The Go-Smart Project is co-funded by: European Commission under grant agreement no. 600641

Go-Smart Glossia

A Tool To Separate Web-user, Manufacturer and Researcher Concerns Through Extensible Simulation Containerization

Phil Weir*, Panchatcharam Mariappan*, Roland Ellerweg', Ronan Flanagan* and Marina Kolesnik'

* - NUMA Engineering Services, Ltd. [IE]
' - Fraunhofer Institute for Applied Information Technology [DE]

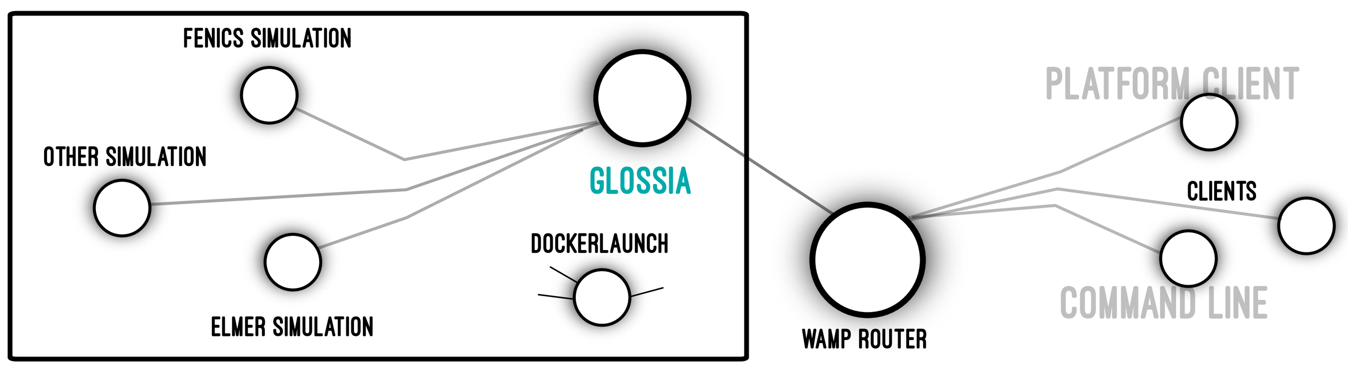
We gratefully acknowledge the contribution of our project technical partners in Aalto University [FI], Technical University of Graz [AU] and University of Oxford [UK]; and our medical partners in Leipzig University [DE], Medical University of Graz [AU], University Hospital Frankfurt [DE], and the Radboud University Medical Centre, Nijmegen [NE]

What is Glossia?

Glossia is an open source tool to manage simulation execution, locally or in the cloud.

Simulations can be run offline or remotely and easily switched between the two, with near-identical environments.

To achieve this, Glossia abstracts external parameters such as case-specific input, to help de-couple code from settings in a way that is reproducible whether on a laptop, on a remote computational machine or on a web platform.



Technical summary: Glossia is managed via a WAMP router (e.g. Crossbar.io) allowing clients to be language independent and run remotely. WAMP is a WebSocket subprotocol. Interaction is via RPC and PubSub patterns. Simulation definition is in an XML format. Region definition is usually in STL or STEP format.

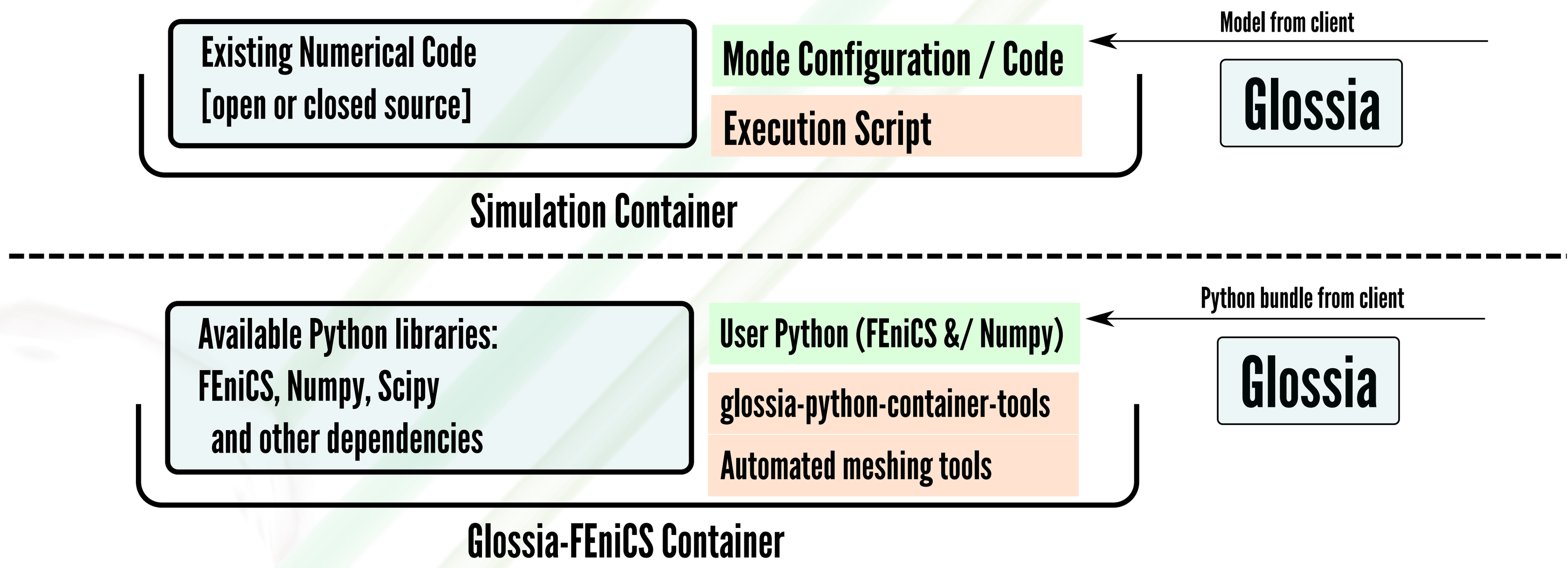
Simulation Families

While Glossia provides support for uploading executable configuration, the actual numerical codes are made available to end users by white-listing numerical tools. These are wrapped in Docker containers, referred to as "families" and containers for new numerical tools may be built quickly from existing templates.

Open source families are available for FEniCS[1] and Elmer[2] - these can be pulled from Dockerhub and are immediately available to Glossia. Glossia AGPL licensing does not affect user code sent to these containers or modifications to these containers, although any license compatibility with internal third-party tools, such as FEniCS, Elmer, CGAL or GMSH should be checked as normal. An industry project has also tested OpenFOAM. FEniCS is recommended, as it provides direct Python bindings, allowing interaction between orchestration and simulation modules. Elmer models are defined as an Elmer SIF, enhanced with Jinja2 templating.

Using an optional template base container, the Glossia tool allows a set of regions to be defined externally (e.g. via a segmentation tool), automatically volumetrically meshed and referenced within the simulation code semantically. At present, in-progress feedback is via status updates, with expansion subject to security considerations.

1 - https://fenicsproject.org; 2 - https://www.csc.fi/web/elmer



A: Illustration of Simulation Families: upper is generic, new containers can be created without using any Glossia tools or libraries. A specific example is shown below, the Glossia-FEniCS container. Any Glossia tools intended for use inside the container are licensed to remain compatible with both open or closed licensing of in-container code and user code. In addition, user STL files (surface meshes) and parameters are injected into the container by Glossia.

Use Cases

SINGLE USER	keeping track of simulations, parameter sets, results. Diff-ing configuration, re-running, viewing progress
SERVICE PROVIDER	running third-party simulations, providing a back-end for non-technical users (e.g. clinicians) to simulate using database-defined models and dynamic parameter sets
INSTITUTION	managing resources, integrating with per-user allocations, internal result sharing, linking members to computational facilities, providing an optimal balance of flexibility and security
TRIAL COORDINATOR	providing a platform to amalgamate and display results, support clinicians, validate models

Running with Glossia

One simulation code may be easily switched between four levels of execution, allowing for easy deployment and diagnostic dissection.

1 Fully independent (development)

Code is written and run as normal, without a container, but with parameters provided by Go-Smart Python modules. These, in turn, read parameters from local YAML files.

2 Containerized (testing/development)

The same code is run in the Docker container that will be used in deployment. The same parameter files are used as in the previous step. Glossia is not run.

3 Local/remote (debug/internal)

Glossia is run on the local machine or local network. A command line utility, glot, facilitates job submission, monitoring and (on local machine) direct retrieval of results or debugging output. Alternatively, Glossia can push results to a file server.

4 Platform-as-a-Service (cloud)

Online service. Parameters may be set through the platform, linked with other components built by users of the platform (e.g. medical equipment profiles). The Python code bundle is uploaded to the server.

WWW.

The system is designed to maximize environment parity between steps 2, 3 and 4. Image IDs of Glossia sandbox containers are added to debug output from Glossia and used to re-run simulations in an identical container locally as remotely. Command line scripts manage execution at each stage, and glot has built in commands to simplify transition.

Application: Minimally Invasive Cancer Treatments

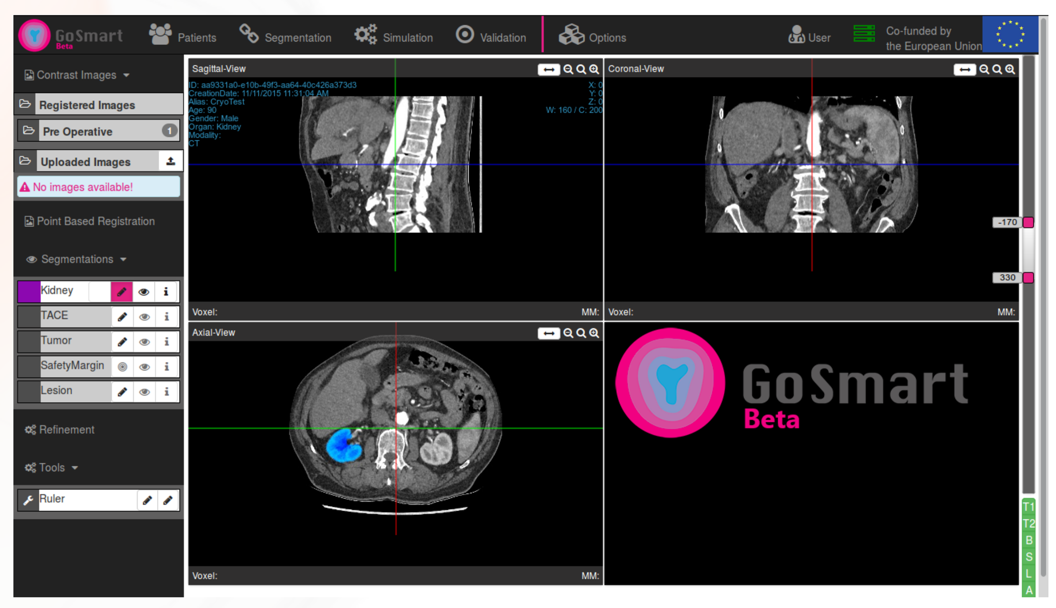
This project has built a platform to help clinicians, researchers and product manufacturers interact online, primarily within the field of Minimally Invasive Cancer Treatments (MICTs).

Clinicians use the platform in, e.g., clinical trials to upload patient data, manage it and perform intervention simulations. They segment data (B), run simulations (C) and may share datasets or perform validation on results.

Researchers may add numerical models to the platform & manufacturers add new equipment (D) or clinical protocols.

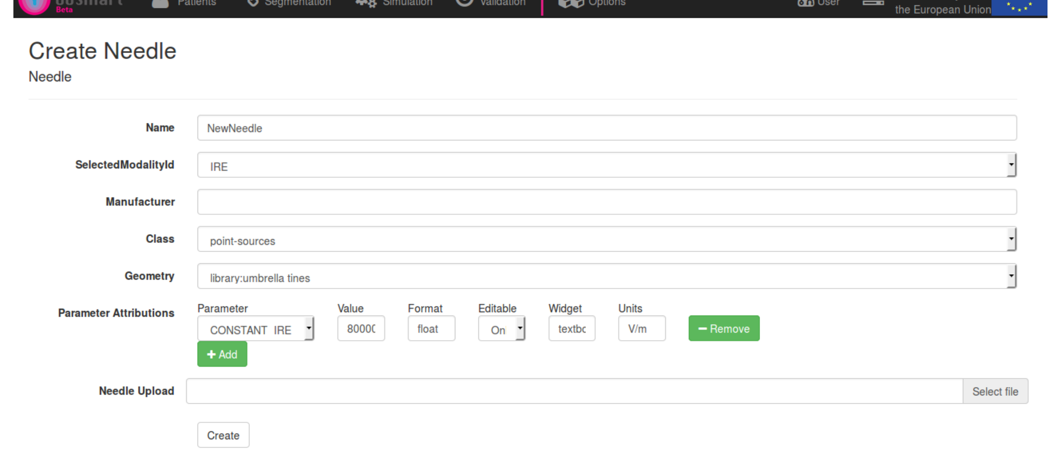
Parameter values may be made adjustable by users at simulation-time, or predefined. Researchers may reuse existing numerical models and equipment to create new simulatable combinations.

This interface is tested by clinicians in several centres across Europe over multiple years and has be designed to be adaptable and model independent. It is distinct from but compatible with Glossia. If you would be interested in finding out more for any clinician-facing or online medical project, research or commercial, please make contact.



B

C



D

The Clinical Domain Model

The Clinical Domain Model (CDM) is not essential to use Glossia, but the Glossia XML format reflects its model-parameter-needle separation.

While Glossia allows model and parameter flexibility, to allow interdependent components to be dynamically built into a single simulatable parameter set, a framework relating them is required. This enables direct/indirect simulation collaboration between clinicians, technical users and manufacturers.

Parameterized Components

MICT Interpretation	Generic Interpretation
Needle	Entity of which a simulation uses 1-N from a predefined collection
Numerical Model	Numerical model
Power Generator	Entity of which a simulation uses 1; may constrain Needle choice
Clinical Protocol	Parameter-replacing series of lambda functions of transient simulation output
Context	Field variables, possibly varying per subdomain.

A simulatable combination is built from components. At simulation time, a combination is chosen and put together with case-specific data. The table to the left shows component types, each supplying named parameters, which may be required from clinicians at simulation-time, or given default values by researchers when defining the component. Components may declare parameter dependencies for each other to supply, restricting the definition of a valid simulatable combination, (commonly used by the Numerical Model).

In the MICT context, for each modality, that is, physically similar set of treatments, many numerical models, clinical protocols and designs of needle and power generator may exist. The modality concept allows grouping.

As the table shows, the MICT context is not fundamental, and the CDM may apply to any setting fitting the Generic Interpretations.

Application: General Simulation Management

This is an administrator/single-user facing prototype system that can read valid simulation combinations from a database, compose simulations using CDM rules then copy, edit, diff, execute and monitor them via Glossia from an up-to-date web browser. Please indicate interest if you would like to follow further developments.

Dockerlaunch

As Docker control requires elevated permissions, a system has been put in place to shield Glossia and the simulation containers from this control. A tool, dockerlaunch, daemonizes and provides a socket to Glossia's own container, that it may use to trigger a highly restricted set of Docker actions, such as starting up to a fixed number of whitelisted Docker containers, or requesting shutdown of containers it has started.

To simplify administrator use, the default Glossia set-up requires no additional configuration, only that Dockerlaunch has already started in the background.

Reference & Links

P Weir et al., "Go-Smart: Web-based computational modeling of minimally invasive cancer treatments", IEEE E-Health and Bioengineering Conference (EHB), 2015, Iași, doi: 10.1109/EHB.2015.7391385

Glossia Documentation: https://go-smart.github.io/glossia
Glot Documentation: https://go-smart.github.io/glot
Dockerlaunch: https://go-smart.github.io/dockerlaunch
Go-Smart Platform: https://smart-mict.eu
This Poster: https://github.com/go-smart/esco-2016-poster
Go-Smart Project: http://gosmart-project.euPoster