement Mapping Table

6.4 CellSim: Non-Functional Requirements

6.4.1 Numerical methods for Neuron Models

- Accurate and efficient methods for integrating differential equations describing the phenomenological models of ion channels, synapses, gap junctions etc. are required. These methods can come with fixed time step and variable time step flavour.

- Accurate and efficient methods for implicitly solving the discretised cable representation of the neurons are required.
- Accurate and efficient methods are required for solving linear electrical models.

6.4.2 Interfaces

All functionality of the CellSim simulator needs to be programmatically accessible. This accessibility needs to be provided on various levels:

- A scripting level for flexible configurability driven by the User or the Brain Simulation Platform.
- APIs in various programming languages for other software to consume the simulator's various output formats.

6.4.3 Efficiency

In addition to the flexibility requirements, CellSim is required to efficiently simulate large-scale networks of detailed neurons. Flexibility and efficiency sometimes are mutually exclusive goals and CellSim therefore needs to provide multiple mechanisms to fulfil one or both requirements.

The base requirement for CellSim is to map to large, massively parallel supercomputers with distributed memory, independently of whether they are optimised for flexibility or efficiency.

In some cases, computational efficiency is the most important requirement and flexibility is traded for compute efficiency. This can be realised through using kernels specialised for a particular set of mathematics and optimised for single thread performance and optimal parallelism utilisation.

In other cases, minimal memory consumption is the most important requirement and flexibility is traded for memory efficiency. This can be realised through using a specialised compute engine.

6.4.4 Parallelisation

The description of the network of neurons needs to be independent from the parallelism of the supercomputer. It should be possible to instantiate tissue models of a particular size (and subject to overall memory constraints) on supercomputer partitions of different sizes. This allows the researcher to choose the size of a compute partition as a function of the required time-to-solution (strong scaling), e.g., in SPBSP-UC-018. It will also support dynamic resource management in the Interactive Supercomputing configurations.

6.5 CellSim: Architectural Overview

The CellSim component is based on the simulator NEURON (http://www.neuron.yale.edu), which will be further developed in the context of the HBP in order to meet the overall goals of the CellSim component.

NEURON is a highly scalable simulation engine that supports modelling fully detailed 3D neuronal network. It is made of six main components (see Figure 11 below), which support I/O operations, memory management, network communication, topological query and modifications, resolution of linear algebraic systems and modelling of biological mechanisms (synapses and channels).

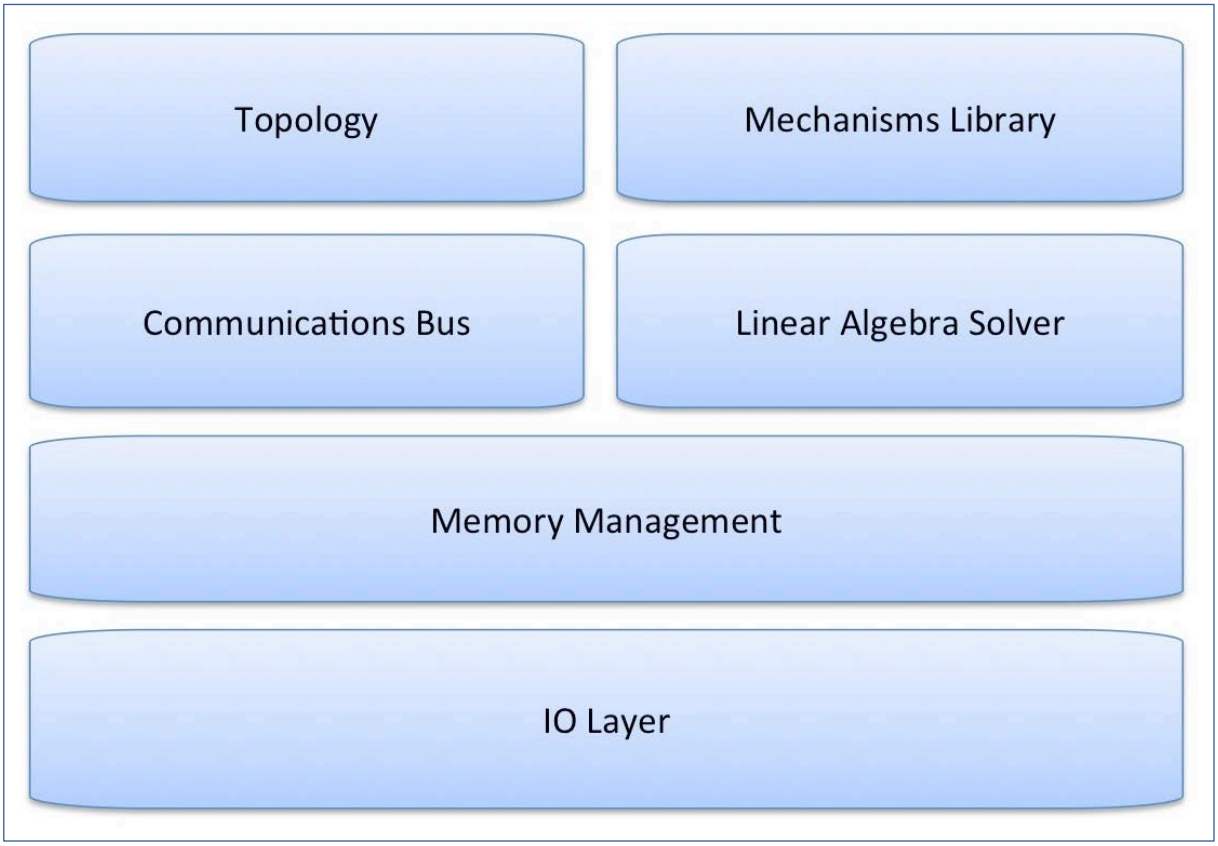

Figure 12: NEURON Architecture and Components

Each component of the NEURON architecture can be supported by multiple implementations depending of the defined performance requirements and targeted hardware architecture. In order to meet the two overall goals—configurability and efficiency for large-scale tissue models—two versions will be developed.

The General CellSim component is based on a combination of NEURON[7] software, EPFL ReportingLib and Neurodamus[8] software. It takes circuit configurations from the Brain Builder component and supports preparing the data layout and communication pattern necessary for simulating it on a parallel supercomputer, it provides all the configurability and introspection capability towards the scientist as well as simulation capabilities. Building on NEURON's 30 years of experience, it supports a very broad set of biological features (Cable theory, Active and passive dendrites, Gap Junction, electrical synapses, dynamics of spike initiation, etc.). Its MPI implementation proved to scale at large scale (more than 64,000 processors) even when the load per processor is reduced to a minimum.[9] [10] Using both HOC And Python interpreters, general CellSim is highly configurable and thus allows very fast prototyping. However, such flexibility and completeness requires overhead in the memory footprint and thus limits the size of tissue models that can be simulated on a given architecture.

The CellSim Engine is a reduced version of NEURON, specialised for minimising memory requirements for large-scale simulations. As part of this development, the NEURON memory footprint will be minimised by carefully designing very compact data structures. Maximum performance and scalability will be achieved via a combination of low-level and sometimes machine-specific optimisations (SIMD, combination of threading and MPI implementation, etc.).

6.6 CellSim: Relations to other Platforms

6.6.1 Functional Requirements on other Platforms

Use Case	External Platform Dependencies
SP6BSP-UC-017	NIP for simulation analysis tools HPC Functional Requirement 1.4.1, 1.4.2, 1.4.3, and 1.4.6.
SP6BSP-UC-018	NIP for simulation analysis tools HPC Functional Requirement 1.4.1, 1.4.2, 1.4.3, and 1.4.6.
SP6BSP-UC-019	NIP for simulation analysis tools HPC Functional Requirement 1.4.1, 1.4.2, 1.4.3, and 1.4.6.

Table 10: COLL Functional Requirements on other Platforms

6.6.2 Services Provided to other Platforms

- Serial and Parallel Molecular Simulation Execution Services.

6.7 CellSim: Dependencies

6.7.1 Required

The following activities are required to satisfy the minimal Use Cases defined in Section 1.3 of this document.

- Display Wall and CAVE installations for running high-end simulation visualisations

6.7.2 Preferred

The following activities will be necessary to achieve the full potential of the Cellular Simulator.

- Extensive optimisation efforts for the subset of the NEURON Simulator that is deemed necessary to enable the Brain Modelling efforts of WP6.4.

7. BSP: Network Simulator (NetSim)

7.1 NetSim: Overall Goals

The network simulator is the HBP's simulation engine for network models at the level of resolution of neurons and synapses. The major requirement informing the design process is the ability to represent, at the given level of resolution, parts of the brain at full scale. This means that brain networks can be represented at their natural size with all their neurons and synapses without downscaling. The focus of the simulator is on large networks of simple model neurons. Neurons are typically abstracted to point models, each of which is described by a set of coupled differential equations for a few dynamic variables. The simulator meets the memory requirements arising from the representation of the natural number of synapses per neuron by distributing the neurons over the nodes of a parallel computer. Connectivity information is stored only locally, on the machine on which the receiving neuron resides. The parallelisation of the data structures and algorithms is hidden from the User, enabling the same network simulation scripts to be executed on a laptop and on a supercomputer, provided the network size is adapted accordingly.

The simulator has a modular structure that is separated into a user interface layer, a simulator kernel, and neuron and synapse models, as described in Section 7.6. This modularity allows Users with modest programming experience, agnostic of parallelisation details, to add new neuron and synapse models to NetSim (Use Case 6.1.4). It also allows Users to employ different numerical solvers tailored to the properties of the neuron model and delivering the accuracy requested by the researcher. Two important classes of solvers are exact integration (applicable to neuron models with linear sub-threshold dynamics) and generic numerical solvers for neurons with non-linear sub-threshold dynamics. In addition to solvers constraining spikes to a fixed time grid, NetSim also supports solvers representing action potential timing in continuous time with double floating point precision as described in Section 6.4.1.

The NetSim component relates to the Neuroinformatics Subproject, in particular to the HBP functional analysis toolkit, as illustrated in Use Case 6.1.1. Firstly, NetSim creates data in a format that can be analysed by the tools developed in the Neuroinformatics Subproject. A researcher thus can use the same analysis workflow for experimental and simulated data. Secondly, the BSP serves as a means to create data sets under well-defined conditions to calibrate newly developed analysis methods.

NetSim has a natural connection to the Theory Subproject, as illustrated in Use Case 6.1.3. The simulator provides neuron models that are frequently used in and amenable to analytical treatment, such as the family of leaky integrate-and-fire models and binary neuron models. Therefore, the simulator helps to assess the accuracy of approximations entering analytical descriptions of network dynamics. It can also be used to explore the dynamics of networks beyond the validity of the theoretical treatment. Finally, it can be used to check the robustness of results obtained for highly abstracted models when they are tested on more realistic network models, e.g., by using a neuron model with an intrinsic action potential generation mechanism instead of a hard threshold, or by comparing realistic-sized network simulations with theoretical results obtained for infinite-sized networks.

NetSim relates to the Neuromorphic Computing Platform, as it allows the researcher to investigate deviations resulting from restrictions of the model implementation on the neuromorphic hardware, as illustrated in Use Case 6.1.4. The modularity of the simulator

that isolates the neuron model implementation from the remainder of the kernel alleviates the implementation of the same restrictions in software imposed by the hardware.

Due to the real time constraints and the reduction to functional networks, networks relevant for the Neurorobotics Platform will usually be specified at the level of description supported by NetSim. The close mapping between neuron and synapse models available in NetSim and those of the Neuromorphic Computing Platform facilitates the transfer of models to hardware solutions for the Neurorobotics Platform.

The NetSim component is based on the network simulator NEST. Therefore, it incorporates a variety of technological aspects, such as the exact integration of models with linear sub-threshold dynamics (see Section 6.3.1), algorithms for synaptic plasticity (see Section 6.2.4), the framework for off-grid spiking including synaptic delays, and two user interfaces, based on a proprietary interpreted language as well as a Python-based user interface (see Section 6.3.2). NetSim combines distribution by MPI, starting one process per compute node, and utilises multi-threaded software components based on OpenMP within each process during the setup and simulation phase (see Section 6.3.4). The use of threads instead of one MPI process per core is essential, because each MPI process entails a memory overhead due to replicated data structures and process management. The neurons of the network are evenly distributed over the compute nodes in a round-robin fashion and communication between machines is performed by collective MPI functions. The delivery of a spike event from a given neuron to its targets requires each receiving machine to have the information available to determine whether the sending neuron has any targets local to that machine. In NetSim, this is realised by storing the outgoing synapses to local targets in a data structure logically forming a target list.

NetSim simulates at the given level of resolution parts of the brain at full-scale. This means that brain networks can be represented at their natural size with all their neurons and synapses without downscaling. It implements efficient methods to construct large networks of simple model neurons and provides precise methods to integrate the dynamics emerging in the network with delayed synaptic pulse coupling.

7.2 NetSim: Use Cases

This section describes Use Cases that are specific to NetSim and the components it depends on.

7.2.1 Simulation of a Multi-Layered Local Cortical Mesocircuit with Full Scale Connectivity (SPBSP-UC-020)

Primary Actor: Alice, a computational neuroscientist.

Success Scenario:

1) Alice wants to investigate the statistics of the spiking activity emerging in a network with realistic layer-specific connectivity.
2) To this end, Alice needs to simulate a point neuron microcircuit with full-scale connectivity. This is a point neuron network model with a number of neurons and synapses in the right order of magnitude for a small brain region (100,000 neurons, 1 billion synapses).
3) Alice specifies her network for NetSim by writing a compact Python script, making use of powerful high-level neuron and connectivity construction routines, which support randomisation of the connectivity itself as well as of connection parameters such as synaptic weights and delays.

4) She stimulates the network with stationary input using stimulation devices provided by NetSim.
5) Alice also connects a subset of neurons in the network to spike detectors, which will record spike trains to file during simulation.
6) Alice runs a downscaled version of her network on a local cluster to ensure that her simulation script works as expected.
7) Simulating the network at full scale requires HPC resources due to memory footprint and simulation time. Alice thus decides to use the API of the Brain Simulation Platform to execute her simulation.
8) The BSP runs Alice's simulation on available HPC resources and registers resulting output data together with the simulation script and NetSim configuration information.
9) Alice retrieves the results and analyses them using the tools provided by the HBP-COLL.
10) To explore the effect of spatial structure on network dynamics, Alice slightly modifies her original script to make connectivity and propagation delays distance-dependent, and re-runs her simulations.
11) Using HBP-COLL analysis tools, she easily compares results from homogeneous and spatially structured networks.

Technical requirements:

- NetSim must be capable of automatic parallelisation of the simulation specification to acquire sufficient main memory for representing the network.
- Simulating 100,000 neurons will require around 48 CPU cores with 1GB of main memory each.
- This will yield reasonable turn-around times, provided that both simulation and network construction are parallelised.

Postconditions:

- The simulation has finished and written the simulated spike data to file.
- The files are ready for further statistical analysis, for example as described in NIP Use Case 6.2.1, HBP Functional Analysis Toolkit.

7.2.2 Macrocircuit Model Combining Local with Macroscopic Connectivity (SPBSP-UC-021)

Primary Actor: Abigail, a computational neuroscientist.

Success Scenario:

1) Abigail is interested in emerging activity when combining a model of the local cortical circuit with the recurrent network exhibited between cortical areas.
2) Using the search functions in the HBP-COLL, Abigail locates the network construction code for the local network from Use Case 1. She slightly modifies the code so that it can be used as building blocks for a larger model.
3) Abigail writes a new script connecting the building blocks using the macroscopic connectivity obtained from anatomical data.
4) Abigail instruments her network with stimulation and recording devices.

5) Using tools provided with NetSim, she specifies tests for the correctness of the network connectivity generated by NetSim and executes these on a downscaled version of the network.
6) She submits her simulation Task to the Brain Simulation Platform for execution on high-end HPC resources.
7) Once the simulation is complete, she uses HBP-COLL tools to investigate how the mean activities and power spectra in different areas depend on random spiking activity representing extrinsic input.

Technical requirements:

- The model size of around 2.4 x 10^7 neurons requires supercomputing. All software components of the simulation engine must be available for supercomputers.
- The parallelisation must be invisible to the User.
- Procedures allowing the estimation of the required resources in terms of CPU cores and main memory are required to choose for a given network the most suitable HPC configuration, i.e., the minimum machine partition providing sufficient memory and reasonable performance.

7.2.3 Verification of the Biological Relevance of Theoretical Prediction Obtained in the Large N Limit (SPBSP-UC-022)

Primary Actors: Abigail a computational neuroscientist working analytically; Bill: a computational neuroscientist with experience in simulation.

Success Scenario:

1) Abigail has derived an expression for the correlation coefficient between the activities of pairs of neurons in a random network of binary neuron models in the limit of an infinite number of neurons. The analytical expression is nice and the relation between the quantities explains the observed small magnitude of the correlations.
2) Bill now wonders whether the terms surviving in the large N limit are really the dominating ones for networks of biologically relevant size.
3) Bill and Abigail team up to carry out simulations with varying N to obtain a graph showing the correlation coefficient as a function of network size.
4) Bill implements Abigail's model in a compact script using NetSim's high-level connectivity functions, making network size N an easily scaled parameter.
5) Bill runs simulations for a range of N values on a local cluster computer.
6) Bill and Abigail analyse the results using HBP-COLL facilities and find that in this case, the experimentally observed correlation coefficient indeed converges towards the theoretical result in the large-N limit.

Technical requirements:

- NetSim simulator supporting random connectivity, binary neurons and measurement of correlation coefficients at runtime.
- A cluster computer allowing simulation of up to 10^6 neurons in parallel with a queuing system allowing efficient execution of simulations of different sizes.

7.2.4 Robustness of Network Activity with Respect to Neuron and Synapse Model (SPBSP-UC-023)

Primary Actors: Abigail computational neuroscientists; Bill: expert on neuromorphic hardware.

Success scenario:

1) Abigail has a full-scale model of the cell-type specific in-vivo spiking activity in the local cortical circuit and wants to port a downscaled version of the model to specific neuromorphic hardware.
2) Bill tells her that unfortunately the hardware implements different neuron and synapse models than those Abigail used in her original work; the hardware also imposes constraints on parameter values.
3) Therefore, Abigail needs to carry out simulations of her circuit with a neuron and a synapse model that are mathematically identical to those available in the hardware.
4) Abigail extends NetSim with a module providing a neuron and a synapse model implementing the models available in neuromorphic hardware. This requires some C++ programming, but thanks to the modular nature of NetSim, Abigail can add these models through a dynamically linked library. No modifications to NetSim proper are required.
5) Abigail now substitutes the original neuron and synapse models with those implementing the hardware models in her local circuit model and tunes parameter values by running simulations repeatedly, until the hardware-like model shows the same dynamics as the original.
6) Bill uses the parameters to configure and run network simulations on the neuromorphic hardware.
7) Abigail and Bill then compare spike trains obtained from the original NetSim model simulations, the hardware-like NetSim model simulations and the hardware using HBP-COLL tools. Where discrepancies occur, they clearly distinguish between those caused by differences between the original and the modified model, and those caused by differences between the real neuromorphic hardware and its mathematical description.
8) To understand some observed discrepancies better, Abigail re-runs NetSim simulations, recording membrane potentials from a small number of neurons in addition to spike trains.

Technical requirements:

- Network simulator allowing the addition of neuron and synapse models in a modular fashion.

Postconditions:

- Three simulations have been executed successfully (NetSim original model, NetSim hardware-like model, and neuromorphic hardware).
- As a result, there are three data files containing the spiking.
- These data can subsequently be analysed for systematic deviations between the model versions.

7.2.5 Verification of Independence of Simulated Network States from Time Discretisation (SPBSP-UC-024)

Primary Actor: Abigail, a computational neuroscientist.

Success scenario:

1) Abigail has a network model exhibiting some degree of synchrony in the activity of the neurons. She wonders whether this really is a property of the dynamics or an artefact of the numerical methods the network simulator employs constraining all spike times to an equidistant grid.
2) In addition to the neuron model Abigail used in her work, NetSim also provides a variant implementation that determines the exact time of threshold crossings in continuous time.
3) Abigail modifies her network model to use this implementation, and configures NetSim to handle all spike times in continuous time.
4) She re-runs her simulations using HPC facilities, as continuous-time simulations are more computationally demanding, and because she needs numerous independent realisations to obtain good statistics.
5) Abigail compares results from grid-based and continuous-time simulations using HBP-COLL tools.

Technical requirements:

- The simulator must implement integration on a time grid as well as in continuous time.
- The precision of the time grid must be changeable by the User and the solution of the differential equations describing the neuronal dynamics on the grid must be exact up to floating point tolerance.
- The same simulation script with minor adaptations can be used to perform both simulations, with discrete timing and in continuous time.

Postconditions:

- The simulations with different temporal resolutions for the time grid and without grid have finished as expected and produced files with spike data.

7.2.6 Flexibility in Specification of New Neuron and Spike-Time Dependent Plasticity (STDP) models (SPBSP-UC-025)

Primary Actor: Abigail, a computational neuroscientist

1) Abigail finds an interesting new neuron model or STDP model in the literature and wants find out how her favourite network model behaves if her own neuron or STDP model is replaced by the newly published one.
2) Unfortunately the network simulator does not yet provide implementations of the new models. Abigail has programmed in C++ before but has no idea of parallelisation and the details of the simulation engine. Luckily the specification of a neuron model is well isolated from the parallelisation and communication parts of the software and basically just requires statements of the dynamical equations and the parameters involved. Thus Abigail takes an existing model as a template and manages the implementation of the new model well.
3) Later she finds that the author of the paper describing the new neuron model states that the reported results can be achieved reliably only with a specific numerical solver.

She therefore creates a variant of her implementation of the new model using the specified solver.

4) Running simulations using her own models as well as the new model with both solvers, she can discern the effects of the model as well as the numerical solvers.

Technical requirements:

- The definition of the neuronal dynamics must be contained in an isolated piece of source code of a few files.
- The integration of the code of a new neuron model into the simulator must be possible in a few lines of code.
- The parallelisation of the code must be automatic, given the solution of the dynamics of a single neuron has been implemented.

7.2.7 Detailed Investigation of the Correlation Structure of Neuronal Activity (SPBSP-UC-026)

Primary Actor: Bill, a computational neuroscientist.

1) Bill wants to investigate the detailed shapes of the time resolved cross-correlations exhibited by the spiking activity of pairs of neurons of different cell types in a large-scale network simulation.
2) Bill estimates that an incredible amount of data would have to be generated to obtain the histograms at the required accuracy.
3) He thus decides not to write out individual spikes but to compute the histograms already while the simulation is running, using the cross-correlation recording device provided by NetSim.
4) Bill runs his simulations, which write only compact correlation data to file, making data management and analysis feasible.

Technical requirements:

- NetSim must provide recording devices that estimate frequently used measures of neuronal activity, such as rates and correlations, on the fly during the simulation without the need to store the individual spike trains.
- The recording devices hold the computed measure in memory after the simulation has terminated. The User must be able to read out the result in a simple way to store or post-process the data.

7.3 NetSim: Functional Requirements

7.3.1 Current-Based Neuron Models (SPBSP-FR-017)

The simulator implements frequently used spiking point neuron models, such as leaky integrate-and-fire models with different choices for the synaptic currents (PSCs): unfiltered (delta function PSCs), exponentially decaying currents, and alpha-shaped PSCs (Use Cases 6.1.1, 6.1.2, 6.1.5, 6.1.6 and 6.1.7). These models are integrated exactly, as described in the Non-Functional Requirements below. For comparison with theoretical results the simulator implements the binary neuron model with a Heaviside gain function, as well as with a smooth sigmoidal gain function. The update of this neuron model is performed asynchronously (Use Case 6.1.3).

7.3.2 Conductance-Based Neuron Models (SPBSP-FR-018)

The class of integrate-and-fire models in NetSim also includes conductance-based synapses, where an incoming spike causes delta-shaped, exponential, or alpha-shaped postsynaptic conductance changes. These models need to be integrated numerically with a state-of-the art adaptive numerical solver. Furthermore, the exponential integrate-and-fire model (AdEx), the Izhikevich model (a simple conductance-based three-compartment integrate-and fire model), and neuron models with Hodgkin-Huxley type dynamics are provided along with methods of integration as described in the original literature (Use Case 6.1.4).

7.3.3 Recording Devices (SPBSP-FR-019)

Selected measures of the network dynamics can be recorded in a flexible manner. To this end, the process of a neuroscientific measurement is mapped in a natural manner to the simulation environment in the form of a set of recording devices. These devices are connected to the subset of neurons, the activity of which is being recorded (Use Cases 6.1.1, 6.1.2, 6.1.3, 6.1.4, 6.1.5, 6.1.6). The devices write the recorded measurements to an accessible file system as either floating-point numbers encoded in plain ASCII or a structured binary format. It is possible to record spike times, membrane voltages and conductances as direct measures. Moreover, a neuron can expose a subset of its dynamic variables as recordable entities. The User chooses the required recording devices in the script defining the network setup.

7.3.4 Recording Device for Correlations (SPBSP-FR-020)

The simulator is able to record pairwise correlations between any pair of neurons within the network. The device computes the cross correlation function between the activity of any pair of neurons within the network already at simulation time and only in the end provides the result to the User (Use Case 6.1.7).

7.3.5 Stimulation Devices (SPBSP-FR-021)

The experimental situation of providing stimulation to the network is mapped to the simulation in a natural fashion. Stimuli can be applied flexibly to an arbitrarily chosen subset of the neurons within the network. A suitable abstraction is a stimulation device. Common stimulation protocols are possible, such as the injection of a sinusoidal current, the injection of an approximation of white noise with constant variance and with sinusoidally modulated variance, the injection of uncorrelated and pairwise correlated sets of Poisson spike trains, and arbitrary spike patterns defined by the User on the level of the simulation script (Use Cases 6.1.1, 6.1.2, 6.1.4, 6.1.5, 6.1.6, 6.1.7).

7.3.6 Synapse and Plasticity Models (SPBSP-FR-022)

Synapses are implemented with synaptic amplitudes individually represented in double precision for each synapse, allowing synaptic amplitudes to be distributed. Synaptic delays are resolved up to the chosen simulation resolution.

In addition to static connections, the simulator implements different forms of synaptic plasticity. Spike-timing dependent plasticity (STDP) allows for additive and multiplicative weight dependence as well as power law dependence, and takes into account the interaction of all presynaptic with all postsynaptic spike times (Use Cases 6.1.4, 6.1.6). As a second form of synaptic dynamics, short-term plasticity (short term depression and facilitation) is required.

7.3.7 Network Connection Routines (SPBSP-FR-023)

The most basic algorithm to set up connections in NetSim expects a list of pre-synaptic neurons (or devices) and a list of the same number of post-synaptic neurons (or devices), and connects the corresponding elements in a one-to-one fashion. Because of the function-call overhead, this is not very efficient when creating large networks. To avoid the overhead, NetSim provides high-level functions, which create divergent or convergent connection patterns with a single call. In addition to connection algorithms with explicit source and target specification, randomised variants exist to support the User in creating networks on the basis of connectivity statistics. These functions also support randomisation of weight, delay and other synaptic parameters (Use Cases 6.1.1, 6.1.2, 6.1.3, 6.1.4, 6.1.5, 6.1.6, 6.1.7).

7.3.8 Network Connection Routines for Structural Connectivity (SPBSP-FR-024)

NetSim supports the representation of spatially structured neuronal networks, with neurons placed on regular grids or freely chosen locations in two and three spatial dimensions. Connections in such networks are created using high-level functions, which allow connection probability, weight and delay to be chosen in a distance-dependent manner (Use Case 6.1.1). A range of distance-dependent functions is provided for this purpose. This range can easily be extended by implementing additional C++-subclasses based on existing function classes.

7.4 NetSim: Non-Functional Requirements

7.4.1 Integration of Neuron Models

The modular structure of the code enables the addition of new neuron and synapse models by only specifying parameters and state variables as well as the dynamical equations. A standard solver can be called to integrate the equations. The time propagation of neuron models with linear sub-threshold dynamics, such as the family of leaky integrate-and-fire models with current-based synapses, is implemented according to the state-of-the-art, propagating the time evolution exactly on the chosen time grid (Use Cases 6.1.1, 6.1.2, 6.1.5, 6.1.7). The time propagator is pre-computed after choosing the simulation resolution, so that the neuronal update only requires floating point multiplications and additions.

7.4.2 Spike Interaction in Continuous Time

NetSim provides a framework to implement neuron models that account for the timing of spikes in continuous time. In this mode, the spike times are represented with double precision upon emission at the sending neuron and are taken into account at their precise time of occurrence plus synaptic delay at the receiving cell (Use Case 6.1.5, 6.1.7).

7.4.3 Interfaces

The full functionality of the simulator is available through the programmatic API. One interface supplies access to the simulator through a Python-based API to allow simple integration into the web-based and GUI frameworks (Use Cases 6.1.3, 6.1.4, 6.1.7, 6.1.6). A second Python-independent interface isolates the simulation code from the Python interpreter and ensures that the simulation engine is usable on platforms that do not provide Python (Use Cases 6.1.1, 6.1.2, 6.1.5). This interface provides a proprietary interpreted language with an interpreter that is compiled along with the simulation kernel and only depends on standard libraries available on today's supercomputers.

7.4.4 Efficiency

The simulation speed is sufficient to allow for a quasi-interactive working style on today's supercomputers that are the size of the K computer. At a typical load of 2,000 leaky integrate-and-fire model neurons per CPU core with 10,000 synapses each, including spike-timing dependent plasticity per neuron, the simulation of one second of biological time should result in wall-time of at most 3,000 seconds, in addition to less than 1,000 seconds for the network setup taking place in parallel on all cores (Use Case 6.1.2).

7.4.5 Parallelisation

The simulation scripts are formulated independently of the employed degree of parallelism (Use Cases 6.1.6, 6.1.7). Parallelisation and the distribution of the network elements over the processors of the parallel machine take place behind the scenes, and do not require intervention by the User. In particular, synapses are stored in a distributed manner that allows NetSim to benefit from the distributed memory of the parallel machine to meet the memory demands of full-scale simulations. The kernel is able to exploit thread parallelism within one compute node, as well as distribution and communication among compute nodes employing the MPI standard during the simulation phase and during network setup. Both memory usage and simulation time scale up to the full size of today's available supercomputers. Given 2 GB of memory per core and 600,000 cores, the maximum possible network size should exceed 1.5 billion leaky integrate-and-fire type point neurons.

In addition, many of the connection algorithms in NetSim are parallelised using OpenMP. Using MPI, the wiring process is trivially parallelised, because each process only establishes the connections that target neurons that are allocated to it (Use Cases 6.1.1, 6.1.2, 6.1.3).

7.4.6 Extensibility

The modular structure of NetSim allows the extension by new neuron models with modest efforts (see 6.4 and Use Cases 6.1.4, 6.1.6). New recording devices can be implemented by simply providing a new C++ class and registering it with the simulation kernel. As recording devices, stimulators can be defined easily by providing the corresponding C++ class.

7.4.7 Use Case Mapping

The table below indicates which Functional Requirements are used to satisfy each Use Case. An X in the matrix below indicates that the Functional Requirement in the column is necessary to satisfy the Use Case in the row.

	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	NFR1	NFR2	NFR3	NFR4	NFR5	NFR6
SPBSP-UC-020	X		X		X		X	X	X		X		X	
SPBSP-UC-021	X		X		X		X		X		X	X	X	
SPBSP-UC-022	X		X				X				X		X	
SPBSP-UC-023		X	X		X	X	X				X			X
SPBSP-UC-024	X		X		X		X		X	X	X			
SPBSP-UC-025	X		X		X	X	X				X		X	X
SPBSP-UC-026	X			X	X		X		X	X	X		X	

Table 11: NetSim Use Case to Requirement Mapping Table

7.5 NetSim: Architectural Overview

This section describes the components that make up the simulator NetSim. NetSim is based on NEST with 20 years of experience in the construction of a highly modular simulation code. There is a culture of continuously publishing the technologies used in NEST in the scientific literature. The simulator consists of a simulation kernel, which stores the network and drives the simulation in time, and the user interface layer that enables a convenient and interactive setup and control of simulations.

The simulation kernel is implemented in C++ and is composed of three main parts: i) the scheduler, which drives the simulation by updating the neurons and devices on the time grid given by the minimal transmission delay in the network, ii) the network class, which provides data structures for the efficient storage of the neurons and devices and the connections between them, and iii) the model manager, which provides factories for neuron and synapse models and allows the User to copy model prototypes for flexible parameterisation. The scheduler uses OpenMP for thread parallelism and MPI for the communication of spikes among compute nodes. It performs the actual calculation, which

is the propagation of the temporal dynamics of the neuron models as well as of the synaptic dynamics. In addition, it manages the communication among the compute nodes of the employed platform. The network class provides the functionality to set up the network elements and their connectivity. The implementations of neuron and synapse models as well as recording and stimulation devices constitute an essential part of NetSim, but model definitions are well separated from other parts of the kernel, so that the adaptation of an existing neuron model typically only requires the change of a single C++ source code file and its corresponding header file. The integration of a new neuron or synapse model in addition requires the adaptation of a few lines of code within the model manager to register the new model.

The user interface layer interfaces with the simulation kernel and consists of an interpreter of the proprietary simulation description language in which simulation scripts are formulated. This allows a convenient interactive control of the simulator without requiring recompilation upon changes to the network description. A parallel path for controlling the simulation kernel is provided by a Python-based interface. This interface builds on top of the proprietary interpreter to isolate the simulation code from the Python interpreter but at the same time enables efficient data exchange with the C++ level.

7.6 NetSim: Relations to other Platforms

7.6.1 Functional Requirements on other Platforms

Use Case	External Platform Dependencies
SP6BSP-UC-020	• NIP for the HBP Functional Analysis toolkit. Specifically requirements leading from NIP Use Case 6.2.1. • HPC Functional Requirement 1.4.2, 1.4.3, and 1.4.6
SP6BSP-UC-021	• HPC Functional Requirement 1.4.2, 1.4.3, and 1.4.6.
SP6BSP-UC-022	• HPC Functional Requirement 1.4.2, 1.4.3, and 1.4.6.
SP6BSP-UC-023	• HPC Functional Requirement 1.4.2, 1.4.3, and 1.4.6. • Neuromorphic simulator or hardware execution services
SP6BSP-UC-024	• HPC Functional Requirement 1.4.2, 1.4.3, and 1.4.6.
SP6BSP-UC-025	• HPC Functional Requirement 1.4.2, 1.4.3, and 1.4.6.

Table 12: COLL Functional Requirements on other Platforms

7.6.2 Services Provided to other Platforms

- Network Simulation Execution Services

7.7 NetSim: Dependencies

7.7.1 Required

- No further non-service development activities are required beyond those specified in Section 6.6.1.

7.7.2 Preferred

- The inclusion of the interaction of network objects by continuous variables would increase the scope of NetSim and connect the level of description of spiking neuron models to non-spiking population models. Whether this can be achieved in the funding period with the given resources depends on how quickly the simulation code for petascale supercomputers can be stabilised. The simulation kernel constructed for NetSim is able to fully exploit the petascale supercomputers within the order of 100,000 processing cores available during the funding period. At the corresponding level of description, this will be a simulation engine used for production in the initial phase of the HBP. While the simulation engine is optimised for the petascale, for exascale systems a completely new communication architecture and major development work will be required.

7.8 Key Performance Indicators

The development of NetSim is driven by the scientific requirements as outlined in Sections 6.3 and 6.4. A dedicated Milestone in the issue-tracking system of NetSim is used to collect all requirements (functional and non-functional), and to identify dependencies between requirements and issues that must be solved in order to regard a requirement fully implemented. To quantify and track the progress towards the completeness of all requirements described above, the number of fixed tickets (versus the total number of tickets belonging to the Milestone) is used as the Key Performance Indicator (KPI).

In total, 44 distinct development Tasks were identified at the start of the project for completing all requirements. The contents and the number of Tasks will be adapted in an agile fashion as the project progresses.

The table below contains the exact number of Tasks for the different requirements. For each of the Tasks, a ticket with detailed information has been created. The number of tickets fixed and the requirements to which they belong will be reported in each of the following interim reports.

	Functional Requirements								Non-functional Requirements					
	1	2	3	4	5	6	7	8	1	2	3	4	5	6
No. of issues	3	3	3	1	4	2	10	4	2	3	1	1	2	5

Table 13: Functional Requirements & Non-Functional Requirements Issue Counts

8. BSP: Initial Brain Models

8.1 Introduction

The BSP is developing a generic data-driven and algorithmic approach for biological reconstruction of the brain. Since it is impractical to map all levels of brain organisation using laboratory methods, the BSP's predictive reconstruction approach is a key companion mechanism to making brain mapping experimentally tractable. The approach identifies minimal data sets required for reconstruction and validation at each level of brain organisation. By employing algorithms based on biological assumptions, and exploiting the interdependencies between biological observables, the approach allows prediction of missing data, which are subsequently experimentally testable. As further interdependencies between observables are revealed, and as the library of validations at all levels of organisation expands ("A model of everything should do everything"), less rather than more data are required to reconstruct the biological organisation of the brain, and the problem of over-fitting is alleviated. The approach advocates the systematic integration of biological data at each level of brain organisation, and therefore constitutes a "plan for data" that is currently missing in neuroscience.

The algorithms and workflows that will be made available in the BSP will enable biological experts to reconstruct the brain at the levels that they focus on in their work. Collectively, this approach will cover all the required biological levels. The algorithms are generic in that they can be applied to reconstruct any aspect or component of the brain of any species, given a minimal data set. The approach will foster truly collaborative and multi-disciplinary research, which is integral to one of the major strategic goals of the project: achieving a multi-level understanding of the brain. The levels of the brain are defined as 1) molecular-level (ions, biochemical, proteins, RNAs, genes), 2) cellular-level (neurons, synapses and glia), 3) micro-level (microcircuits, modules; defined by local dendrites), 4) meso-level (brain regions, nuclei, e.g. somatosensory cortex; defined by axons projecting within a brain region), and 5) macro-level (brain systems, e.g. neocortex, limbic system and the whole brain; defined by axons projecting between brain regions).

The primary goal of WP6.4 (Initial Brain Models) is to define the initial data sets for reconstructing and validating reconstructions at each level of organisation, test the reconstruction process, guide and refine the reconstruction algorithms, and establish a core group of "early adopters" of the Brain Builder who will provide feedback to the HBP-COLL. To achieve these goals, WP6.4 will use the Brain Builder to develop initial brain models at the molecular-level (neurons, glial and vascular), cellular-level (neurons from different brain regions), meso-level (somatosensory cortex, hippocampus CA1, cerebellum) and macro-level (hippocampus, basal ganglia and neocortex). The cellular and micro-levels have been established previously for the somatosensory cortex and will be improved as implicit components of the meso- and macro-level of reconstruction.

WP6.4 will interact closely with SP1, SP2 and SP5 to satisfy the data needs of initial brain models. With SP1 and SP2, targeted experimental brain mapping will be defined to provide a standardised data set that will seed the initial models. A framework to integrate specific data generated in Partnering Projects, and general data from European and International Collaborations will be established. With SP5, techniques for organising reconstruction and validation data in generic ways that can be consumed automatically by BSP algorithms will be established and refined.

The various sub-Tasks and modelling milestones of WP6.4 are synchronised with the Platform development efforts in Work Packages 6.1, 6.2, and 6.5 so that Platform development is guided by the modelling, and the modelling can leverage and populate the

emerging Platform. As such, WP6.4 provides concrete and relevant Use Cases and beta testing of Platform functionality. In doing so, it populates the Platform with exemplar workflows that will inform Platform Users of the steps, data requirements and principles for the reconstruction of biologically accurate brain models. The success of WP6.4 activities is intended to lead the way for community adoption of the Platform.

Once the models are built and validated, the BSP will be used to configure virtual experiments that run on one of the included simulator systems. The BSP will have analysis packages developed by various HBP groups that will be usable through the web interface. The results of the simulations and analysis can be shared through the HBP-COLL.

8.2 A Generic Strategy for Reconstruction of Hierarchical Complex Systems from Sparse Data

Reconstruction at all levels of modelling in WP6.4 is guided by a multi-level, multi-constraint approach to reconstruction of hierarchical complex systems, based on previous work by the Blue Brain Project, and is termed the "biological reconstruction process" (BRP). According to this process, a complex system such as the brain is defined as a compound model composed of multiple embedded levels of component models. Component models in turn may themselves be compound models composed of component models. If a component model has no sub-components, it is referred to as *unitary*. It could be said that the hierarchy of models making up the complete compound model is a tree of models, where the unitary component models are the leaves.

Unitary component models encapsulate their biological, biophysical and physical mechanisms, often phenomenologically, and thereby define the point at which to stop accounting for deeper underlying mechanisms. For example, it is unlikely that the brain must be modelled at a quantum mechanical level of description (as unitary components) to accurately reconstruct its behaviour. Although quantum mechanical effects could turn out to be important for understanding the phenomenology of aspects such as synaptic dynamics and ion channels at a biophysical level, such phenomena can be accounted for by unitary component models of synapses and ion channels, which generate observables relevant for the operating regimes of interest. By contrast, much of the complex machinery of synaptic plasticity remains unobservable, so developing molecular-level models of plasticity plays an important role in refining our understanding its phenomenology.

The biological parameters of a component model at any particular level of organisation are constrained by known biological principles and their associated necessary and sufficient experimental data sets at that level. They are also constrained by the requirement that the component model must account for (be validated against) observed phenomena at the same level of organisation (i.e. not at the higher compound level). If sufficient data are available to constrain the parameters of the a component model, gaps in data can be predicted. If insufficient data are available, the parameterisation of component models that pass validation may not be unique, but the parameters of component models are nevertheless frozen and not adjusted when integrated into a higher-level compound model.

The biological assumptions and parameterisation of a compound model are validated only against data and phenomena at the same level. If validation fails, the biological data and assumptions at the same or lower levels need to be revisited, and additional biological data gathered to justify refinement. This discipline supports a systematic reconstruction and refinement process by: a) decoupling the biological refinement at compound levels from its components, greatly reducing free parameters to consider at any given level; b)

minimising error amplification and maintaining generalisation power of higher level compound models; c) guiding new experiments where data are missing or expected to be highly informative for models; and d) challenging our understanding of the principles of organisation at the component level, which account for observed phenomena at the same level. This process is fundamentally different from top-down modelling where component models are adjusted to reproduce high-level phenomena.

Validation takes two forms, intrinsic and extrinsic validation. The reconstruction data are used for intrinsic validation, and a different data set is used for extrinsic validation. The latter are typically a) a subset of reconstruction data not used in the reconstruction process, b) sparsely accessible experimental data of phenomena at the compound model level, and c) a data set describing phenomena (emergent properties) at the higher compound levels. Validation data sets used for extrinsic validation are expected to be sparse, e.g. the spatial distribution of synaptic innervation will be available for only a few morphological types due to the prohibitive cost of collecting such data sets in a dense manner. Nevertheless, sparse data sets support the validation Use Case, as they are intended to verify the model by testing its predictions, and not to parameterise it.

Intrinsic validation ensures that the model successfully accounts for the target phenomena. Free parameters are constrained (e.g., iteratively optimised) during the reconstruction process to ensure intrinsic validation passes (i.e., to reproduce the target phenomenon). Models that pass intrinsic validation are then tested for generalisation power – extrinsic validation. Reconstruction parameters may not be adjusted or optimised to pass extrinsic validation. Failing extrinsic validation requires the biological assumptions and data of the model to be revisited. Sub-component models may also be revisited in the same fashion, but this must occur with models isolated from each other, and with the outside of the free parameter optimisation loop ensuring intrinsic validation. This discipline is intended to protect against over-fitting and error amplification when integrating into subsequent compound models. It is an iterative process of refinement of biological data and assumptions (not parameters) that continues until the compound model passes extrinsic validation. Only then is the compound model ready to be integrated into a higher-level compound model.

The reconstruction process can be seen as exploiting interdependencies, such as self-similarity, in a complex system to solve the inverse problem of predicting missing data. According to this approach, a reconstruction is a hypothesis that can be falsified by examining how its myriad predictions hold up to all available validation experiments. This highly pragmatic approach forces us to refine our biological understanding, and it is necessary to effectively use any available data to reconstruct the highest fidelity models possible. We anticipate that such a biological reconstruction process, and the predictions generated by each reconstruction, will guide future data generation and facilitate a more coordinated global approach to mapping the brain.

In the following sections, reference reconstruction and validation processes and data are provided for the various levels of initial brain models. At each level of detail, it is anticipated that the best-characterised systems will provide benchmarks for the level of data integrated into the models, and the processes by which the data are integrated. These processes, with their required data sets, are expected to evolve over the course of the project as more data become available. Moreover, these benchmark models will drive refinement of the reconstruction process by uncovering biological principles that can be used when suboptimal reconstruction and validation data sets are available for other systems at the same level. Thus, for any given system of interest, reconstruction is an iterative process that leverages the insights obtained from the benchmark reconstructions to make the best use of available data. It also prioritises key reconstruction and validation data sets, which will have maximal impact on the fidelity of the reconstruction.

8.3 Cellular-Level Models

8.3.1 Overall Goals

Cellular-level models of neurons and glia are defined with the following scope: 1) they are structurally accurate at the light microscopic level of resolution (morphologies), and 2) they are functionally accurate at the phenomenological level of observable biological states at this resolution. This functional accuracy includes electrical (e.g., ionic currents, voltages), and chemical (e.g., calcium imaging of dendrites) properties (i.e. they do not explicitly contain molecules). Neurons or glial cells contain unitary component models of ion channels, receptors, signalling, etc. For cellular-level models, these component models are phenomenological in nature. WP3 will use molecular dynamics and other strategies, and Partnering Projects in SP1 and SP2 will generate biophysical data on, for example, ion channel biophysics to refine these models. SP5 will also gather all available data to develop highly accurate models of the components.

Such progressive refinement of component models is performed at that level of the reconstruction. Lower levels of description may be added (i.e. if an ion channel is reconstructed, then from a unitary model it becomes a compound model and its components are refined). For cellular-level models, we focus on the manner in which the biological components are integrated, and not on the component models themselves. The refinement of the component models is performed in a separate process driven by molecular level models. This isolation decouples data integration and refinement at each level, allowing the cellular-level modelling to continue while refinement of our understanding of the components at the molecular-level proceeds in parallel.

The Blue Brain Project has developed a generic reconstruction workflow for a cellular-level model of any single neuron, as described below. T6.1.3 will develop this approach to refine existing cellular-level models, and to apply it to a broader range of cell types with their repertoire of morphologies, electrical behaviours and component ion channels. For the subsequent integration of cell models into microcircuit models, sufficiently extensive populations of the experimentally observed morphological and electrical-behavioural classes are needed to impose a systematic, high-throughput automated approach to cellular modelling. A suitable strategy has been developed previously by the Blue Brain Project, as will be elaborated in the microcircuit modelling section below.

8.3.2 Model Description

The detailed neuronal modelling approach involves populating morphological reconstructions with enough classes of ion channel models, distributing them according to data or *a priori* assumptions, and optimising channel densities to match firing features of experimental traces collected under standardised protocols. Numerical solution of the models employs the cellular-level simulator (T6.2.2). This workflow will be adapted and developed further for glial cells in T6.4.1.

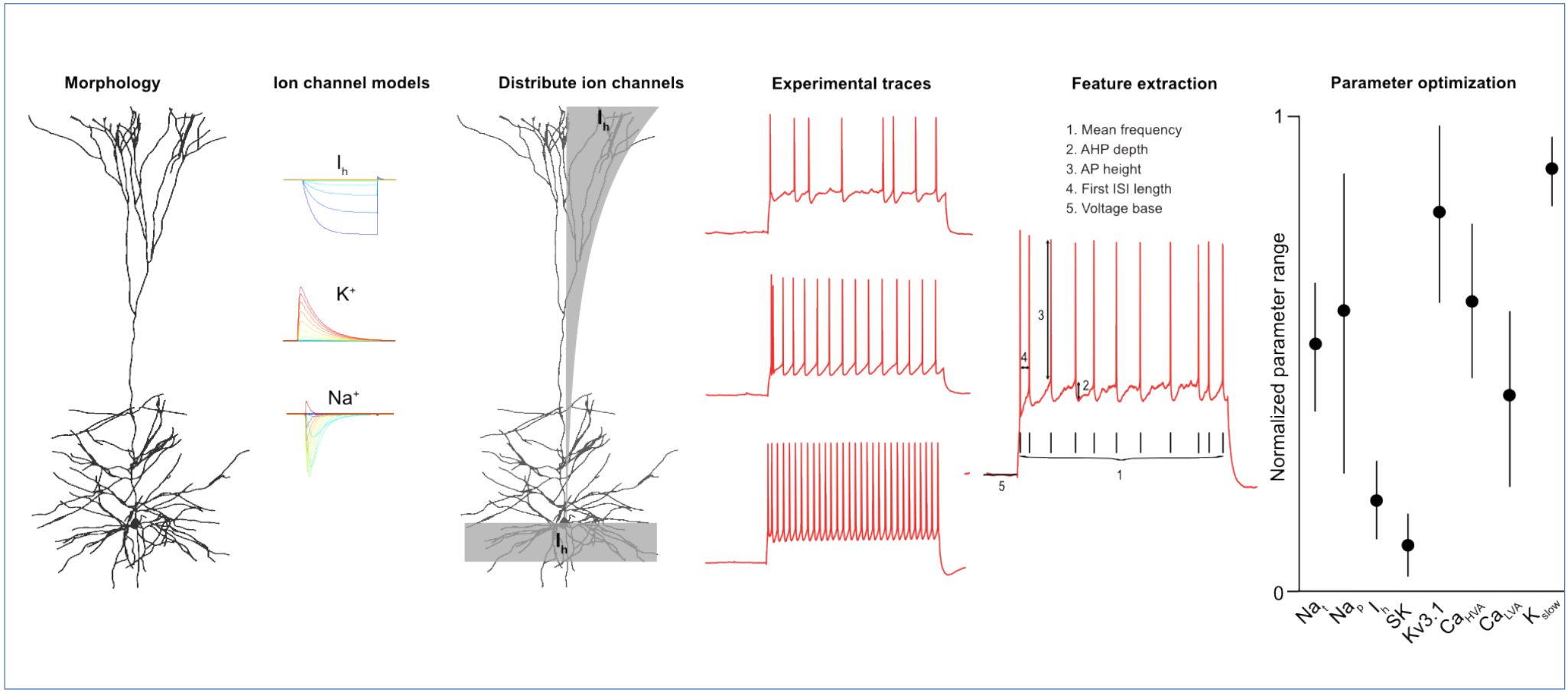

Figure 13: Schematic of the Cellular-Level Reconstruction Process

8.3.2.1 Reconstruction Data

The following experimental data, specific to the neuron to be modelled, are required or preferred for the reference reconstruction process as indicated:

1) Morphologically detailed light-microscopic reconstruction of a neuron.
2) Ion channels (component models):
   a) Main classes of ion channel kinetics & relevant models for each.
   b) Data on the distribution of ion channel classes along the arbours of the neuron.
3) Target phenomena:
   a) Standardised protocols characterising the electrophysiological responses of the neuron to current injection.
   b) Back-propagating action potential attenuation as a function of distance from the soma (morphology-type specific).

These data are sufficient to reproduce morphologically detailed and electrically accurate models of any neuron in a manner that does not depend on the exact component ion channel models used, so long as they are representative. By focusing on the principles of integrating ion channels rather than the ion channels themselves, we can leave the refinement of ion channel properties on an independent roadmap, refining as data become available. However, the data set required to integrate ion channels in a cellular-level model (distribution of conductances, absolute conductances) is also not available. To impose a process of systematic refinement we only allow one parameter to remain free. We select the parameter based on the data set that is likely to be the most individual. In this case the distributions are likely to follow strict rules imposed by common molecular machinery, while ion channel conductances could be different in each neuron. By setting conductance distributions we can now find the vector of ion channel conductances that are able to reproduce the particular electrical behaviour. As biological data on distributions become available these settings are updated, thus refining the biological accuracy of the model. It is important to note that for the resulting models, no significance can be placed on the resulting vector of ion channels since they are formed by classes of ion channels and not by a set of genetically expressed ion channels in a particular cell.

The ideal ion channel data set that would result in the electrical behaviour with accurate descriptions of the biological machinery in each cell type has been identified and includes:

1) Single cell gene expression data indicating the combination(s) of ion channels expressed in a particular morphological type of neuron.
2) Spatial distribution of each genetically expressed ion channel in neurons.
3) Biophysical properties of genetically expressed ion channels, together with mathematical models describing each of the ion channels (around 450 different genetically expressed ion channels in the brain).
4) Biophysical properties of ion channels under manipulations of the extracellular ionic composition to allow for changes under different simulated conditions (e.g. bath manipulation experiments).
5) Target phenomena: Electrical response characteristics to capture a full spectrum of phenomena – a subset for reconstruction and a subset for validation.

If, in addition to this ideal data set, the conductance of each ion channel were known, then no fitting would be required to reproduce the electrical phenomena. Since the absolute conductances of ion channels are likely to be different for individual neurons, we

propose that their measurement is of low priority since they can be predicted given the other data.

The reconstruction process as defined below allows for these data (combinations, distributions, kinetics) to be integrated as they become available, thus putting each cellular model on its own path of refinement.

To reconstruct structurally and functionally accurate model neurons that can then be modulated by receptor activation, we begin by freezing the electrical models, and adding and treating receptors in much the same way as ion channels. In other words, we:

1) Select the combination of receptors (the R-set model)
2) Distribute them according to any available data or based on *a priori* assumptions (the distribution model)
3) Use any available or develop new component models of the ligand-receptor activation for each receptor (ligand activation models)
4) Use any available or develop new component models that capture the relationship between a particular receptor and the set of ion channels (R-I Interaction model).
5) Adjust only the relative density of receptors with the null-hypothesis that the component models are sufficiently accurate to capture the phenomenon.

All component models are frozen, and only the concentration of the receptor is changed to determine whether the phenomena of receptor-induced changes in electrical properties (e.g. blocking spike train accommodation by activation of a muscarinic receptor) can be reproduced. The first model is likely to fail validation and when it does, the component models are curated in order of data availability. In this case, the biological data used to develop the distribution model are first reassessed. If the distribution model is found to be a reasonable starting point, then the other component models are curated. Each is treated in isolation as a compound model itself and their components are refined. For example, the tests that validated the receptor-ion (R-I) channel interaction model (e.g. dose-dependent changes in ion channel kinetics) are re-examined. This may trigger a new search for data, new experiments, or a reformulation of the component models. After curation, the models are once again frozen and integrated in the compound model and receptor densities are adjusted.

When two receptors are added to the neuron model, then only the receptor densities are adjusted to determine whether receptor-induced phenomena can be reproduced. In this case, there may exist an even larger repertoire of phenomena reported in the literature that may include phenomena caused by interacting receptor effects. For the biological reconstruction process, these interactions do not make the problem more complex, but actually provide an opportunity to constrain the components even further (they are valuable because they introduce interdependencies, which can be exploited to constrain the parameter space). The reconstruction must now reproduce interactions by only adjusting the densities of receptors. When the model fails, the distributions are once again re-evaluated and then each component model is curated, frozen and re-integrated. In this way, the reconstruction process guides which data sets are required for receptor-enhanced electrical neuron models. These include:

1) Combination of receptors expressed in a neuron.
2) Distribution of each receptor on a neuron.
3) Ligand-activation model: Dose-response curves of ligand vs. receptor activation, single receptor recordings during activation, single receptor conductances, etc.

4) Receptor-Ion channel interaction model: Biochemical and biophysical data on intracellular signal transduction pathways and effects on ion channels (e.g. dose-response of receptor activation and ion channel kinetics, dose-responses describing changes in receptor-activated second messengers, etc.).

5) Target phenomena: Changes in electrical behaviour when receptors are activated individually and/or in combination (i.e. the phenomena that must be reproduced); data on intracellular calcium dynamics (e.g. calcium spikes).

8.3.2.2 Validation Data

Here we list example data sets describing phenomena that were not used as target phenomena in the reconstruction, and against which the resulting reconstruction can be validated.

1) Electrical neuron models
   a) EPSP attenuation as a function of distance from the soma (morphology-type specific).
   b) Electrophysiological changes under different conditions such as changing the extracellular ionic compositions.
   c) Neuronal properties such as the input-output functions, resonance, pace-making, onset and location of calcium spikes.
2) Receptor-enhanced electrical neuron models: The effect of receptor activation individually or in combination with 1 a-c above.

8.3.2.3 Reconstruction Process

The reconstruction process follows steps in a workflow. While different workflows can be invoked for specific cases, a reference workflow is:

1) Electrical model:
   a) Select morphology
   b) Define the neuronal compartments—divided into morphological regions such as dendrites (apical and basal if applicable), soma, axon, and axon initial segment—with all relevant features (e.g. dendritic spines, axon hillock etc.).
   c) Configure the passive properties (e.g. capacitance, axial resistance, capacitance correction for spines).
   d) Select a class of ion channel models.
   e) Distribution model: Distribute channel conductances in a manner specific to the morphological regions of the model according to data on distributions if available, or *a priori* assumptions if not.
   f) Free-parameter: set any broad constraints on ion channel conductances.
   g) Extract features from experimental voltage traces of standardised protocols for electrophysiological characterisation (e.g. related to APs, AHPs, spiking frequency, etc).
   h) Employ multi-objective optimisation (MOO) fitting algorithms to optimise free parameter to capture features by mimicking the standardised electrophysiological protocols *in silico.*
   i) Perform intrinsic validation (with definition of statistical model acceptance criteria).

2) Receptor-enhanced electrical neuron model:
   a) Add receptor(s).
   b) Add receptor distribution models (informed/assumed distributions).
   c) Add ligand-receptor activation model.
   d) Add receptor-interaction model.
   e) Free parameter: receptor densities.
   f) Extract features from experimental voltage traces of changes in electrical properties when receptors are activated (e.g. changes in APs, AHPs, spiking frequency, etc.).
   g) Employ multi-objective optimisation (MOO) fitting algorithms to optimise free parameter to capture features by mimicking the standardised electrophysiological protocols *in silico.*
   h) Perform intrinsic validation.

8.3.2.4 Validation Process

Validation of cellular models follows the generic validation process described in 8.2. Intrinsic validation is against the reconstruction data and target phenomena. Extrinsic validation is against the validation data provided above.

8.3.3 Modelling Objectives

Accurate models of the diversity of neurons observed in a representative repertoire of operating regimes are a prerequisite for accurate models of neural tissues exhibiting their ranges of network behaviours. The procedure given here describes the workflow and requires experimental constraints to construct a single such neuron model. In the subsequent section, the generalisation of this approach to the population of neurons making up a local tissue volume, the microcircuit, is given.

8.3.4 Relation to other Models, Milestones and Deliverables

For the micro-, meso-, and macro-scale tissue reconstructions that follow, myriad neuronal models for the diversity of neuron types observed in the target systems of interest (cortex, hippocampus, cerebellum) will be required in sufficient quantities to represent the population diversity. Methods for this modelling will be developed in T6.1.3 and delivered as MS111. In some cases, where the available morphological reconstructions are insufficient, morphological synthesis algorithms could be used as delivered in MS112 in the context of T6.1.1. The availability of single-cell transcriptome data, the data analysis modules to exploit such data, and the experimental characterisation of these genetically expressed ion channels will further constrain neuron models with experimental data, relieving reliance on optimisation approaches.

8.3.5 Dependencies

8.3.5.1 Required

The following activities are required to demonstrate that the model can be generated by and used in the Brain Simulation Platform.

- Neuron models will require the cellular simulator (T6.2.2) for numerical solution.
- The neuronal modelling and optimisation capabilities developed in T6.1.3 (MS111) must be available in the Brain Simulation Platform.

8.3.5.2 Preferred

The following activities will be necessary to achieve the full potential of the model.

- Exposure of systematic and automated neuron model validation workflows in the BSP, which automatically configure and launch the required simulations for the various validation protocols, analyse the produced traces, perform statistical testing and generate reports.

8.4 Micro-Level Models (Microcircuits/Modules/Columns)

8.4.1 Overall Goals

Microcircuit models are defined by the dimensions of local arbourisation. The size of a microcircuit in any part of the brain can be objectively defined as the minimum size required to fill the neutrophil volume with dendrites. Microcircuit models integrate a coherent and self-consistent wealth of anatomical and physiological data on local volumes of nervous tissue. These data have been accumulated from experiments ranging from *in vitro* brain slices using sharp electrodes and patch-clamp recordings, 3D morphological reconstruction, electron-microscopy (EM) and histology to post-mortem anatomical studies such as immunohistochemical staining, cell counting etc.

While such experiments reveal the complexity of nervous tissue, the work of Ramon y Cajal already established in the early 1900s a generic principle of neuroscience. This general principle stated that the diversity and complexity of neuronal tissue could be effectively organised through classification of its constituents, and that the identification of anatomical regions of relative homogeneity defined by anatomical features and landmarks such as brain areas, regions, and sub-structure such as layers, etc. As a result, the integration of data into microcircuit models must work well with current paradigmatic classification schemes of cellular and anatomical features. The objective of microcircuit modelling is to unify all data on its anatomical and physiological composition under common classification schemes, ontologies, and partonomies, and use these data to reconstruct a minimal, yet representative, and characteristic microcircuit. This microcircuit is used subsequently as a component model in meso- and macro-level models to populate brain atlas co-registered volumes of nervous systems.

8.4.2 Model Description

A schematic of such a process for reconstructing the neocortical microcircuit, based on previous work in the Blue Brain Project, is depicted below. The unifying classifications of morphological, electrical, combined morpho-electrical, and synaptic behaviours are referred to as m- (morphology), e- (electrophysiology), me- (morpho-electrical) and s (synaptic)-types, respectively.

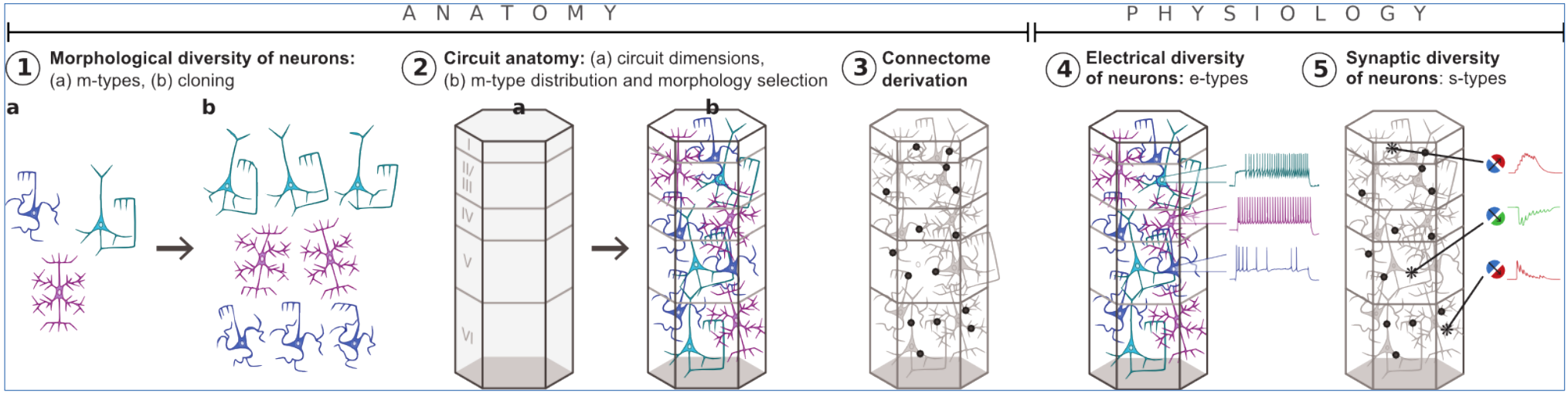

Figure 14: Schematic of the Microcircuit-Level Reconstruction Process

8.4.2.1 Reconstruction Data

The following reconstruction data, specific to the microcircuit to be modelled, are required for the reference reconstruction process.

- Architecture:
  - Tissue level properties are obtained from global tissue staining, allowing the identification of sub-structures—such as layers and mini-columns—from variations in cell soma sizes and densities, protein staining, and gene expression maps.
  - Data below on morphologies, morphological composition, cell density and cell positioning models are required for the Architectural Model to define the volume, dimensions and sub-dimensions of the microcircuit reconstruction.
- Neuron anatomy: Enough 3D neuron morphological reconstructions to allow clustering of morphologies into classes (m-types)**,** morphometrical analysis, repair (if the neurons are from *in vitro* slices), and cloning (to make unlimited copies with statistical variations).
- Morphological composition:
  - Total cell density from cell counting of Nissl stainings or similar in tissue blocks.
  - Relative fractions of excitatory and inhibitory neurons, and glia from simultaneous specific marker stainings.
  - Enough neurons anatomically identified in stained tissue to estimate relative numbers of each involved in the microcircuit.
- Cell positioning: Layer-annotated neuronal reconstructions to identify target-oriented clustering of arbours.
- Morpho-electrical composition: Recordings of neurons to assess major e-types expressed in different m-types.
- Synaptic anatomy**:**
  - The number and distribution of appositions and functional synapses between any two morphologies (pathway-specific connectivity).
  - Density of boutons along axonal reconstructions (m-type specific).
  - Bouton density profile and lateral extent for projections into the microcircuit or reconstructions of projection fibres of representative numbers.
- Cellular-level neuron models for the diversity of electrical firing classes exhibited by constituent neurons.
- Synaptic physiology models (component models):
  - Pathway-specific distribution of PSPs (post-synaptic potential amplitudes).
  - Pathway-specific characterisation of synapse dynamics (depression, facilitation).
  - Pathway-specific axonal conduction velocity.
  - Pathway-specific modulation of PSP amplitude with extracellular calcium concentration.
  - Physiological characterisation of projections into the microcircuit.

The question of whether synaptic properties can be predicted from the combined gene expression of any two neurons will be explored. Such a neuroinformatic predictor may be

assisted by a deeper understanding of the translation of proteins by genes and the synaptic proteome obtained in SP1 and SP2.

Quality criteria: uniformity of age, species, brain region; local data-specific quality criteria.

8.4.2.2 Validation Data

Here we list example data sets describing observed properties at the target level that not only serve as validation for the tissue level reconstruction, but also for its component models, including:

1) Architecture:
    a) Global gene expression maps
2) Neuronal anatomy:
    a) Neuropil volume fractions from EM
3) Morphological composition:
    a) Gene and protein markers of M-types
    b) Global protein expression maps
    c) Global gene expression maps
4) Cell density and positioning models:
    a) Inhibitory synapse density from EM
    b) Number of inhibitory synapses on somata
    c) Spatial distribution of synaptic innervation (m-type specific)
5) Morpho-electrical composition:
    a) Targeted recording of specific m-types to validate relative occurrence of expression of e-types.
6) Synaptic anatomy:
    a) Inhibitory synapse density from EM
    b) Number of inhibitory synapses on somata
    c) Pathway specific connection probabilities
    d) Distance dependent connection probabilities
    e) Spatial distribution of synaptic innervation (m-type specific)
7) Neuronal electrical models:
    a) Electrical properties of specific m-types during microcircuit activation
8) Synaptic physiology models:
    a) Amplitudes and rise/decay times of post-synaptic potentials (PSPs) and currents (PSCs)
    b) Concomitant voltage and current recordings of synaptic connections
    c) Dendritic recordings of PSPs and PSCs
    d) CV of PSP amplitude, failure rates, latencies

e) Network activity under depolarising bath manipulation (increased potassium) and high and low extracellular calcium concentrations

f) Layer specific EPSPs of external projections *in vitro* and *in vivo.*

Quality criterion: uniformity of age, species, brain region; local data-specific quality criteria.

8.4.2.3 Reconstruction Process

The reconstruction process follows steps in a workflow (as summarised in the schematic above). While different workflows can be invoked for specific cases, a reference workflow is:

1) The morphological diversity of neurons:
   a) The morphologies of reconstructed neurons are repaired if they were obtained from brain slices.
   b) Missing morphologies may be algorithmically synthesised, if a synthesis model exists.
   c) Reconstructions are analysed for their morphometric features for objective clustering and determination of m-types.
   d) The reconstructions are cloned by m-type to produce a large data set of sample morphologies for each m-type.
2) Architecture:
   a) Circuit dimensions:
      i) Neurons are loaded in a 3D space respecting:
         (1) Cell density- and position—related numbers that respect the morphological composition (relative fractions).
         (2) Any sub-structure organisation (e.g., layers) based on transitions in cell density and soma size in global staining map data.
      ii) The diameter of the microcircuit is increased until the dendritic neuropil saturates at the centre.
      iii) The diameter at 95% cut off is taken as the diameter of the microcircuit.
   b) Sub-structures:
      i) Define layers as extracted from reconstruction data.
      ii) If applicable, place mini-column centres according to a space-filling algorithm consistent with data on density.
3) The morphological composition:
   a) The 3D volume established above is used to define the dimensions of the microcircuit.
   b) The Neuron Morphology Models are loaded respecting cell densities, ratios of excitatory to inhibitory neurons, their positions within any sub-structures (e.g. layers), and fractions of m-type expressions.
4) The connectome:
   a) Identify all appositions between the morphologies using a touch distance determined by electron microscopic estimates.

b) Convert appositions into functional synapses according to data on functional and structural anatomy of synaptic connections using the connectome algorithm developed in the Blue Brain Project, or equivalent for tissue being reconstructed.

c) Adjust for any region specific differences and repeat.

5) The morpho-electrical composition: Apply fractions of e-type to each m-type.

6) The physiology of synapses:

a) Iteratively scale synaptic conductances from values quoted experimentally (to account for the space-clamp error) in a pathway-specific manner to match data on pathway-specific distributions of post-synaptic potential (PSP) amplitudes.

b) Map known synaptic dynamics between different m-, e-, and me-type neurons, and use generalising rules to assign synaptic dynamics to unknown pathways.

8.4.2.4 Validation Process

Validation of micro-level models follows the generic validation process described in 8.2. Intrinsic validation is against the reconstruction data and target phenomena. Extrinsic validation is against the validation data provided above.

8.4.3 Modelling Objectives

The generic microcircuit reconstruction strategy will first be applied to build a model of a rat somatosensory cortex required for M119, based on prior experience of building such models in the Blue Brain Project. This process will proceed in parallel with the development of the Brain Builder, and serve as a target Use Case to guide its development. Moreover, it will serve as an initial example model and generic microcircuit reconstruction workflow in the Brain Builder. It will establish for HBP microcircuit models the minimal rigor by which the various reconstruction and validation experimental data should be curated, model components should be validated, and emergent properties should be explored and predicted. Until M30, the model will be subject to refinement. It will undergo a refinement-release cycle, further guiding and testing the supporting model lifecycle functionality of the Portal. Further microcircuit models will follow, including hippocampal (CA1) and cerebellum.

8.4.4 Relations to other Models, Milestones and Deliverables

A somatosensory cortex model (M119) is due to be delivered at Month 12, six months prior to the delivery of the HBP Brain Simulation Platform (BSP), as it is intended to serve as a first Use Case and test-case guiding the development of the BSP. It also serves as the first model to populate the BSP as a guiding model for early adopters to learn from and explore. The model will be developed by leveraging and augmenting an initial BBP model public release with deeper model access for the HBP, via platform-enabled capabilities.

A detailed model of the CA1 region (meso-scale) of the hippocampus (M121) is planned for delivery at Month 24, for which a necessary condition is a hippocampus CA1 microcircuit model. This work will serve as a test case for the general ability of the BSP to reproduce experimentally verifiable features of microcircuits beyond the cortex. WP6.4.5 deals specifically with developing meso-scale and microcircuit models of the cerebellum.

All microcircuit models will be integrated into meso- and macro-building workflows for the planned models of the full neocortex, cerebellum and hippocampus (M122) in Month 30.

8.4.5 Dependencies

8.4.5.1 Required

Microcircuit models are planned for the somatosensory cortex, hippocampal CA1 and cerebellum. Required data availability is summarised as follows.

- Somatosensory cortex - Reconstruction data sets are available from experiments performed by EPFL-BBP, HBP partners, or literature for a rodent somatosensory non-barrel cortex. Validation data sets for a rodent somatosensory non-barrel cortex are available from experiments performed by EPFL-BBP, HBP partners, or literature.
- Hippocampus CA1 - Reconstruction data sets are available for a young adult rodent from experiments performed by UCL, HBP partners, or literature. Where specific information is not available, data from other rodent species may provide an adequate alternative in some cases. A subset of the validation data sets will be available from HBP partners or literature.
- Cerebellum – Precise and highly detailed models of cerebellar neurons have been constructed and validated. These include granule cells [10,11,12], Golgi cells[12,13], Purkinje cells (Masoli *et al*., in preparation), inferior olive cells (Masoli *et al*., in preparation), and unipolar brush cells (Sathyaa *et al*., in preparation). In previous work at Universita Degli Studi Di Pavia a large set of microcircuit model parameters has been precisely tuned against a multiplicity of well-organised experimental data. The microcircuit validation data sets need to be produced.

8.4.5.2 Preferred

The following activities will be necessary to achieve the full potential of the model.

- The BSP simulation tools should be able to support simulations embedding biochemical networks and embedding in large-scale network models.
- BSP should allow use of models in simulations in for closed-loop robotic simulations.
- The development of reconstruction and validation processes would benefit from a flexible API framework allowing for the analysis of simulation outputs, or of neuron or microcircuit properties.

8.5 Meso-Level Models (Brain Regions/Nuclei)

8.5.1 Overall Goals

Whereas microcircuit models represent an average prototypical microcircuit of an identified brain structure, meso-scale models of neural tissues are defined here as whole volume registered (in a standardised atlas) reconstructions of single brain regions, such as non-barrel somatosensory cortex S1, visual cortex V1, hippocampus CA1, etc. Brain regions are also defined intrinsically by the axonal projections of their neurons. Axons that project beyond the range of a microcircuit, but not to other brain regions, are defined as meso-projecting axons. These axons connect microcircuits and the major challenge in reconstructing meso-level circuits is to derive the meso-level axonal projections to respect the clustered projections that are observed experimentally.

8.5.2 Description of Model

Meso-level models circuits differ from microcircuits in that they form a larger volume of brain tissue corresponding to a brain region containing many microcircuits; neurons project beyond the range of the local microcircuit connections (connecting microcircuits); and the

cellular composition may change medio-laterally or anterio-posteriorally. Given that the microcircuits for the region have been reconstructed, the main difference is that the entire volume and sub-structures (e.g. layer thicknesses) need to be demarcated for the whole volume. Any changes in cellular composition (cell densities, E:I ratios, m-types, me-types, positions) need to be defined, meso-projecting neurons need to be identified, and then these axons need to be synthesised to form the meso-circuit connectivity. Finally, the afferents and efferent inputs need to be defined.

Brain regions are volume registered in brain atlases (e.g. Allen Brain Atlas, Waxholm space, etc.), and therefore, meso-circuit reconstructions begin with data analysis/mining of these atlases for a given location to demarcate the volume and sub-structures. In cases where there is extensive curvature of the brain structure, morphology synthesis may be required to prepare candidate morphologies. Volume registration allows the addition of brain vasculature data sets locally for the region of interest (ROI).

Validation data sets at the meso-scale are generally emergent properties of network dynamics observed from preparations of intact animals *in vivo*, such as voltage sensitive dye (VSD) imaging, calcium imaging, multi-electrode recordings, etc.

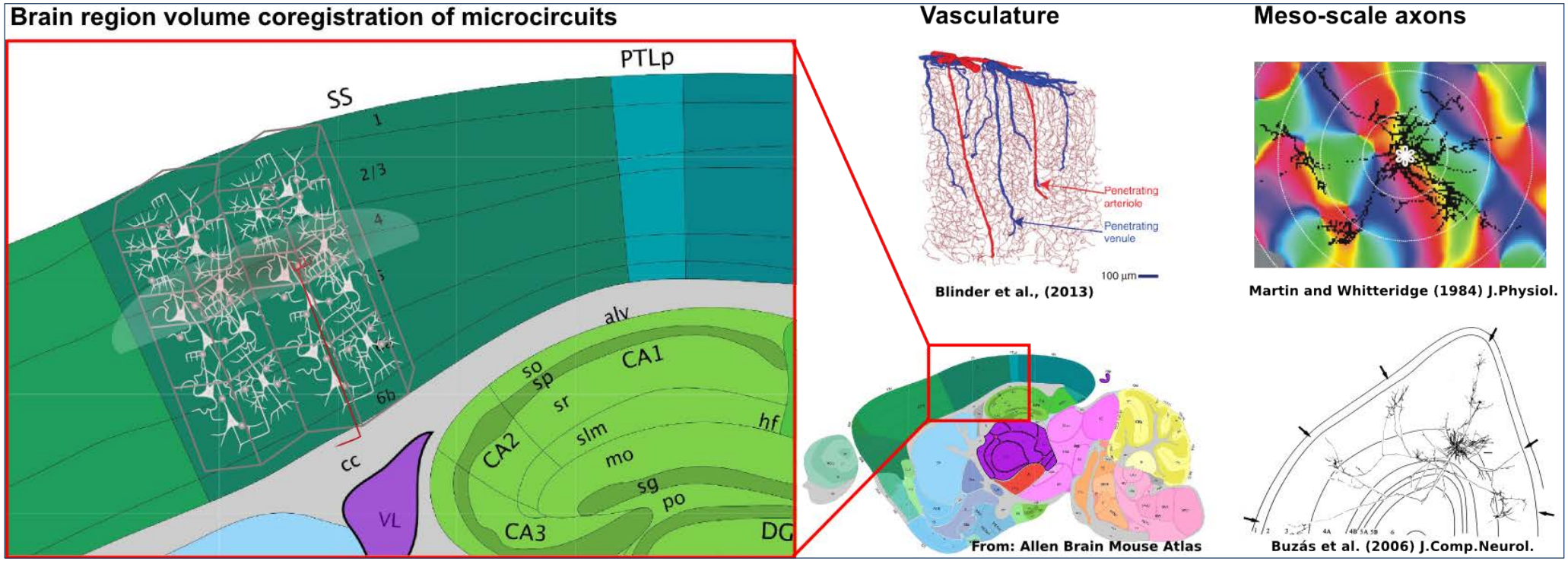

Figure 15: Schematic of the Meso-Level Reconstruction Process

8.5.2.1 Reconstruction Data

The following reconstruction data, specific to the meso-scale brain region to be modelled, are required for the reference reconstruction process:

- Micro-level reconstruction of the anatomy and physiology of the tissue (see Section 8.4).
- Neuron anatomy:
  - The micro-level reconstruction already provides samples of each m-type.
  - Additional reconstruction samples of representative cell types in areas with a high degree of curvature to guide transformations of morphologies.
  - Additional reconstructions to match variations in layer thicknesses throughout the region.
  - An alternative approach is morphology synthesis.
- Glial anatomy:
  - High-resolution light and/or EM reconstructions of glial cells.
- Architecture:
  - Multiple co-registered volumetric data sets providing information on variations in neuronal densities (e.g. NeuN stains).
  - Annotated landmarks of anatomical features such as layers and areas within a brain region (Allen Brain Mouse Atlas, Waxholm space).
- Cellular composition:
  - Neurons and glia stained using the clarity protocol and whole brain maps for cell counting and density estimation.
  - Ratios of excitatory to inhibitory cells (simultaneous NeuN and GABA co-staining).
  - Meso-level staining with neuron-type markers such as calcium binding protein and neuropeptide staining maps to guide gradients in the cellular composition.
  - Meso-level staining of a key subset of genes together with eventual single-cell transcriptomes that guide derivation of the cellular composition (i.e. the distribution of genetically identified cells that reproduce global staining maps).
- Meso-axonal projections:
  - Reconstructions of *in vivo* labelled axonal arbours >0.5mm (e.g. cortical PCs). The axonal arbours are analysed, including the number of meso-projecting axons, angles of projections, distances of projections, target layers, and extent of the end-terminal cluster).
- Vascular anatomy:
  - Synchrotron data of the ROI.
- Afferent and efferent projections:
  - Tract tracing data to estimated number of input and output fibres.
  - Tract tracing data to guide topographic mapping of afferent input fibres.
  - High-resolution light and/or EM data of afferent fibres; marking, counting and estimating Bouton densities along the input fibres.

- Synaptic physiology:
  - Data on synaptic physiology of meso-scale axonal connections where available; otherwise, infer from synapses in the microcircuit with the same presynaptic and postsynaptic pairs of neurons.
  - Where available, the physiology of input fibres; otherwise, apply some principles of synaptic transmission found in the microcircuit for glutamatergic input to a variety of postsynaptic neuron types.

8.5.2.2 Validation Data

Here we list example data sets describing observed properties at the target level that validate not only the tissue-level reconstruction, but also its component models, including:

1) Multi-electrode recordings (e.g. in cortex, synchronisation in the gamma band over d>1mm in awake juvenile rats[15].
2) Calcium or VSD imaging showing spatio-temporal patterns of activity (spontaneous and evoked) *in vivo* (propagation velocity of evoked waves, correlation structure in on-going activity, etc.).
3) Optogenetic stimulation experiments *in vivo.*
4) PC→all connections: selectivity maps for external projections, and rules for clustering of meso-scale axonal ramifications based on these maps.
5) Emergent properties caused by afferent input from other brain regions.

8.5.2.3 Reconstruction Process

The reconstruction process follows steps in a workflow. While different workflows can be invoked for specific cases, a reference workflow is:

1) Anatomy:
   a) If applicable, pre-process multiple co-registered volumetric data sets using suitable algorithms to extract required anatomical features such as geometrical primitives (e.g., pia and layer meshes).
   b) Place vasculature morphology.
   c) Generalise the following to 3D mesh or voxel regions where appropriate:
      i) Microcircuit geometry (e.g., layer thicknesses)
      ii) Recipe and soma placement algorithms (avoid collision with vasculature)
   d) Place morphologies. Transform or synthesise morphologies suitable for all parts of the meso-scale reconstruction departing markedly from the associated microcircuit geometry.
   e) Synthesise long-range meso-scale axons in an m-type specific manner consistent with less developed animals if stimulus-response maps are not known. Implement axonal ramifications according to observed rules (e.g. regions of like selectivities) if stimulus-response maps are known.
2) Synaptic physiology: constrained by reconstruction data.

8.5.2.4 Validation Process

Validation of meso-level models follows the generic validation process described in 8.2. Intrinsic validation is against the reconstruction data and target phenomena. Extrinsic validation is against the validation data provided above.

Implementations are required for automated *in silico* versions of the protocols used for validation data sets on emergent network behaviour and dynamics, such as calcium imaging, VSD, and multi-electrode recordings, to assess quantitatively the agreement between the validation data and the *in silico* experimental protocols undertaken on the meso-scale reconstruction.

8.5.3 Modelling Objectives

- Assess those network phenomena emerging at the meso-scale, which are absent in the microcircuit model.
- Assess the role of meso-scale axons in long-range synchronisation of neuronal populations in the gamma band in neocortex.
- Implementation of somatosensory cortex (M119), and hippocampus CA1 (M120) models.
- Provide the meso-scale brain region building blocks towards macro-scale tissue reconstruction (cortex, hippocampus, and cerebellum).

8.5.4 Relation to other Models, Milestones and Deliverables

Milestones directly involving meso-scale reconstructions include: somatosensory cortex (M119), hippocampus CA1 (M120) models, and cerebellum models (M122) of WP6.4.4 and WP6.4.5.

Molecular level modelling of neuromodulation and NGV interactions in parallel will drive refinement of meso-scale models.

8.5.5 Dependencies

8.5.5.1 Required

- BSP support for building and simulating meso-scale reconstructions.

8.5.5.2 Preferred

The following activities will be necessary to achieve the full potential of the model.

- Support in the BSP for automatic and systematic validation of emergent network behaviour at the meso-scale.

8.6 Macro-Level Models (Whole Brain/Brain System)

8.6.1 Overall Goals

An objective of WP6.4 exemplified by Milestone MS122 is the development of a proof-of-concept generic reconstruction strategy to build entire brain areas. Rodent full neocortex and hippocampus will be reconstructed. Macro-level models are component models towards a whole-brain model.

The challenge of macro-scale model building lies in utilising the indirect, large-scale and brain global data sets to infer a dense connectome, with sufficiently constrained micro-scale reconstruction parameters between meso-scale models of brain volumes.

8.6.2 Model Description

Macro-scale models consist of component meso-scale models for each brain region within the brain system of interest, and reconstructing their connectome by inferring parameters

from macro-scale connectome data (white matter) due to large-scale tracer studies, diffusion-tensor imaging (DTI), myelin stains, etc.

As detailed microcircuit data sufficient to densely parameterise all component meso-scale volumes of a given macro-scale brain area are expected to be lacking, rich data mining and analysis techniques to infer the necessary parameters from global staining maps will be developed (e.g. inferring me-type composition via single cell gene expression). These techniques. These techniques will be validated on available meso-scale models to predictively complete a dense data set on the anatomy and physiology of local microcircuitry.

A non-exhaustive list of the parameters required to reconstruct efferent projection innervations includes bouton/axon density profiles, lateral extents per fibre and density of incoming fibres per bundle. Innervation profiles in target volumes yielded by macro-scale data sets (large-scale tracer studies, DTI, myelin stains) do not directly yield the required parameters; rather, they yield the density of post-synaptic morphological arbours targeted by the projection, as the anterograde tracers employed cause the targeted post-synaptic cell to become florescent, thus obfuscating the anatomy of the projection axons themselves. The strategy of solving the inverse problem of inferring the required reconstruction parameters from knowledge of tracer fluorescence profiles, and from the microcircuit morphological composition, will be evaluated for feasibility. The approach will be validated with sparse but rich microcircuit-level data on innervation anatomy in, for example, microcircuit models of the somatosensory cortex previously developed by the Blue Brain Project. Assuming that, through this work, the tracer fluorescence profiles can subsequently be classified into projection types (p-types) mapped to inferred microscopic reconstruction parameters, the result would be a global and dense parameter set sufficient for detailed reconstruction of post-synaptic innervation, as depicted below.

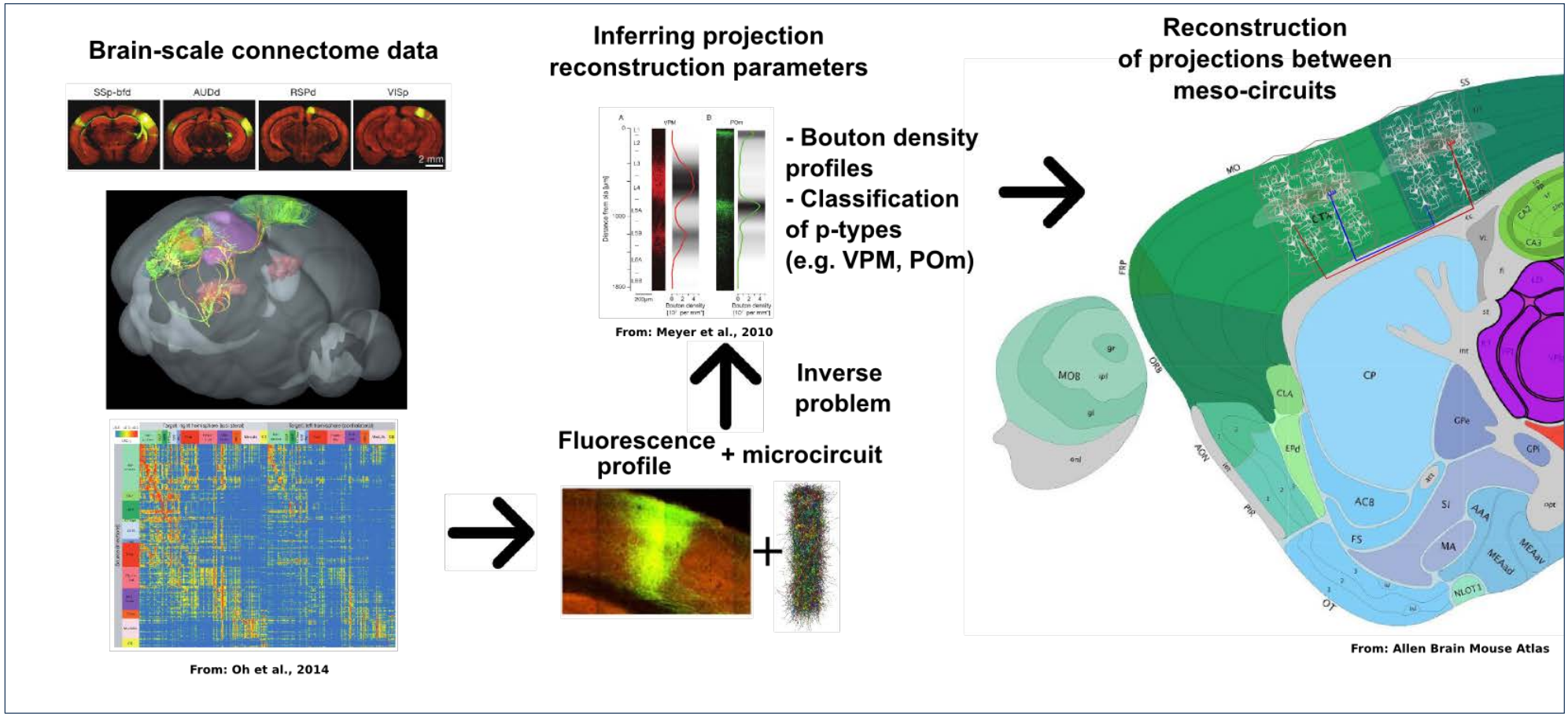

Figure 16: Schematic of the Macro-Level Reconstruction Process for Whole Brain Connectivity

Conversely, large-scale retrograde labelling data sets would be required to identify source neuronal population/m-type if they cannot be inferred from the p-type by other means (e.g. sparse data in literature).

Knowledge of the pre-synaptic population of a given projection constrains the density of fibres making up the projection bundle in the white matter. Further constraints on the density and trajectory of fibre bundles in white matter can be gleaned from DTI, myelin stains, etc. If these constraints are combined with a representative population of reconstructed axon morphologies in the projection zones for each p-type and their bouton densities per unit length, the problem would be sufficiently constrained to realise a first draft reconstruction.

8.6.2.1 Reconstruction Data

The following reconstruction data, specific to the meso-scale brain region to be modelled, are either required or preferred for the reference reconstruction process.

Required:

1) Anatomy
    a) Meso-scale models of ROIs; generic meso-scale models for ROIs where specific data are unavailable.
    b) Large-scale anterograde and retrograde tracer studies (e.g. Allen Brain Mouse Connectome Atlas).
    c) Microcircuit studies of bouton density profiles (e.g. Meyer *et al.*, 2010 for rat barrel cortex[16]) for each p-type.
    d) Representative population of reconstructions and bouton densities (per unit length) of axonal morphology in the projection zone for each p-type.

Preferred:

1) Anatomy
    a) Single-cell gene expression and global staining maps.
    b) Diffusion tensor imaging (DTI) of macroscopic organisation of fibre tracts.
    c) Large-scale systematic myelin staining.

8.6.2.2 Validation Data

Data sets suitable for validation of macro-scale models are generally of emergent properties of network dynamics, including:

1) Physiology
    a) *In vivo* vs. *in silico* fMRI.
    b) *In vivo* vs. *in silico* EEG.
    c) Same test and validations as for the meso-circuit.
    d) Paired recordings between brain regions to assess their connection strengths.

8.6.2.3 Reconstruction Process

The reconstruction process follows steps in a workflow. While different workflows can be invoked for specific cases, a reference workflow is:

1) Anatomy

a) Populate all neuronal volumes with their respective region-specific meso-scale circuits, where available, and with area-specific generic circuits where not available (e.g. use a generic "average" cortical circuit where detailed microcircuit data are unavailable).

b) First pass projectome: the addressing system of projections between meso-level brain regions:

- Analyse large-scale anterograde tracer data sets to identify the source and target meso-scale region for each projection
- Analyse large-scale retrograde tracer data sets to identify pre-synaptic neuronal sub-populations from which identified projections originate
- Solve the constraint problem—taking into account projection source population neuron density, white matter fibre density, DTI and myelin staining data on fibre tract trajectory—to yield coarse geometry and density of fibres mapping source neuronal populations to target region projection fibres.
- The result of this process is a list of projecting cell types by layer and by region; their target region(s); and the coarse fibre density, numbers and geometry between regions.

c) Micro-level reconstruction of projections: reconstruction of the micro-level afferent innervation patterns for identified region-to-region projections:

- Analyse large-scale anterograde tracer data sets to extract projection classes (p-types).
- For each p-type, solve the inverse problem, inferring from fluorescence and microcircuit the bouton density profile required for reconstruction (as discussed in meso-scale model section).
- Analyse projection fibre morphological reconstructions to establish lateral profiles.
- Innervate each meso-scale region with the parameters determined by the above steps. Prototype tools for this purpose have been demonstrated previously by the Blue Brain Project. Connect such meso-scale innervations to presynaptic source neuron populations in a topographic manner as defined by the projection fibre bundles.

NB: The above process is given for macro-scale connectivity in cortex. Analogous workflows would need to be established for hippocampus, where projection anatomy takes different geometrical forms.

8.6.2.4 Validation Process

Validation of macro-level models follows the generic validation process described in 8.2. Intrinsic validation is against the reconstruction data and target phenomena. Extrinsic validation is against the validation data provided above.

Implementations are required for automated *in silico* versions of the protocols used for validation data sets on emergent network behaviour and dynamics—such as *in silico* fMRI or EEG, calcium imaging, VSD, and multi-electrode recordings—to assess quantitatively the agreement between the data and the *in silico* experimental protocols undertaken on the macro-scale reconstruction.

8.6.3 Modelling Objectives

Achievement of macro-models of whole cortex and hippocampus provides the foundation for investigating a wide range of fundamental open questions in neuroscience and its relationship to cognitive science, psychophysics and psychology. A somewhat arbitrary selection of examples is listed here:

1) A widely held paradigm for cortical processing is that it occurs in a hierarchical manner, with the cortical laminar structure featuring prominently in the architectural organisation of the hierarchy[17]. The way this cortical hierarchy processes information in detail has not yet been fully described, nor has cortical laminar architecture played a role in this processing hierarchy. Understanding at this macro-architectural level would provide insight into the functional purpose of specific microcircuit organisation (e.g. the role of Martinotti cell L1 reaching axons and pyramidal cell apical dendritic tufts).
2) Short- and long-term memories of sensory experiences are thought to involve an interaction of neocortex and hippocampus, but the mechanisms behind this interaction are poorly understood. A macro-scale model of whole cortex and hippocampus would provide unique tools to drive our understanding further in this open area of research.
3) Association and binding of sensory percepts, distributed throughout the local modality-specific sensory regions, has been hypothesised to arise through long-range synchronising oscillations in the gamma band (30-80Hz). A macro-scale model of distributed cortical sensory processing would allow detailed study of the mechanism behind sensory binding, providing an opportunity to confirm or disprove the "binding by gamma" hypothesis.

8.6.4 Relations to other Models, Milestones and Deliverables

Macro-level models of full cortex and hippocampus integrate meso-scale models of somatosensory cortex (M119), and hippocampus CA1 (M120), and thus inherit all of the specific dependencies of those milestones.

8.6.5 Dependencies

8.6.5.1 Required

The following activities are required to satisfy that the model can be generated by and used in the Brain Simulation Platform.

1) Data requirements:
   a) Large-scale retrograde data would need to be collected.
   b) Representative populations and bouton density estimates of axonal reconstructions for each identified p-type.
   c) Detailed experimental characterisation of bouton density profile for each identified p-type (see Ref 16).
2) In addition to the requirements for meso-scale models, the Platform must have the following functionality for macro-scale models:
   a) Data analysis tools to extract the inter-brain-region connectivity map from the various data sets, classify p-types, and solve the inverse problem of inferring p-type bouton density profiles from fluorescence profiles and morphological composition of the microcircuit.
   b) Data analysis tools to constrain quantitative parameters of projection fibre bundles from, for example, DTI, fMRI, and myelin stain data sets.

8.6.5.2 Preferred

The following activities will be necessary to achieve the full potential of the model.

1) Data requirements:
   a) Single cell gene expression and global staining maps for densely inferring cellular composition at the whole-brain scale.
2) Platform requirements:
   a) The development of reconstruction and validation processes would benefit a flexible API framework allowing for analysis of simulation outputs, and of neuron or macro-circuit properties.

8.7 Molecular-Level Models (Intracellular)

8.7.1 Overall Goals

The HBP aims to develop generic strategies to integrate molecular details into model neurons, synapses and glia. The models must be reconstructed to match experimental data, reproduce cellular and molecular level phenomena, predict experimentally verifiable interactions, and guide new experiments. Since the vast majority of the initial data parameters are missing, the models must be designed to predict missing biological data. They must also be designed for iterative refinement in cycles of reconstruction, simulation and validation based on new data and on new principles of organisation. In this way, molecular models of neurons, synapses and glia can undergo collaborative refinement towards highly accurate representations of the biological design.

Since the vast majority of biological parameters are missing (e.g. list of proteins involved, concentrations and distribution of each protein, reaction partners for each protein, sub-states of each protein, reaction kinetics, etc.) we developed a strategy where fundamental biological principles are used to constrain parameters. This strategy relies on multi-constraint rules to narrow down the molecular parameter space. The result is a hypothesis of the parameter space that can be tested experimentally.

Reaction kinetics for proteins and protein complexes is one of the largest and most important missing data sets. WP6.3 aims to fill this gap using molecular dynamics modelling, docking and simulation. In the Ramp-Up Phase, WP6.3 is providing protein interaction models, and kinetic and thermodynamic parameters for selected proteins as needed in the neuronal pathway modelling. In the long run, the MolSim User will obtain estimates of molecular-level parameters via a semi-automatic procedure, under the guidance of the WP6.3 staff.

Since the molecular cascade caused by the activation of receptors results in the modulation of ion channels and consequent changes in the electrical discharge properties of neurons, we will develop initial draft molecular models focusing on the intracellular signalling cascades triggered by the concomitant activation of receptor-induced cascades, and the interaction between them.

These interaction cascades are present in all neurons and will thus be used as the exemplary approach to the “molecularisation” of neurons, synapses and glia. These models will also serve as a starting point from which variations can be introduced for different types of cells. Since these models also provide a means by which neuromodulators can affect membrane currents, simplified versions of these interactions can be used for neuromodulation of larger microcircuit, mesocircuit and whole brain activity. Finally, this generic modelling of the interactions will provide a first test case for constraining model

parameters through molecular simulations, to be performed in WP6.3. In particular, the first computational predictions of the G-protein dependent activation of adenylyl cyclase will be provided.

8.7.2 Model Description

Figure 16 represents a reaction network composed of nodes (species) and edges (reactions), which is mathematically expressed as a system of ordinary differential equations.

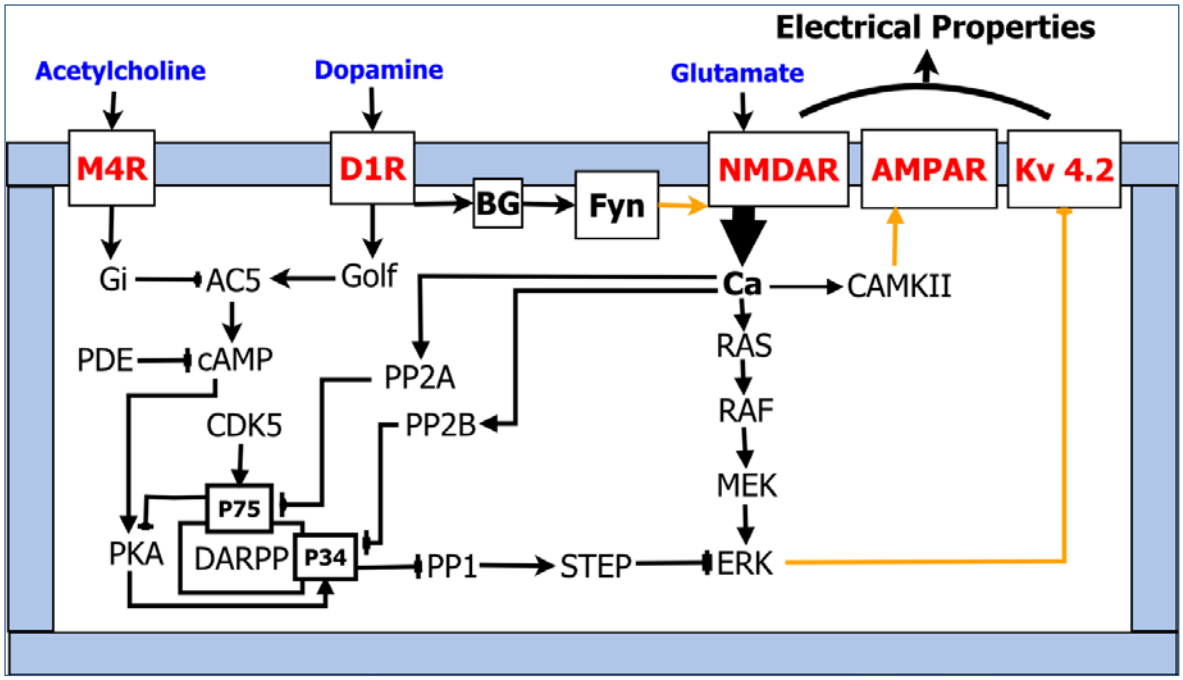

Figure 17: Schematic of a Molecular-Level Reconstruction of a Synapse

8.7.2.1 Reconstruction Data and Parameter Sets

The set of molecular parameters is a) drawn from a biological data set where available, b) predicted where not, and c) further constrained during the reconstruction process as fundamental principles are applied. The data and parameter sets will refine with each reconstruction cycle as predictions are verified or falsified. The initial reconstruction parameter sets include:

1) Receptors:
   a) Target set of receptors (specific subtypes of acetylcholine, dopamine, glutamate receptors).
   b) Absolute and relative concentrations.
   c) General distribution along the neuron, synapse or glial (concentration gradients).
   d) Specific localisation (in sub-compartments such as a spine head, organelle, presynaptic density, etc.).
   e) Mobility of the molecule (fixed or diffusing).
   f) Specific co-localisation (anchored, clustered) with other molecules involved.
   g) Sub-states of each receptor.
   h) The molecules that each receptor interacts with in each sub-state.

i) The reaction kinetics of each interaction (Kf, Kb and Kd).

j) The chemical equation for each interaction (e.g. AB->C+2D).

k) Dose-dependent ligand-activation biophysical properties.

l) Changes in electrical properties when one or more receptors are activated.

2) Signalling molecules:

a) Target set of molecules involved in the signalling pathway (focusing on the kinases and phosphatases at this stage).

b) The same data and parameter set as for receptors above 1b)-j).

3) Ion channels and transporters:

a) Set of ion channels/transporters

b) Distributions

c) Absolute and relative concentrations

d) Kinetics

e) Conductances

f) The extent to which channel/transporter amounts, kinetics and conductances are affected by receptor-mediated signalling.

4) Neurons:

a) Digital reconstructions of a neuron morphology (including spines on the postsynaptic neuron), or

b) Morphologies extracted after a microcircuit has been reconstructed (see below) with derived spines and the full set of synaptic loci.

c) The set of receptors, signalling molecules and ion channels specific to the type of neuron.

5) Synapses:

a) Generic synapse mesh with artificial boutons and postsynaptic specialisations such as spines, or

b) Extracted after microcircuit reconstruction with derived locations of synapses and synthesised boutons and postsynaptic specialisations such as spines.

c) The set of receptors, signalling molecules and ion channels specific to the type of synapse.

6) Glia:

a) Digital reconstructions of glial morphologies, or

b) Glial morphologies extracted after microcircuit reconstruction with derived geometrical relationship of glial processes with neurons, synapses and vasculature.

c) The set of receptors, signalling molecules, ion channels specific to the type of glia.

Each reconstruction cycle challenges the accuracy of the data set and the validity of the parameter set, driving iterative refinement of the reconstruction data and parameter sets.

8.7.2.2 Validation Data

Here we list example data sets describing observed properties at the target level that not only serve as validation for the molecular-level reconstruction, but also for its component models, including:

1) Neurons:
   a) Changes in active and passive properties when one or more receptors are activated.
   b) Time courses of phosphorylation and de-phosphorylation.
   c) Activation profiles of target proteins.
   d) Concentration profiles of associated biochemicals (e.g. changes in intracellular calcium).
2) Synapses:
   a) Any of properties b) to e) listed above for neurons.
   b) Changes in synaptic transmission following the activation of one or more receptors.
3) Glia:
   a) Any of properties b) to e) listed above for neurons.
   b) Changes in interactions observed between neurons, synapses, and vasculature.

8.7.2.3 Reconstruction Process

The reconstruction process follows steps in a workflow. Different workflows can be invoked. An example workflow is:

1) Select the detailed morphology (type of neuron, whole neuron, synapse, glial cell, part of a dendrite, a dendritic spine),
2) Select a pre-existing electrical model of the neuron, synapse or glia providing a configured set of ion channels.
3) Perform computational synthesis of the subcellular elements (organelles, intra and/or extracellular spaces) to account for the internal geometry.
4) Select a set of receptors, signalling and other related molecules.
5) Select sub-states of each molecule.
6) Configure binding kinetics for receptors and signalling molecules for each sub-state.
7) Select absolute densities for each molecule and providing any relative density constraints between molecules.
8) Set the distribution of each molecule,
9) Set the localisation parameters of each molecule.
10) Select the set of reaction partners for each molecule,
11) Set the reaction kinetics for each interaction,
12) Select the experimental data on the target response from the reconstruction data set,
13) Analyse the target response and extract a feature set (target features).
14) Adjust free parameters to match the target response, using the applicable case below:

a) For models where ion channels were not preconfigured, freeze all settings and adjust only the ion channel density profile – the target response is the electrical discharge response.

b) For models with given electrical behaviour and only one receptor type, freeze all settings and adjust only the density profile of signalling molecules – the target response is the signalling profile.

c) For models with multiple receptors, freeze all settings and adjust only the relative density of receptors – the target response is the changes in electrical discharge.

15) Adjust the free parameter vector using an established multi-objective optimisation (MOO) process.

8.7.2.4 Validation Process

Validation of molecular-level models follows the generic validation process described in 8.2. Intrinsic validation is against the reconstruction data and target phenomena. Extrinsic validation is against the validation data provided above.

8.7.3 Modelling Objectives

1) Determine what is required to induce local vs. global neuromodulatory effects of a neuron, i.e. the relationship between synaptic inputs on proximal/distal dendrites and resulting modulation effects, etc.

2) Determine what quantitative effects local neuromodulation has on synaptic integration.

3) Determine how co-activated cascades interact with respect to their ultimate effects on membrane proteins.

4) Use validated models to create hypotheses for the underlying mechanisms controlling the observations in 3) above.

8.7.4 Relations to other Models, Milestones and Deliverables

The planned work will drive the reconstruction process at the molecular level. This model work will consume outputs (predicted reaction kinetics) from WP6.3. The process will be extended to include progressively more molecules, allowing more biological, electrophysiological and pharmacological phenomena to be captured. In the Operational Phase, a generic strategy to “molecularise” cells (neurons, glia and synapses) will be provided. The results of these models will drive the refinement of simplified models of the interactions between receptors and ion channels, enabling the simulation of neuromodulation in meso-level models (brain region, Task 6.4.5) and macro-level models (whole brain, Task 6.4.6).

The Molecular Simulator and the initial molecular-level models of neurons, glia and synapses require kinetic parameters for the mathematical description of molecular interactions. As pointed out in Section 7.1.1, the vast majority of parameters that depend on molecular interactions are missing. WP6.3 uses molecular simulation and bioinformatics tools to estimate and constrain the missing parameters. Consistent with the structural information available for the target biomolecule (either from the Protein Data Bank or from homology modelling), a range of molecular simulation techniques may be applied to investigate protein-protein association, ligand binding, enzymatic catalysis and diffusion.

During the Ramp-Up Phase, WP6.3 is working in strict collaboration with the developers of the initial molecular-level models of neurons, glia and synapses to provide parameters for the modelling of the G-protein dependent activation of adenylyl cyclase. In the longer

term, WP6.3 will provide kinetic and thermodynamic data of biomolecular interactions to supplement the Reaction Database used by MolSim. In this sense, WP6.3 will provide the data required to setup and run MolSim. Targets for molecular simulations and bioinformatics analysis to expand the Reaction Database will be decided in collaboration with the scientific developers of the brain models. Eventually, a largely automated Task will enable the MolSim Users to run molecular simulations in order to obtain the Target of interest. WP6.3 will provide guidance for the setup, running and analysis of the simulations.

In addition to the kinetic parameters, atomistic modelling will unravel the molecular details of interaction processes and reactivity, providing valuable insight for the design of ligands capable of interacting with the target biomolecule, and thus enabling new experiments.

8.7.5 Dependencies

8.7.5.1 Required

The following features are required to ensure that the model can be generated by and used in the BSP:

- The BSP must be able to read in data describing the presence and distribution of various proteins and their reactions.
- The BSP must be able to read in and run/simulate models described in SBML format. It must be possible to simulate the resulting models both in the cellular simulator (NEURON based) as well as in the molecular simulator (STEPS based). The latter must be able to run a specific model in both a stochastic and deterministic simulation mode.
- The validation data sets, as exemplified above, need to be produced.

8.7.5.2 Preferred

The platform cellular level and molecular level simulation software should be able to support co-simulations, where a biochemical signalling network model can interact during runtime with the electrical level of the model.

8.8 Models of Neuro-Glia-Vasculature Interaction

8.8.1 Overall Goals

We aim to develop generic strategies to integrate molecular details into models of neuro-glia-vasculature (NGV) coupling and to develop a successively more detailed series of NGV metabolism and coupling models. Ultimately, this work will provide a better understanding of how local neural activity results in changes in blood flow to regions of the brain, and the homeostatic role of glial cells, whose dysfunction could be related to a number of brain disorders including migraines and specific forms of epilepsy. The models must be reconstructed to match experimental data and to guide new experiments.

As the vast majority of biological parameters are missing or are poorly characterised (e.g. metabolite exchange affinities, list of proteins involved, reactions and concentrations and distribution of each protein, reaction partners for each protein, spatial structure of glia, etc.), the models must be able to accommodate uncertainties in the parameters while improving their predictions as the quality of the data improves.

The first Iteration will focus on single-compartment models of regions of space encompassing neurons, glia, and vasculature, in which fine spatial structure of the component entities is not explicitly represented. These models will capture the reaction

networks underlying glial metabolism and use of glucose, lactate, and oxygen, as well as factors leading to vasodilation and vasoconstriction. They will also track external concentrations of ions and metabolites. The next generation will include spatial features such as multi-compartment glia cells and explicit vasculature networks. We will also include the ability to export highly detailed spatial models of the NGV. Finally, we will close the loop to link neural activity with blood flow.

8.8.2 Model Description

The following diagram depicts a reaction network of metabolism in the NGV, which is mathematically expressed as a system of ordinary differential equations.

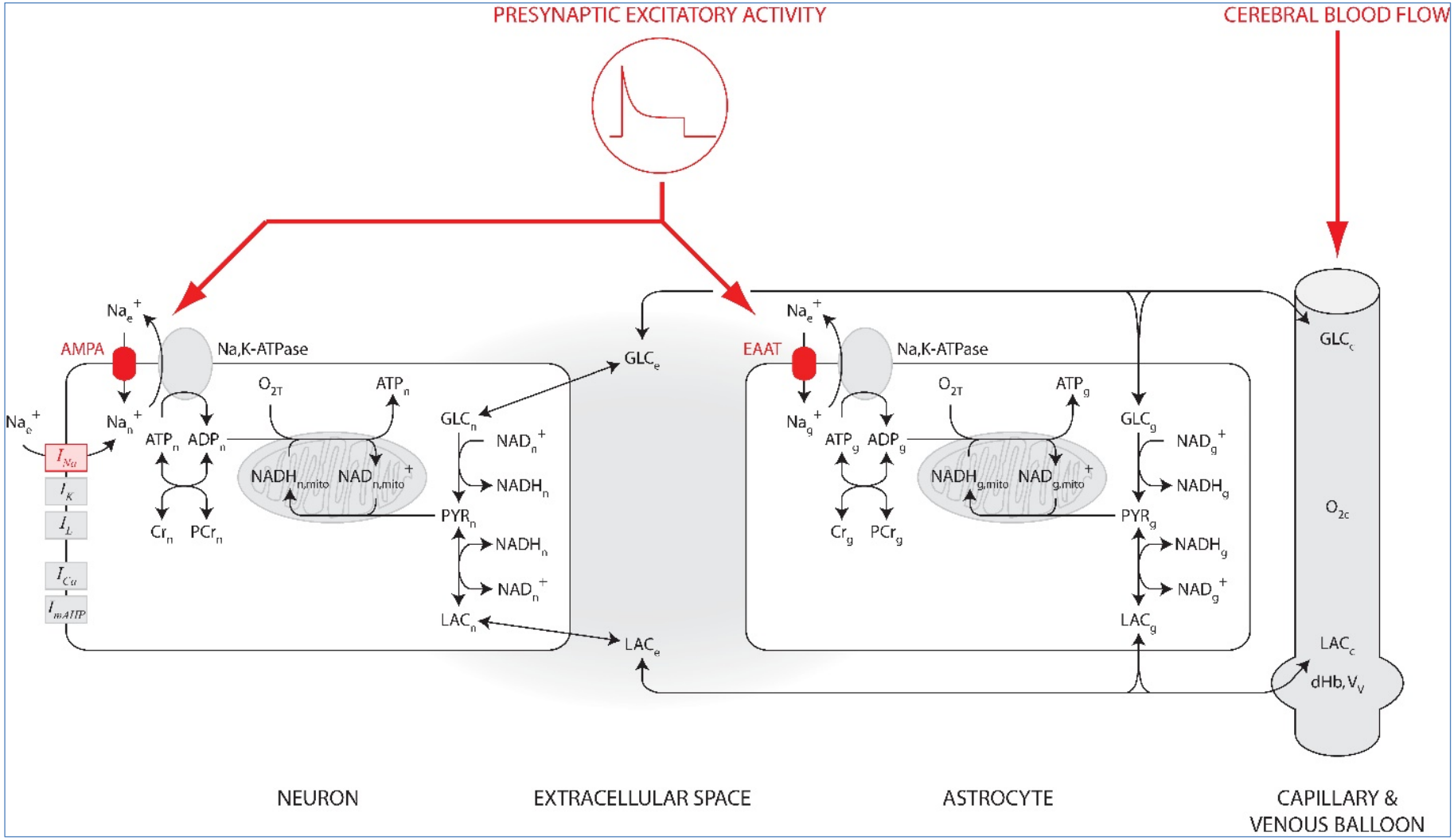

Figure 18: Schematic of the NGV Reaction Network

Neuron concentrations of glucose and lactate are modified by neuronal activity and the TCA cycle within neurons. Glucose and lactate are exchanged between the extracellular space and neurons, glia and capillaries.

8.8.2.1 Reconstruction Data and Parameter Sets

The set of NGV parameters is a) drawn from a biological data set where available, b) predicted where not, and c) further constrained during the reconstruction process as fundamental principles are applied. The data and parameter sets themselves will be refined with each reconstruction cycle as predictions are verified *or* falsified. The initial reconstruction parameter sets include:

1) Channels:
    a) Sodium, potassium, and calcium channels on neurons (used in electrical circuit).
    b) Sodium channels on glia.
2) Ion transporters: Na+, K+ ATPase.
3) Metabolic pathways involved in TCA handling of:
    a) Glucose
    b) Pyruvate
    c) Lactate
    d) ATP/ ADP/ NADH/ NAD+.
4) Ions and Metabolites: Measured concentrations.
5) Vasculature:
    a) Equations governing O_2 and blood flow in capillaries.
    b) Morphology of the capillary network.
6) Synapses: Locations of synapses relative to glia.
7) Glia:
    a) Digital reconstructions of glial morphologies, or
    b) Morphologies derived from microcircuit reconstruction and with the appropriate geometrical relationship of glial processes with neurons, synapses and vasculature.
    c) The set of receptors, signalling molecules, ion channels specific to type of glia.

Each reconstruction cycle challenges the accuracy of the data set and the validity of the parameter set, driving iterative refinement of the reconstruction data and parameter sets.

8.8.2.2 Validation Data

Here we list example data sets describing observed properties at the target level that serve as validation not only for the reconstruction, but also for its component models.

2) Neurons:
    a) Changes in channel currents and signalling in response to pharmacological manipulation.
    b) Activation profiles of target proteins.
3) Glia:
    a) Any subset of the reconstruction data.

b) Measurements of tissue volume filled.

c) Calcium imaging in glia.

4) Vasculature:

a) Any subset of the reconstruction data.

b) Measurements of tissue volume filled.

c) EM data.

5) Tissue:

a) Temporal dynamics of lactate and oxygen and other metabolites.

b) BOLD signal.

8.8.2.3 Reconstruction and Simulation Process

The reconstruction and simulation process follows steps in a workflow. The following is the reference workflow:

1) Create morphological representations of neurons, glia, and vasculature.

a) These can derive directly from reconstructions or be synthesised based on statistical measurements.

b) Fine-scale mesh models can be constructed from these morphological models at this stage, as inputs to separate molecular simulations.

2) Voxelise circuit space into cubic domains. Parts of neurons, glia, and vasculature will fall within each domain.

3) Set up NGV equations:

a) Decide which molecules to use for neurons and glia.

b) Input their reactions and kinetics.

c) Designate molecule concentrations and densities.

d) Add additional relationships to equations not captured by reaction network.

e) Discretise NGV equations (voxelisation).

4) Map neuron activity to the NGV model. Currents in neuronal sections are mapped to each voxel and coupled to the NGV model. This will enable driving of metabolic pathways and tracking of ion concentrations in voxels.

5) Map synaptic activity to the NGV model. Neurotransmitter concentration will be tracked in each voxel and used to drive metabolism.

6) Blue gene simulation:

a) Couple the running circuit simulation to the NGV model solver at long intervals (relative to the neuron voltage solver time step) defined by diffusion constants, for example.

b) For each such time interval:

– Exchange current information from circuit with voxels.

– Exchange cumulative neurotransmitter release with voxels.

– Run the metabolic NGV simulation. Simulate diffusion of molecules in glia structure, if applicable.

- Diffuse ions and metabolites between voxels.
- Update vasoconstriction/dilation information and run blood flow simulation, if applicable.

7) Visualise and analyse the results.

8.8.2.4 Validation Process

Validation of molecular-level models follows the generic validation process described in 8.2. Intrinsic validation is against the reconstruction data and target phenomena. Extrinsic validation is against the validation data provided above.

8.8.3 Modelling Objectives

1) Determine how the neuronal activity affects metabolic processes within neurons and glia.
2) Determine how this links to overall energy usage.
3) Determine how neuronal activity couples to blood flow.
4) Identify the contributions of different neuronal types to the above processes.
5) Determine how the spatial organisation of glia and vasculature affects the above processes.
6) Identify the role of NGV interactions in the homeostatic regulation of neuronal activity.

8.8.4 Relations to other Models, Milestones and Deliverables

The planned work will drive the reconstruction process at the molecular and structural levels of the NGV. This model work will use the outputs (predicted reaction kinetics) from WP6.3, and vasculature morphology from WP1.2. The process will be extended to include progressively more proteins, metabolites and ions, allowing the capture of more biological, electrophysiological and pharmacological phenomena. The results will also be used to implement volume transmission of neuromodulators in meso-level models (brain region, WP6.4.5) and macro-level models (whole brain, WP6.4.6).

This model directly supports Milestone MS122 by producing initial models of NGV coupling in cerebellum, full neocortex and hippocampus. Some modification of the model will be necessary to adapt it to different brain regions, as cellular composition and geometry vary according to location.

8.8.5 Dependencies

8.8.5.1 Required

The following activities are required to ensure that the model can be generated by and used in the BSP.

1) The BSP must be able to read in data describing presence and distribution of various proteins and reactions between the proteins.
2) The BSP must be able to read in and run/simulate models described in SBML format.
3) The BSP must be able to simulate reactions and diffusion in the extracellular space throughout the entire brain.

8.8.5.2 Preferred

The following activities will be necessary to achieve the full potential of the model.

1) The Platform should support some modelling of blood flow through the reconstructed vasculature network, coupled to the NGV model.
2) The Platform cellular level and molecular level simulation software should be able to support co-simulations, where the NGV model can interact during runtime with the electrical level of the model to influence activity of the network.

9. Key Performance Indicators (KPIs)

9.1 Software Development Methodology

One the goals of Human Brain Project regarding the ICT Platforms is "to demonstrate how the Platforms could be used to produce immediately valuable outputs for neuroscience, medicine and computing"[18]. To achieve this goal, the Platform teams must produce the right software for the scientific customer, taking into account the classic project management triangle of quality, cost and delivery speed. Because many of the requirements for scientific tools are not known ahead of time, building the right software can only be achieved cost-effectively through close collaboration between software development teams and scientific customers. This challenges classical “big design first” software engineering process models.

The Agile movement grew out of dissatisfaction with existing software development methodologies and attempts to provide a viable approach for the cost effective delivery of complex and risky software projects. The principles are captured in the Agile Manifesto (http://agilemanifesto.org/):

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

There are many methodologies that are built on the principles above. Scrum, Kanban and Extreme Programming have been used with great success in many organisations. Each has a well-established discipline, with clearly defined strategies for adoption and adaptation. For reasons of team experience and alignment with current team structure, the COLL and BSP are being developed using the Scrum Agile methodology.

9.2 Scrum: Roles and Procedures

In Scrum, a shippable increment of the NP is produced at the end of each Iteration (also known as a 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 sprint-planning meeting for identifying and estimating Tasks at the beginning and a sprint review meeting for progress monitoring and future prospects at the end of each sprint.

For the COLL and BSP, it means that at the end of each Iteration there will be a usable version available for the Users to use or test. Research groups and academic partners that don't need all the features of the Platform will 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, 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 as an ordered list of requirements.

The Product Owner will prioritise each story in consultation with Stakeholders and Users. At the beginning of each Iteration, the team will look at the stories with the highest priority and decide which of them can be implemented in the Iteration. At the end of each Iteration, the team will demonstrate to the Product Owner and the Users that the stories are completely implemented (tested and documented).

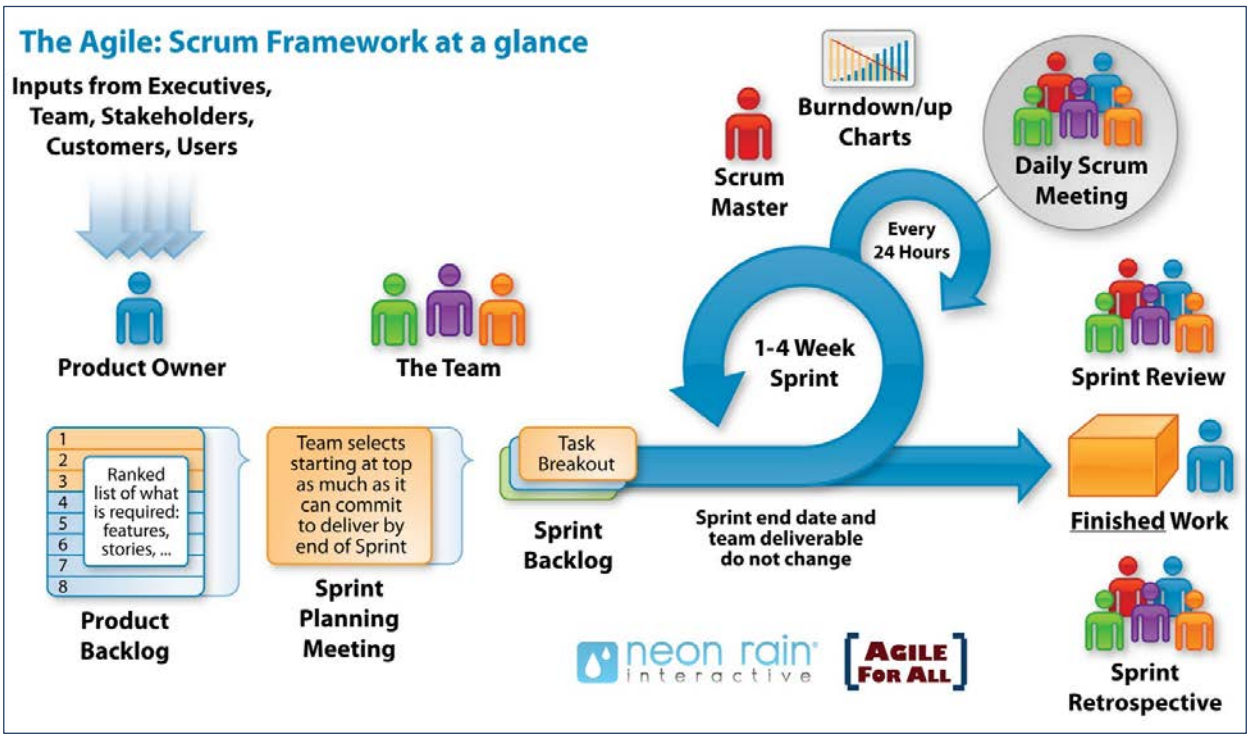

Figure 19: Agile Scrum Iteration Workflow: from the Product Owner Definition of the Backlog to a Finished Product

The Scrum process model outlined above enables high adaptability to the Users' needs. It is also inherently able to deal with changes in Users' requirements: Let one User realise, for example, that a new feature will be required that 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 regardless of the feedback, the Product Owner can take quick concrete actions in order to satisfy the Users' needs.

9.2.1 Scrum Review

One of the goals of working with small Iterations is to have quicker feedback from the Users than with a longer 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 Iteration. It will allow the Users as well as the stakeholders to see the software 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. Once the MVP state is reached, more Users will begin testing the Platform. Throughout the duration of the development, the stakeholders and the team will make sure that enough advertisement of the Platform is done in order to attract beta Users.

The feedback of the Users will be addressed during every Iteration when the Backlog is discussed among the stakeholders and the Product Owner. Integrating this feedback is vital for the Platform. It will reduce the risk of the final delivery failing to correspond to the Users' expectations (see Figure 19).

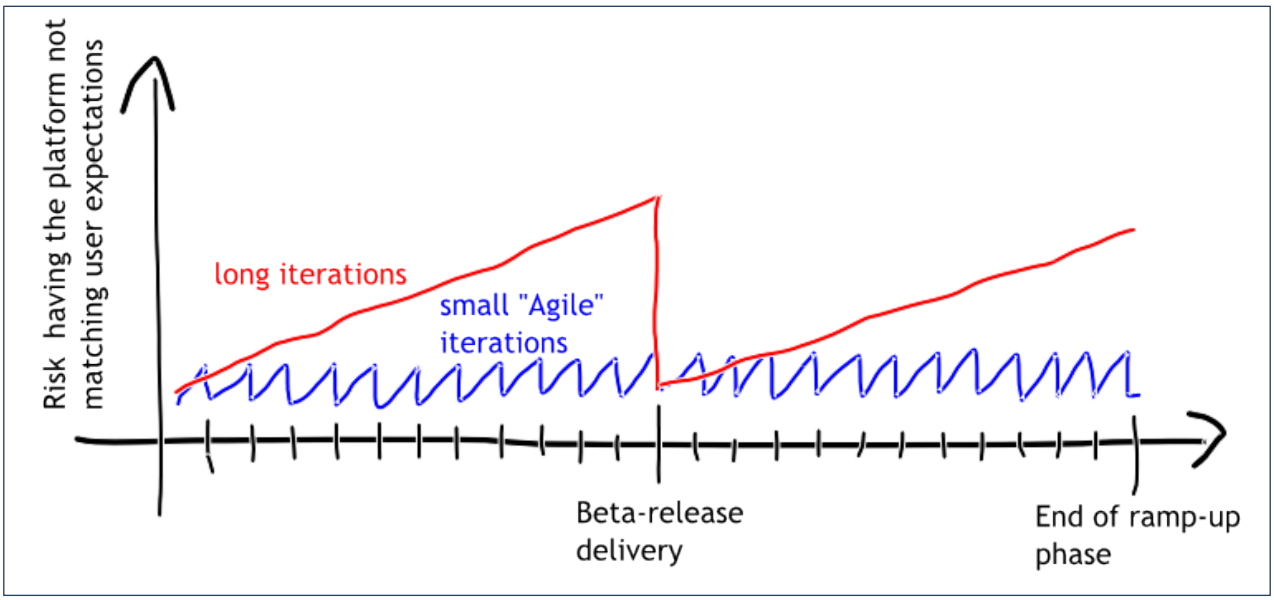

Figure 20: Risk Reduction using a Short Feedback Loop with Users.

For every Iteration, the Development Team will present the new features developed during the Iteration in a demo-oriented presentation. Stakeholders and Users will be invited to this presentation.

9.2.2 Backlog

In the Scrum methodology, the Backlog is the container of all Backlog items. Backlog items are usually User Stories that deliver User-visible functionality. However, they might also be bugs to be fixed or tasks to be performed. Every Backlog item has an effort estimate and a priority. Backlog items are grouped into Epics that are collections of stories needed to deliver a larger piece of functionality. An Epic might collect all items required to configure and launch simulations from the COLL.

The Backlog will be available to the stakeholders at all times. The Backlog will be regularly reviewed by the Product Owner and the stakeholders to determine whether some modification to the Backlog contents or priorities have to be made. There are several possible types of modifications:

- A change of priority in the Backlog: For example, one of the Users needs a specific feature to make a demonstration of the software during a conference. The Product

Owner, in consultation with the stakeholders, 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 of the Users needs a feature for his project that is not contained in the Backlog. This User can ask the Product Owner to prioritise it.
- A removal of User Stories: After having added needed User Stories to the Backlog the Product Owner realises that the release will not be delivered on time. They can decide to remove the less needed stories of the release Backlog in order to make sure that the release is delivered on time. These stories will be put back in the request Backlog. They can also be dropped if there 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. Regular re-estimation will improve the overall confidence for meeting the deadlines.

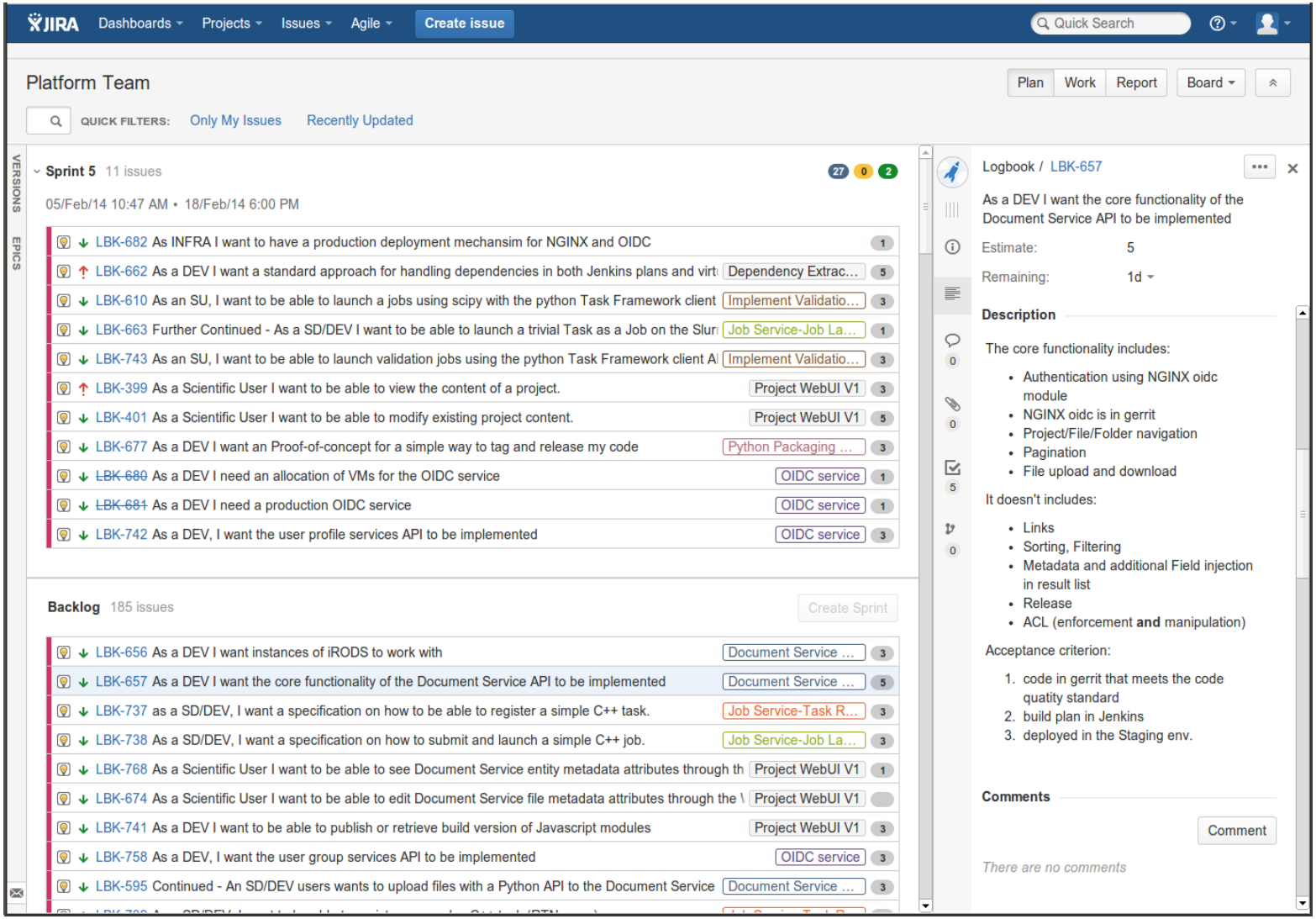

Figure 21: View of the Backlog of the Portal Team

Sprint 5 = current Iteration User Stories, Backlog = rest of Backlog. The order reflects priorities decided by the Product Owner in consultation with the stakeholders.

9.3 Progress Monitoring

9.3.1 Common

The COLL, BSP, Simulators and Initial Brain Models share some common progress monitoring characteristics. Each will have a Backlog. Each Backlog will be separated into three parts: the request Backlog, the release Backlog and the sprint Backlog. The sprint Backlog is the list of Backlog items that will be worked on in the current sprint. The items for the sprint are selected from the release Backlog. The release Backlog is made up of selected Epics that will group related User Stories under a larger theme. There will be release Backlogs for each planned release.

This approach makes it possible to use burndown charts to visually represent the development progress (see Fig. 21). 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 Backlog items as well as the ratio of implemented and not yet implemented Backlog items.
- 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 using what has been done and what has been added to the requirements.

One burndown chart will be provided for each release. They will be updated at the end of each Iteration and will be made available to the stakeholders. The burndown chart will be generated by an automated process with each Iteration. The y-axis is in the Tracking Unit of the respective team. The x-axis is time, in Iterations.

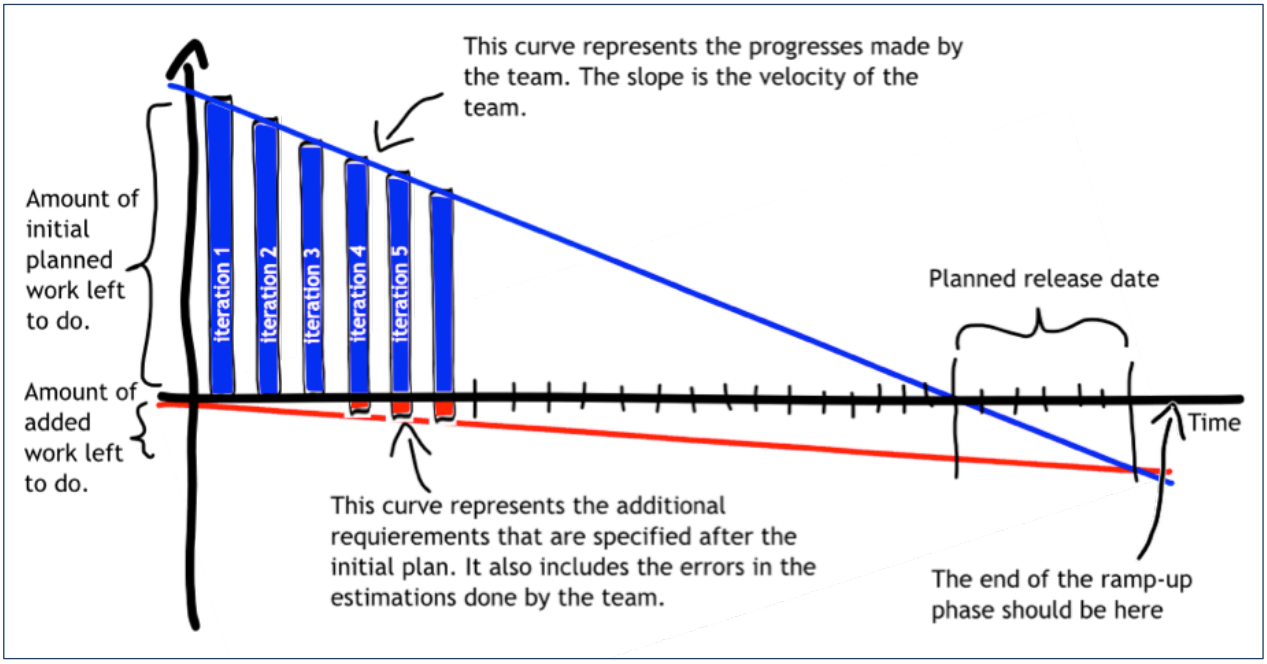

Figure 22: Burndown Chart Example

The goal of this metric is to always have the release date matching the next target release date.

9.3.2 COLL, BSP, CellSim and MolSim Specifics

Backlog items	Stories are centred on a User & what they would like to do. Bugs are software issues that need to be fixed. Tasks are all other activities.
Iteration Length	2 weeks
Tracking unit	Points - an abstract measure of Backlog item size and complexity, 1 point corresponds to 1 estimated person-day, 3 points corresponds to 1 estimated person-week, 5 points is 2+ person-weeks (too large and must be broken up). Correspondence between points and time-estimates is affected by many factors in the team's environment. It is expected to be team specific but relatively constant from Iteration to Iteration.
Release Backlog Preparation	Immediately following the completion of the previous release.

9.3.3 NetSim Specifics

Backlog items	Tickets – a ticket in a ticket tracking system, no estimated person-time cost.
Iteration Length	Reporting Frequency; should be 2 weeks.
Tracking unit	Tickets
Release Backlog Preparation	Month 6

9.3.4 Initial Brain Model Specifics

Backlog items	Tasks - elements in the Initial Brain Models release Backlog.
Iteration Length	Reporting Frequency; should be 2 weeks.
Tracking unit	Points - an abstract measure of Backlog item size and complexity. In the case of the Initial Brain Models, 1 point corresponds to one week. Generally a Step should be less than 2 points or it should be broken up into smaller steps. Otherwise the Step is probably not well understood and the estimate should be considered suspect.
Release Backlog Preparation	Immediately following the completion of the previous release.

9.4 KPI Sheets

Key Performance indicator (KPI) sheets are included in this report as an Excel attachment. The KPI Sheets in their current form are meant to indicate the information that will be included, but the layout and style of the KPI Sheets may change.

Human Brain Project - Key Performance Indicators

KPIs

Software Development

Release Name	Target Completion Date (month)	Estimated Completion Date (month)	Point Completion	Point Volatility	Iteration Length (weeks)	Team Velocity (points/itera	Velocity Volatility
Software							
Unifying Portal - MVP	12	11	60%	6%	2	20.25	13%
Brain Simulation Platform - Somatosensory workflows V1	12	14	47%	10%	2	24	15%
Molecular Simulator - multi-processor	30	30	13%	10%	2	9	20%
Cellular Simulator - 10 million cells on CSCS	24	24	27%	10%	2	11	17%
Network Simulator	30	30	17%	5%	2	2	33%
Modelling							
Phase Name							
Meso							
Somatosensory Cortex - Phase 1	10	11	51%	6%	2	6	20%
Somatosensory Cortex - Phase 2 - MS119 - not started	12						
Cellular level modelling - Phase 1	12	11	76%	25%	2	2.5	33%
Cellular level modelling - Phase 2 - Platform integration MS120	18						
Hippocampus CA1 - Phase 1 - Not started							
Hippocampus CA1 - Phase 2 - MS120 - Not started	24						
Cerebullum - MS122	28						
Full Hippocampus - MS122	28						
Neocortex - MS122	28						
Macro							
Cerebullum, Full Hippocampus and Neocortex - MS122	30						

Table 14: Software Development and Modelling KPIs

Human Brain Project - Key Performance Indicators		
KPIs		
Usage	**Target Value - M12**	**Actual Value - M12**
Activity		
# Active Users	8	15
# Projects Lanched	4	10
# Projects Released	1	3
Data		
# Searches	300	1378
Data links added - External (count)	50	87
Data uploaded (MB)	2000	1500
Data inputs consumed (count)	150	2765
Tools		
Total Jobs Launched	300	930
Analysis Jobs Launched	200	573
Build Jobs Launched	20	13
Simulation Jobs Launched	80	231
Tools Registered (new or updated)	50	65
Validations Registered	20	25

Table 15: Usage KPIs

10. Functions

The COLL and BSP are being developed using the SCRUM Agile methodology. The Functions are provided, not to measure progress, but to allow coarse-grained prioritisation in the roadmap and to synchronise the various HBP Platform development efforts.

10.1 Collaboratory (HBP-COLL)

Task No:	6.5.1	Partner:	EPFL
Function No:	6.5.1.1	Leader:	Jeff Muller
Function Name:	Low Volume Scientific Data Sharing, single COLL Project with upload		
Test Use Case:	SP6COLL-UC-001		
Planned Start Date:	October 2013	Planned Completion Date:	May 2014
Requires Functions:			

Task No:	6.5.1	Partner:	EPFL
Function No:	6.5.1.2	Leader:	Jeff Muller
Function Name:	Low Volume Scientific Data Sharing, multi COLL Project		
Test Use Case:	SP6COLL-UC-002		
Planned Start Date:	October 2013	Planned Completion Date:	May 2014
Requires Functions:			

Task No:	6.5.1	Partner:	EPFL
Function No:	6.5.1.3	Leader:	Jeff Muller
Function Name:	Viewing of Data in HBP-COLL – Basic Integrated Viewers		
Test Use Case:	SP6COLL-UC-003, no search, viewers are available from the main COLL Project view. Viewer list: images, movies, morphologies, provenance, validation reports		
Planned Start Date:	October 2013	Planned Completion Date:	July 2014
Requires Functions:			

Task No:	6.5.1	Partner:	EPFL
Function No:	6.5.1.4	Leader:	Jeff Muller
Function Name:	Viewing of Data in HBP-COLL – Basic Search Integration		
Test Use Case:	SP6COLL-UC-003, search returns list view with embedded mimetype viewers. Viewer list: images, movies, morphologies, provenance, validation reports		
Planned Start Date:	October 2013	Planned Completion Date:	September 2014
Requires Functions:			

Task No:	6.5.1	Partner:	EPFL
Function No:	6.5.1.5	Leader:	Jeff Muller
Function Name:	Data hiding		
Test Use Case:	SP6COLL-UC-004		
Planned Start Date:	October 2013	Planned Completion Date:	July 2014
Requires Functions:			

Task No:	6.5.1	Partner:	EPFL
Function No:	6.5.1.6	Leader:	Jeff Muller
Function Name:	Data Release		
Test Use Case:	SP6COLL-UC-005		
Planned Start Date:	October 2013	Planned Completion Date:	July 2014
Requires Functions:			

Task No:	6.5.1	Partner:	EPFL
Function No:	6.5.1.7	Leader:	Jeff Muller
Function Name:	Collaborative Scientific Analysis		
Test Use Case:	SP6COLL-UC-006, initial analysis available in the COLL		
Planned Start Date:	October 2013	Planned Completion Date:	July 2014
Requires Functions:			

Task No:	6.5.1	Partner:	EPFL
Function No:	6.5.1.8	Leader:	Jeff Muller
Function Name:	Portal Developer services and component reuse		
Test Use Case:	SP6COLL-UC-007 initial support for Portal extension by partners for integration of client-side components as in 1.2.7.3		
Planned Start Date:	October 2013	Planned Completion Date:	November 2015
Requires Functions:			

Task No:	6.5.1	Partner:	EPFL
Function No:	6.5.1.9	Leader:	Jeff Muller
Function Name:	Portal Developer services and component reuse		
Test Use Case:	SP6COLL-UC-007 Second round support for Portal extension by partners for registration of server-side components as in 1.2.7.4		
Planned Start Date:	Depends on demand	Planned Completion Date:	Depends on demand
Requires Functions:			

Task No:	6.5.1	Partner:	EPFL
Function No:	6.5.1.10	Leader:	Jeff Muller
Function Name:	Scientific Developer iterative workflow development		
Test Use Case:	SP6COLL-UC-008 initial Task SDK released to selected Users. The Task SDK functionality will be continually improved following the initial release to improve rapid workflow improvement Iteration by Users. The Task SDK will be released to an ever-widening audience of HBP Users throughout the Ramp-Up Phase.		
Planned Start Date:	October 2013	Planned Completion Date:	July 2014
Requires Functions:			

Task No:	6.5.1	Partner:	EPFL
Function No:	6.5.1.11	Leader:	Jeff Muller
Function Name:	Visualisation Developer component reuse		
Test Use Case:	SP6COLL-UC-009 – This Use Case will be enabled by services provided in 6.5.1.4, but will not be implemented by the Platform team. It will be left to the HPC Platform teams to decide when this functionality should be exploited to satisfy SP6COLL-UC-009.		
Planned Start Date:	October 2013	Planned Completion Date:	Uncertain
Requires Functions:	6.5.1.3, 6.5.1.4		

Task No:	6.2.1	Partner:	EPFL
Function No:	6.2.1.1	Leader:	Jeff Muller
Function Name:	Interactive Atlas Exploration		
Test Use Case:	SP6COLL-UC-010		
Planned Start Date:	December 2013	Planned Completion Date:	January 2015
Requires Functions:	6.5.1.3, 6.5.1.8		

Task No:	6.2.1	Partner:	EPFL
Function No:	6.2.1.2	Leader:	Jeff Muller
Function Name:	The BAEM as a User interface component for cell subset selection		
Test Use Case:	SP6COLL-UC-011		
Planned Start Date:	October 2013	Planned Completion Date:	November 2014
Requires Functions:	6.5.1.8		

Task No:	6.2.1	Partner:	EPFL
Function No:	6.2.1.3	Leader:	Jeff Muller
Function Name:	Interactive Exploration of the Circuit Model		
Test Use Case:	SP6COLL-UC-012		
Planned Start Date:	October 2013	Planned Completion Date:	November 2014
Requires Functions:	6.5.1.8		

Task No:	6.2.1	Partner:	EPFL
Function No:	6.2.1.4	Leader:	Jeff Muller
Function Name:	Finding Data through direct search		
Test Use Case:	SP6COLL-UC-013		
Planned Start Date:	October 2013	Planned Completion Date:	November 2014
Requires Functions:	6.5.1.3		

Task No:	6.2.1	Partner:	EPFL
Function No:	6.2.1.5	Leader:	Jeff Muller
Function Name:	Finding Data during Task configuration		
Test Use Case:	SP6COLL-UC-014		
Planned Start Date:	October 2013	Planned Completion Date:	January 2015
Requires Functions:	6.5.1.3		

10.2 Brain Simulation Platform (BSP)

The Functions below will be used to describe the initial delivery of such functionality in the COLL, and the BSP will be updated periodically throughout the Ramp-Up Phase in conjunction with improvements in the Initial Brain Models.

Task No:	6.1.1	Partner:	EPFL
Function No:	6.1.1.1	Leader:	Jeff Muller
Function Name:	Brain Builder		
Test Use Case:	SP6BSP-UC-001 – Builders will be added and updated constantly over the course development of the COLL and BSP. Initial Builder availability date is listed below.		
Planned Start Date:	July 2014	Initial Availability Date:	November 2015
Requires Functions:	6.5.1.3, 6.5.1.8, 6.5.1.10, 7.5.2.1, 7.5.7.5		

Task No:	6.1.1	Partner:	EPFL
Function No:	6.1.1.2	Leader:	Jeff Muller
Function Name:	Validations		
Test Use Case:	SP6BSP-UC-002 – Validations will be added and updated constantly over the course development of the COLL and BSP. Initial Validations availability date is listed below.		
Planned Start Date:	March 2014	Initial Availability Date:	July 2014
Requires Functions:	6.5.1.3, 6.5.1.10		

Task No:	6.1.1	Partner:	EPFL
Function No:	6.1.1.3	Leader:	Jeff Muller
Function Name:	Compound Model and Model Component Analysis		
Test Use Case:	SP6BSP-UC-003 – Analyses will be added and updated constantly over the course development of the COLL and BSP. Initial Analysis Task availability date is listed below.		
Planned Start Date:	March 2014	Initial Availability Date:	July 2014
Requires Functions:	6.5.1.3, 6.5.1.10		

Task No:	6.1.5	Partner:	EPFL
Function No:	6.1.5.1	Leader:	Jeff Muller
Function Name:	Simulation Configure and Launch		
Test Use Case:	SP6BSP-UC-004 without the selection of computing resource and HPC Platform resource allocation recommendations (SP6-UC-005-4).		
Planned Start Date:	March 2014	Initial Availability Date:	November 2014
Requires Functions:	6.5.1.3, 6.5.1.8, 6.5.1.10, 7.5.2.1, 7.5.7.5		

Task No:	6.1.1	Partner:	EPFL
Function No:	6.1.1.4	Leader:	Jeff Muller
Function Name:	Simulation Analysis Tools		
Test Use Case:	SP6BSP-UC-005		
Planned Start Date:	March 2014	Initial Builder Availability Date:	July 2014
Requires Functions:	6.5.1.3, 6.5.1.7, 7.5.2.1, 7.5.7.5		

Task No:	6.1.1	Partner:	EPFL
Function No:	6.1.1.5	Leader:	Jeff Muller
Function Name:	Collaborative Review Process		
Test Use Case:	SP6BSP-UC-006		
Planned Start Date:	March 2014	Initial Builder Availability Date:	July 2014
Requires Functions:	6.5.1.3, 6.5.1.7		

10.3 BSP: Brain Builder

Task No:	6.1.1	Partner:	EPFL
Function No:	6.1.1.6	Leader:	Jeff Muller
Function Name:	Repair and diversification of reconstructed morphologies		
Test Use Case:	SP6BSP-UC-007		
Planned Start Date:	March 2014	Initial Availability Date:	August 2014
Requires Functions:	6.5.1.3, 6.5.1.10		

Task No:	6.1.1	Partner:	EPFL
Function No:	6.1.1.7	Leader:	Jeff Muller
Function Name:	Synthesise full cell morphologies		
Test Use Case:	SP6BSP-UC-008		
Planned Start Date:	March 2014	Initial Builder Availability Date:	November 2014
Requires Functions:	6.5.1.3, 6.5.1.10		

Task No:	6.1.3	Partner:	EPFL
Function No:	6.1.3.1	Leader:	Jeff Muller
Function Name:	Create a complete Cell model using automated fitting of conductance densities		
Test Use Case:	SP6BSP-UC-009		
Planned Start Date:	March 2014	Initial Builder Availability Date:	November 2014
Requires Functions:	6.5.1.3, 6.5.1.10		

Task No:	6.1.1	Partner:	EPFL
Function No:	6.1.1.8	Leader:	Jeff Muller
Function Name:	Distribute cells and use this to create a point neuron model of a brain region		
Test Use Case:	SP6BSP-UC-010		
Planned Start Date:	March 2014	Initial Builder Availability Date:	January 2015
Requires Functions:	6.5.1.10		

Task No:	6.1.1	Partner:	EPFL
Function No:	6.1.1.9	Leader:	Jeff Muller
Function Name:	Distribute cells and use this to create a point neuron model of a whole mouse brain		
Test Use Case:	SP6BSP-UC-011		
Planned Start Date:	March 2014	Initial Builder Availability Date:	February 2014
Requires Functions:	6.5.1.10		

Task No:	6.1.1	Partner:	EPFL
Function No:	6.1.1.10	Leader:	Jeff Muller
Function Name:	Distribute cells and use this to create a detailed neuron model of a mouse neuronal microcircuit		
Test Use Case:	SP6BSP-UC-012		
Planned Start Date:	March 2014	Initial Builder Availability Date:	October 2014
Requires Functions:	6.5.1.10		

Task No:	6.1.5	Partner:	EPFL
Function No:	6.1.5.13	Leader:	Jeff Muller
Function Name:	Simplify the Cellular level model to a Network level model		
Test Use Case:	SP6BSP-UC-013		
Planned Start Date:	March 2014	Initial Builder Availability Date:	October 2014
Requires Functions:	6.5.1.10		

Task No:	6.1.5	Partner:	EPFL
Function No:	6.1.5.14	Leader:	Jeff Muller
Function Name:	Export a volume region of the Cellular level model and add molecular level detail to produce a Molecular level model		
Test Use Case:	SP6BSP-UC-013		
Planned Start Date:	March 2014	Initial Builder Availability Date:	October 2014
Requires Functions:	6.5.1.10		

10.4 BSP: Molecular Simulator

Task No:	6.2.1	Partner:	EPFL
Function No:	6.2.1.1	Leader:	Fabien Delalondre
Function Name:	Geometrically accurate synapse model with molecular reactions and diffusion		
Test Use Case:	SP6BSP-UC-015 without multi-objective feature optimisation (SPBSP-UC-015.1)		
Planned Start Date:	March 2014	Initial Simulator Platform Availability Date:	April 2015
Requires Functions:	6.5.1.7		

Task No:	6.2.1	Partner:	EPFL
Function No:	6.2.1.2	Leader:	Jeff Muller and Fabien Delalondre
Function Name:	Molecular Neuron simulation using MolSim		
Test Use Case:	SP6BSP-UC-016 without interactive cockpit analysis.		
Planned Start Date:	March 2014	Initial Simulator Platform Availability Date:	April 2015
Requires Functions:	6.5.1.7 for SPBSP-UC-016.1 to SPBSP-UC-016.4, 7.3.4.1, 7.3.4.2, 7.3.4.4 for SPBSP-UC-016.5		

Task No:	6.2.1	Partner:	EPFL
Function No:	6.2.1.3	Leader:	Fabien Delalondre
Function Name:	MS115 – Parallel Molecular Simulator		
Test Use Case:	STEPS simulator can take advantage of multiple threads in multi-core machines to decrease wall-time of larger STEPS simulations.		
Planned Start Date:	March 2014	Initial Simulator Platform Availability Date:	April 2016
Requires Functions:	6.5.1.7		

10.5 BSP: Cellular Simulator

Task No:	6.2.2	Partner:	EPFL
Function No:	6.2.2.1	Leader:	Jeff Muller
Function Name:	Simulation of a microcircuit with biophysically realistic neurons		
Test Use Case:	SP6BSP-UC-017		
Planned Start Date:	March 2014	Initial Simulator Platform Availability Date:	October 2014
Requires Functions:	6.1.5.1, 6.1.1.4		

Task No:	6.2.2	Partner:	EPFL
Function No:	6.2.2.2	Leader:	Fabien Delalondre
Function Name:	Multi-parameter exploration of medium sized networks with biophysically detailed neurons		
Test Use Case:	SP6BSP-UC-018		
Planned Start Date:	March 2014	Initial Simulator Platform Availability Date:	October 2014
Requires Functions:	6.1.5.1, 6.1.1.4		

Task No:	6.2.2	Partner:	EPFL
Function No:	6.2.2.3	Leader:	Fabien Delalondre
Function Name:	Full-scale simulation of an entire brain region with biophysically realistic neurons		
Test Use Case:	SP6BSP-UC-019		
Planned Start Date:	March 2014	Initial Simulator Platform Availability Date:	October 2014
Requires Functions:	6.1.5.1, 6.1.1.4 Requires a version of Neuron that can scale to a full brain region on current supercomputers. Also requires the interactive supercomputing stack be able to provide streaming LFP.		

10.6 BSP: Network Simulator

Functions are not provided for the Network Simulator, as the progress tracking approach is different.

Annex A: Glossary

Term	Description
0-9	
2D Atlas	A 2D reference space, a collection of 2D parcellations, 2D images or a collection of registered data.
2D Parcellation	A collection of closed polygonal or spline boundaries at specific cut planes in 3D space. Each boundary is linked to one or more ontological elements.
2D Reference space	A collection of 2D subspaces each aligned with a specific cut plane in 3D space. Each subspace has a single coordinate origin and affine transformation.
3D Atlas	A 3D reference space, a collection of 3D parcellations, 3D voxel volumes and a collection of registered data.
3D Parcellation	A collection of meshes that define 3D boundaries. Each boundary is linked to one or more ontological elements.
3D Reference space	A set of 3D basis vectors with a single coordinate origin and affine transformation.
A	
Anchor	A spatial location with orientation and scale or a semantic-spatial with optional orientation and scale.
Artefact	A high data-density discrete data element, primarily meant to denote a file larger than 10kB which is not human readable or editable.
Atlas	A 2D or 3D atlas
B	
Biophysically realistic	Mathematical description of physical phenomena relevant to the biological processes of cellular behaviour. In particular, but not limited to, a model of the neuronal tree with the cable equation and ion channels by the phenomenological Hodgkin-Huxley formulation.
Brain System	A specific set of interacting brain regions.
C	
Capability supercomputing	Tightly integrated parallel supercomputer providing a high-speed and low-latency network between the computing nodes (in contrast to embarrassingly parallel computers).
Cockpit	Desktop, display wall or cave visualisation resource with a mechanism for good data locality.
Cognitive Architecture	A specific set of brain regions and interactions that are proposed to underlie specific cognitive capabilities.
Compute resource	A computer or collection of computers where a job can be executed
Configurability	The ability of the simulator to provide extensibility for additional mathematical formulations of novel physical phenomena and integrating this mathematics in the compute-critical inner integration loop.

Term	Description
Continuous time spike interaction	Representation of the time point of an action potential as a floating-point number with double precision on a continuous time axis. The action potential is generated at the time point of threshold crossing of the sending neuron and is communicated with full precision to its target cells, where it affects the receiving neuron, acting at the point in time after the application synapse delay.
CRUD	A commonly used acronym for Create, Read, Update, Delete.
Curation	A manual or analytic process involving a human to make decisions about some property of the reconstruction or one of its components. Curation can be applied to everything from electrical channel models in the Hodgkin-Huxley model to ontology names for a particular neuron morphology class.
D	
Datatype	A datatype is a semantically enriched mimetype. For example, the mimetype of a particular data file might be XML, but the datatype would be CircuitML, implying that the data file can be interpreted in a richer way. This allows the selection of editing interfaces and input data much more User friendly in the Collaboratory.
Detailed model	The finest level of representation with full geometry
E	
e-type	A short-hand form of the ne-type abbreviation defined below
Entity	A COLL Project, a file or a folder. Theses exist as part of a hierarchy; they may have a parent and children. Each entity has a series of predefined key values associated to it (like name, creation date...) and can also have some custom Metadata associated to it.
Exact integration	A method applicable to the integration of sets of linear differential equations. The solution agrees to the mathematically exact solution. Often formulated in terms of a matrix exponential.
F	
File	Entities that are required to have a parent but cannot have any children. In addition to the standard attribute they also have a Content URL that defines how to access the content of the file.
Folder	Plain Entities that are required to have a parent.
Full scale	Representation of a network with the natural number of neurons and synapses per neuron as found in the biological system.
G	
Glial cell	Non-neuronal cells that function in homeostasis and energy usage, which provide support and protection for neurons. They can be divided into microglia and macroglia types.
H	
HBP Collaboratory (HBP-COLL)	The unified web interface through which the web-accessible components of the six HBP Platforms and all other HBP activities are made available.
HBPMIN	A minimum metadata specification. Similar in spirit to the Carmen MINI specification but tailored to the Use Cases of the HBP.

Term	Description
Hidden Entity	Entities can be hidden from the Document Service REST API and through the Web GUI. Hidden data are not visible by default in the COLL Project browser. Hidden data will still be optionally visible (thought marked as hidden) to anyone it has been shared with. Hidden data are a separate function from true deletion. Deletion of data is a highly privileged operation that must be done by System Administrators on User request. See the Data Hiding Use Case for more information.
Host	A single operating system instance, running on virtualised or real hardware
I	
Ionic conductance models	Models that represent ionic permeation through the plasma membrane. Both stochastic and deterministic approaches should be covered. Extension toward molecular models (WP6.4.1) is envisaged.
J	
Job	An instance of an execution of a Task on a compute resource. For some Tasks, the compute resource will be selected by the User in the COLL on job launch. For other Tasks the execution will be decided by the Task.
L	
Level of resolution	The choice of abstraction applied to the representation of the network. The level of resolution of MolSim corresponds to single neurons or synapses. The level of resolution of NetSim corresponds to single neurons and synapses. The level of resolution of CellSims equates to electrical compartments coupled by conductances.
M	
m-type	A short-hand form of the nm-type abbreviation below
me-type	A short-hand form of the nme-type abbreviation below
Macrocircuit	The definition of the whole brain as a set of brain regions connected through long-range fibre tracts - the whole brain.
Mesocircuit	The definition of the smallest collection of midrange interacting microcircuits through their intra-areal or regional arbours - a brain area or region.
Metabolism	The set of chemical transformations within the cells of living organisms that maintain life.
Microcircuit	The definition of the smallest collection of short-range interacting neurons through their local arbours.
Microcircuit models	Models that represent an entire microcircuit, including 3D geometrical architecture, synaptic connectivity and neuronal and synaptic models.
Molecular level models	Models that are structurally accurate at the subcellular level (organelles, intracellular and extracellular spaces) and that contain molecules that ultimately follow cell biological rules of production, transport, localisation and degradation as well as the environment-dependent thermodynamics and kinetics of their interactions. Both stochastic as well as deterministic versions will be covered.

Term	Description
Molecular Simulations	Numerical simulations at the atomistic or coarse-grained level used for predicting structures of molecular complexes and the estimation of kinetic and thermodynamic parameters for molecular interactions. Molecular Simulations are based on atomistic structures of proteins available either from the Protein Data Bank or from homology modelling.
Morphology	The geometric definition of the shape of a neuron.
Multi-constraint fitting	The process whereby one data parameter or property constrains other data parameters or properties.
N	
n-type	A class of brain cells or a particular instance, depending on context.
ne-type	Abbreviation for an electrophysiological type of cell. This abbreviation is used to refer to a class of cells or a particular instance, depending on context.
ng-type	Abbreviation for a genetic type of cell. This abbreviation is used to refer to a class of cells or a particular instance, depending on context.
nm-type	Abbreviation for a morphology type of cell. This abbreviation is used to refer to a class of cells or a particular instance, depending on context.
nme-type	Abbreviation for a morpho-electrophysiological combination type of cell. This abbreviation is used to refer to a class of cells or a particular instance, depending on context.
np-type	Abbreviation for a protein type of cell. This abbreviation is used to refer to a class of cells or a particular instance, depending on context.
Neuro-glia vasculature (NGV)	The three principal components in neural tissue, which function as a unit to regulate blood flow and metabolism.
NEURON	Open source simulator NEURON (http://www.neuron.yale.edu) developed by Michael L. Hines.
Neuron model	Implementation of neuron dynamics defined as a set of differential equations. The implementation solves the dynamics within a finite time span given the incoming spike events are supplied. Incoming synapses can be modelled as currents or conductances.
Neuropil	Any area in the nervous system composed of mostly unmyelinated axons, dendrites and glial cell processes that form a synaptically dense region containing a relatively low number of cell bodies.
P	
p-type	An abbreviation for projection type, a to-be-determined classification scheme for determining classes of projections between meso-scale brain regions.
Parameter	A low data-density discrete data element that is primarily meant to denote a value that one might enter into a single form element. It might also be used to refer to a richer configuration document containing a group of settings.
Parcellation	One or more spatial boundaries associated with a set of discrete semantic concepts. Usually developed by manual, semi-automated or automated image analysis of landmarks.
Platform	Software components: libraries, services, APIs and their documentation that are to be used to build portals or cockpits.
Predictive reconstruction	The process whereby multi-constraint solutions yield a hypothesis and hence a prediction of the data parameter space.

Term	Description
COLL Project	COLL Projects are Entities with no parent. In addition with the standard attribute and Metadata associated with Entities, COLL Projects have also ACL that define which Users can access their content.
R	
Reconstruction data	Data that is used to parameterise a model.
Reconstruction process	A workflow that uses a configuration of the data parameters and implements a set of fundamental biological principles to constrain and instantiate the model.
Reference space	In 2D, a collection of slices with an optional 2D *parcellation*. In 3D, a voxel volume with an optional 3D *parcellation*.
Registered data	A URL accessible data set with an *anchor*.
Resources	Parameters, Artefacts, services, or compute capacity
REST	An acronym for REpresentational State Transfer, for a definition see http://en.wikipedia.org/wiki/Representational_state_transfer
S	
s-type	Abbreviation for the type of synaptic connection. This abbreviation is used to refer to a class of cells or a particular instance depending on context. Abbreviations for specific dimensions of a synapse include;
sa-type	Abbreviation for the anatomical type of synaptic connection. This abbreviation is used to refer to a class of synaptic connection or a particular instance depending on context.
sp-type	Abbreviation for the physiological type of synaptic connection. This abbreviation is used to refer to a class of synaptic connection or a particular instance depending on context.
SAN	An acronym for Storage Area Network, http://en.wikipedia.org/wiki/Storage_area_network
Semantic-spatial location	Association of semantic concept (e.g.: cerebellum) with a spatial boundary.
Service	A software function performed by a third party for a User or other Service. In the language of the COLL, Services consume Parameters, Artefacts and compute capacity. Services produce Artefacts and parameters.
Single neuron models	Models that represent entire neurons, including 3D structure, electroresponsiveness, synaptic activation and intracellular biochemical cascades (developed in WP6.4.1).
Site	A collection of hosts collected together in a single location. The grouping is potentially arbitrary. QIJ might be considered one site, LNMC another or one might consider EPFL a site unto itself.
Spatial location	2D or 3D location
Synapse model	A model representing synaptic plasticity, such as spike timing dependent plasticity (STDP). The implementation solves the dynamic equation describing the evolution of the synaptic amplitude, typically formulated as a differential equation, given the spike times of the presynaptic and possibly the postsynaptic neuron are given.

Term	Description
Synaptic models	Models that represent processes of synaptic transmission, including neurotransmitter release and postsynaptic receptor activation. Both stochastic and deterministic approaches should be covered. Extension toward molecular models and molecular networks (WP6.4.1) of neuromodulation, synaptic plasticity and homeostasis is envisaged.
Systems Biology Markup Language (SBML)	A mark-up language for representing standardised reaction networks within compartments.
T	
Task	A logical software unit. A Task takes Artefacts and Parameters as input, and produces Artefacts and Parameters as output. It may or may not be visible as a Service. A Task identifies its dependencies and its default parameters. Concretely, it is a software component that combines: • A Python-based Task entry point • A git repository or Python package index URL for the Task • A repository revision or package content specified by sha1 • A requirements file specifying all required dependencies. Tasks can have dependencies in non-Python languages, but these dependencies must be packaged for reproducible deployment.
Task definition	The collection of data that defines an individual Task
Task repository	A database of Task definitions
V	
Validation data	Data that is used to validate a model.
Validation process	A workflow that compares results obtained in the model when experimental protocols used to obtain the validation data are applied to the model..
Vasoconstriction	Narrowing of blood vessels resulting from constricting of smooth muscle cells within the vessel walls
Vasodilation	Widening of blood vessels due to relaxation of smooth muscle cells within the vessel walls
Voxel	A 3D unit volume, the 3D analogue of an image pixel.
Voxel volume	A 3D volume made up of voxels. Typically, the voxels densely fill a rectangular prism spatial bounding volume.
W	
Workflow	A tree of decision structures and Tasks. A Workflow takes Artefacts and Parameters as input, and produces Artefacts and Parameters as output. It may or may not be visible as a Service.

Annex B: References

[1] Tauheed S. Scalable Exploration of Spatial Data in Large-Scale Scientific Simulations [PhD thesis]. École Polytechnique Fédérale de Lausanne, Lausanne; 2014. Available from: http://infoscience.epfl.ch/record/199427/files/EPFL_TH6125.pdf. vii.

[2] Wils S, De Schutter E. "STEPS: Modeling and Simulating Complex Reaction-Diffusion Systems with Python." Front Neuroinform. 2009 Jun 29;3:15.

[3] Hepburn I, Chen W, Wils S, De Schutter E. "STEPS: efficient simulation of stochastic reaction-diffusion models in realistic morphologies." BMC Syst Biol. 2012 May 10;6:36

[4] Wils S, De Schutter E. "STEPS: Modeling and Simulating Complex Reaction-Diffusion Systems with Python." Front Neuroinform. 2009 Jun 29;3:15.

[5] Boman E. G., CatalyurekU. V. ,Chevalier C., DevineK. D., The Zoltan and Isorropia Parallel Toolkits for Combinatorial Scientific Computing: Partitioning, Ordering, and Colorin, 20-2, 2012

[6] Parmetis website, http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview

[7] Carnevale, N.T., Hines, M.L.: The NEURON book. Cambridge University Press, Cambridge; New York (2005)

[8] J.G. King, M. Hines, S. Hill, P.H. Goodman, H. Markram, F. Schuermann, A Component-Based Extension Framework for Large-Scale Parallel Simulations in NEURON, Frontiers in Neuroinformatics, 3:10, 2009

[9] M.L. Hines, H. Markram, F. Schuermann, Fully Implicit Parallel Simulation of Single Neurons, Journal of Computational Neuroscience, 25(3): 439-448, 2008

[10] M. Hines, S. Kumar, F. Schuermann, Comparison of Neuronal Spike Exchange Methods on a Blue Gene/P Supercomputer, Frontiers in Computational Neuroscience, 5:40, 2011

[10] D'Angelo E., Nieus T., Maffei A., Armano S., Rossi P., Taglietti V., Fontana A., Naldi G. (2001) Theta-frequency bursting and resonance in cerebellar granule cells: experimental evidence and modeling of a slow K+-dependent mechanism. J. Neurosci. 21, pp. 759-770.

[11] Diwakar S, Lombardo P, Solinas S, Naldi G, D'Angelo E. (2011). Local field potential modeling predicts dense activation in cerebellar granule cells clusters under LTP and LTD control. PLoS One. 2011;6(7).

[12] Thierry Nieus, Elisabetta Sola, Jonathan Mapelli, Elena Saftenku, Paola Rossi, Egidio D'Angelo. (2006) Regulation of repetitive neurotransmission and firing by releaSe probability at the input stage of cerebellum: experimental observations and theoretical predictions on the role of LTP. J Neurophysiol 95, pp. 686-699.

[13] Sergio M. Solinas, Lia Forti, Elisabetta Cesana, Jonathan Mapelli, Erik De Schutter and Egidio D`Angelo (2007) Computational reconstruction of pacemaking and intrinsic electroresponsiveness in cerebellar Golgi cells. Front. Cell. Neurosci. 1:2. doi:10.3389/neuro.03.002.2007

[14] Sergio M. Solinas, Lia Forti, Elisabetta Cesana, Jonathan Mapelli, Erik De Schutter and Egidio D`Angelo. (2007) Fast-reset of pacemaking and theta-frequency resonance patterns in cerebellar Golgi cells: simulations of their impact in vivo. Front. Cell. Neurosci. 1:4. doi:10.3389/neuro.03.004.2007.

[15] Colonnese, M. T., & Khazipov, R. (2010), "Slow activity transients" in infant rat visual cortex: a spreading synchronous oscillation patterned by retinal waves. *The Journal of neuroscience*, *30*(12), 4325-4337,

[16] Meyer, Hanno S., *et al*. (2010), "Cell type–specific thalamic innervation in a column of rat vibrissal cortex." Cerebral cortex, 20(10): 2287-2303.

[17] Felleman DJ, Van Essen DC, Distributed hierarchical processing in the primate cerebral cortex, Cerebral cortex, 1(1):1-47, Jan-Feb 1991

[18] Markram, H *et al*. (2012), The Human Brain Project Preparatory Study

https://www.humanbrainproject.eu/