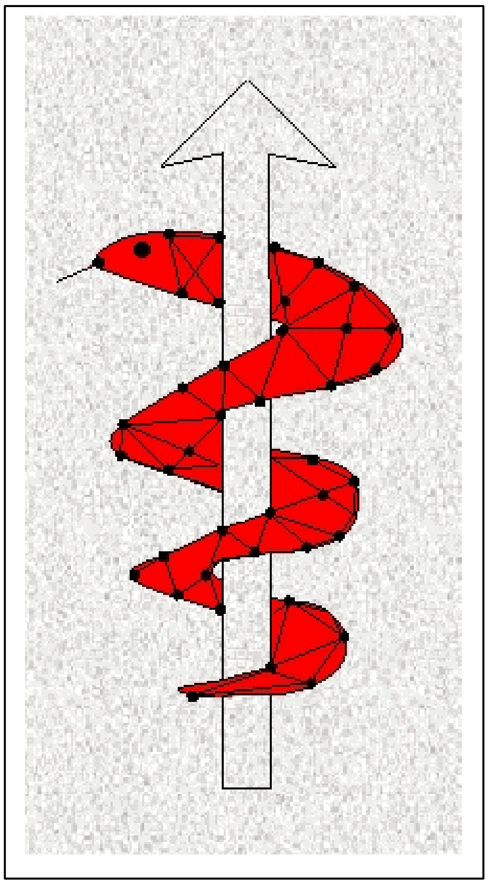
The IST Programme
Project No. 10378

SimBio

SimBio - A Generic Environment for Bio-numerical Simulation

http://www.simbio.de



Deliverable 6c
Release Notes on Final Component Interaction (CI) Release

Status:	Draft
Version:	2.0
Security:	Restricted
Responsible:	ESI
Authoring Partners:	ESI and NEC

Release History

Version	Date
1.0	07.05.2002
2.0	22.05.2002

The SimBio Consortium:
NEC Europe Ltd. – UK
A.N.T. Software – The Netherlands
CNRS-DR18 – France
Sheffield University – UK
MPI of Cognitive Neuroscience – Germany
Biomagnetisches Zentrum Jena – Germany
ESI Group – France
Smith & Nephew - UK
U. Maribor - Slovenia
Teaching Hospital Maribor - Slovenia

Contents

Figures

Introduction

Summary

This document describes the final prototype of the WP6 SimBio project.

The purpose of this package is to allow all SimBio codes, and also in the future: external codes, to interact between each other inside the SimBio environment.
The interoperability of the environment's components will be realised using a portable, object-oriented, interoperable architecture, such as CORBA.

The purpose of this document is:
- To present current status of the WP6 prototype.
- To describe WP6 prototype installation and functionalities.

The Component Interaction Work Package Definition

The goal of this Component Interaction Module is to organise interaction between all SimBio components. The object-oriented implementation of the SimBio environment will allow to simply sending requests to the SimBio server that will organise a solution by calling the appropriate components.

The component-based SimBio environment will offer the ability of distributed, heterogeneous execution of its components. The Common Object Request Broker Architecture (CORBA) is the most appropriate client/server middleware for the object-oriented implementation of the SimBio system.
CORBA provides a high flexibility and enables the user to optimally exploit the computer resources being at his disposal. A CORBA object bus defines the "shape" of the SimBio components that reside within the bus and also fixes how these components inter-operate. Beyond interoperability CORBA specifies an extensive set of bus-related services for creating and deleting objects, accessing them by name, externalising their states and defining ad hoc relationships between them. To make use of CORBA for SimBio an interface has to be specified for each component. These specifications are written in the Interface Definition Language (IDL) and define a component's boundary, i.e. a "contractual" interface with potential clients.

A particular example of the need for interacting components arises with the simulation of a prosthetic implant, where close interactive links between the visualisation package, the material database, an external prosthesis database and the biomechanical solvers are required. The portability of CORBA will allow the SimBio-internal and SimBio-to-external interactions to take place in a seamless manner. The platform-independence of this approach is of decisive advantage in the clinical field as hospitals are often equipped with very heterogeneous computer clusters.

Some Definitions

1. ***SimBio Environment***: set of all applications, tools, products, etc. developed inside the SimBio project or used by SimBio applications.
2. ***SimBio Application***: tools developed or used inside the SimBio project such as the mesh generator, solver and so on.
3. ***SimBio Component***: each application and its interface (GUI, CORBA, Web Access…) developed inside the SimBio Environment; each component provides a list of services inside the SimBio Environment.
4. ***SimBio Service***: functionality provided by a SimBio component such as mesh generation, mesh visualisation, cropping, etc, for example.

The WP6 Prototype Definition

After a lot of discussions with every partners involved in the SimBio project, depending on their needs, it has been decided the following.

Partners asked WP6 to develop a simple design concept for the Component Interaction. The basic idea was to have something with a complexity similar to shell scripts but easier for the novice to use.

Furthermore, the SimBio environment should be comprised of two sites:

- At the first site, the complete pre-computing will be performed. To be more precisely: independent on the employed tools and their interactions, the output of this phase is the corresponding mesh. We will call this phase local.
- The other site, we call it “Remote compute server”, is responsible for performing the designated simulations. The output of the “Remote compute server” is the final visual data set.

All those users who do NOT intend to use this dedicated SimBio “Remote compute server” have to take care about the simulation phase themselves.

To generate the SimBio request that will describe the user’s simulation, we will provide to the end-user an editor. This editor will be used to define the user needs (application and parameters) and then will generate a SimBio Request (or Simulation request) File.
This editor will be called ***‘SimBio Scenario Editor’*** and developed by ESI.

The contents of the Request will be analysed by a special application. This program launches the programs sequentially according to their appearance in the request. It is also in charge of initiating the calls for the “Remote compute server” related part. This application is called ***‘SimBio pilot’*** and was developed at NEC. The implementation of the **“Remote compute server”** functionality has also been taken over by NEC.

Software Dependencies

Introduction

As currently conceived, the SimBio pilot will not have dependencies on other software as NEC will only be delivering binaries for Linux computers.

On the other hand, the SimBio editor will need installation of dynamic shared Apache Xerces-C++ library to run; this library could be provided by ESI (Linux version) or could be compiled by the end-user from the Xerces-C++ sources package.

ESI and NEC will provide binaries for Linux computers. If somebody asked, we could also compile it on other platforms; for the moment only SGI has been already done for debugging purpose during the WP6 development.
ESI SimBio Editor source codes could also be provided if asked.

CORBA

For remote computing on the NEC-compute server, the users will require a local installation of the ORBacus CORBA environment. Furthermore, there are some firewall issues which need to be negotiated between the user and NEC.

QTÔ

The SimBio Editor graphic interface was created using the QT Library but installation of QT is not necessary to run the SimBio Editor.

The following text describing QT comes from http://www.trolltech.com/products/qt/ and more information could be found on this WEB Site.
Qt is a C++ toolkit for application development. It lets application developers target all major operating systems with a single application source code.
Qt provides a platform-independent API to all central platform functionality: GUI, database access, networking, file handling, etc. The Qt library encapsulates the different APIs of different operating systems, providing the application programmer with a single, common API for all operating systems. The native C APIs are encapsulated in a set of well-designed, fully object-oriented C++ classes.

A free version of QT is available for **non-commercial** use under the GNU general public license and can be dowlnoaded over the internet.

XML Simulation Request File Description

The File format used for this Simulation Request file is XML (eXtended Marked Language) format.
This XML format is also used to save SimBio Editor preferences.
You will find short descriptions of the SimBio Request XML and SimBio Editor Preferences XML files in the following sections.
There is also an XML Simulation example file with the corresponding Document Type Definition (DTD) file at the end of the document.
For more information about XML format, please consult http://www.xml.org/, http://www.w3.org/XML/ or http://xml.apache.org/index.html.

You will find one XML Simulation and XML preferences example files on the Annexes in this document and the corresponding Document Type Definition (DTD) file.

The XML Simulation Request file contains the following information in addition to the requested XML heading:

1. Simulation Request ID.

2. User Environment description:
   - Working directory,
   - Log file,
   - Notify address: person to prevent when simulation is finished.

3. Request Sequence: list of applications to be run.

4. Each Request (called application) contains the following parameters:
   - Application name (command name).
   - Standard input filename.
   - Standard output filename.
   - Standard error filename.
   - Command line parameters; each parameter is made of option name (command line name) and corresponding value.

Apache XERCESÔ

Please have a look on http://xml.apache.org/xerces-c/index.html for more information (and also source codes) about Xerces.
Xerces-C++ is a validating XML parser written in a portable subset of C++. Xerces-C++ makes it easy to give your application the ability to read and write XML data. A shared library is provided for parsing, generating, manipulating, and validating XML documents.

Xerces-C++ library has been used to read and write XML inside SimBio Editor.
Dynamic shared libraries for Xerces could be provided by ESI if necessary on Linux and SGI or could be compiled by the end-user with the Xerces-C++ sources package.

WP6 Scenario Editor - User's Manual

Introduction

This is a simple GUI application, which allows creating scenarios by selected desired components (application), define application parameters and then execute the scenario through the SimBio Pilot.

The ultimate goal of this GUI is to provide each class of users a means to automatically generate the corresponding SimBio request. The SimBio request describes the complete pre and/or post processing. The ideal GUI should be capable to reflect a dynamically changing pool of services.
However, since we are aware of the fact, that the generation of such a tool is extraordinary complex on the one hand, and imposes a significant overhead to the majority of (static) cases on the other we propose to deal with this issue in the following way:

1. **Step 1:**
   Development of a GUI for controlling the fixed set of tools which currently constitute the backbone of the project. The GUI should generate an XML file that serves as a control file for pre and/or post processing. Thus, every user gets an elegant and efficient tool for driving the environment.
2. **Step 2:**
   Due to the fact that we cannot assume that the set of tools remains constant in the future, a professional GUI should be able to dynamically adapt itself to changes. For this advanced tool we propose to base the generation of the GUI on an XML file containing the explicit descriptions of all participating tools. However, the development of this GUI should NOT be mandatory for the scope of WP6. Anyway, the reflection about this step has already started at ESI and the current editor prototype allows such possibilities in a simple way. Depending on the job to do during the next month of the project (improvement, debugging, modifications, partners need after the evaluation phase), it will be possible to improve this functionality if necessary.

Technical Description

This application is written in C++ and used QT library (http://www.trolltech.com/products/index.html) to define GUI and Xerces-C++ (http://xml.apache.org/xerces-c/index.html) for XML read and write. See Software dependencies to have more information about these two libraries.

Installation

The user must first install all the applications from the other workpackages that he/she wants to use. Each application executables must be in the current user path that the editor could launch them. Then they install the editor and the pilot. Then if the user wants to run jobs remotely, they need to install the CORBA package mentioned above and finally the executables that manage the communication to NEC's compute server.

The SimBio editor distribution will contain the following files:

bin/irix32_release_wp6Editor	SimBio editor SGI executable.
bin/linux_release_wp6Editor	SimBio editor Linux executable.
bin/wp6_editor.csh	A simple script to automatically select the xerces library and select the right executable depending on the current platform.
lib_irix/ libxerces-c1_7_0.so	SGI Xerces-C++ dynamic shared library.
lib_linux/ libxerces-c1_7_0.so	Linux Xerces-C++ dynamic shared library.
resources/simbio-editor-config.dtd	DTD file necessary to read and write preference files.
resources/simbio-request.dtd	DTD file necessary to read and write scenario files.
scenarios/*.xml	Some scenario examples.

SimBio Scenario Editor Main Interface

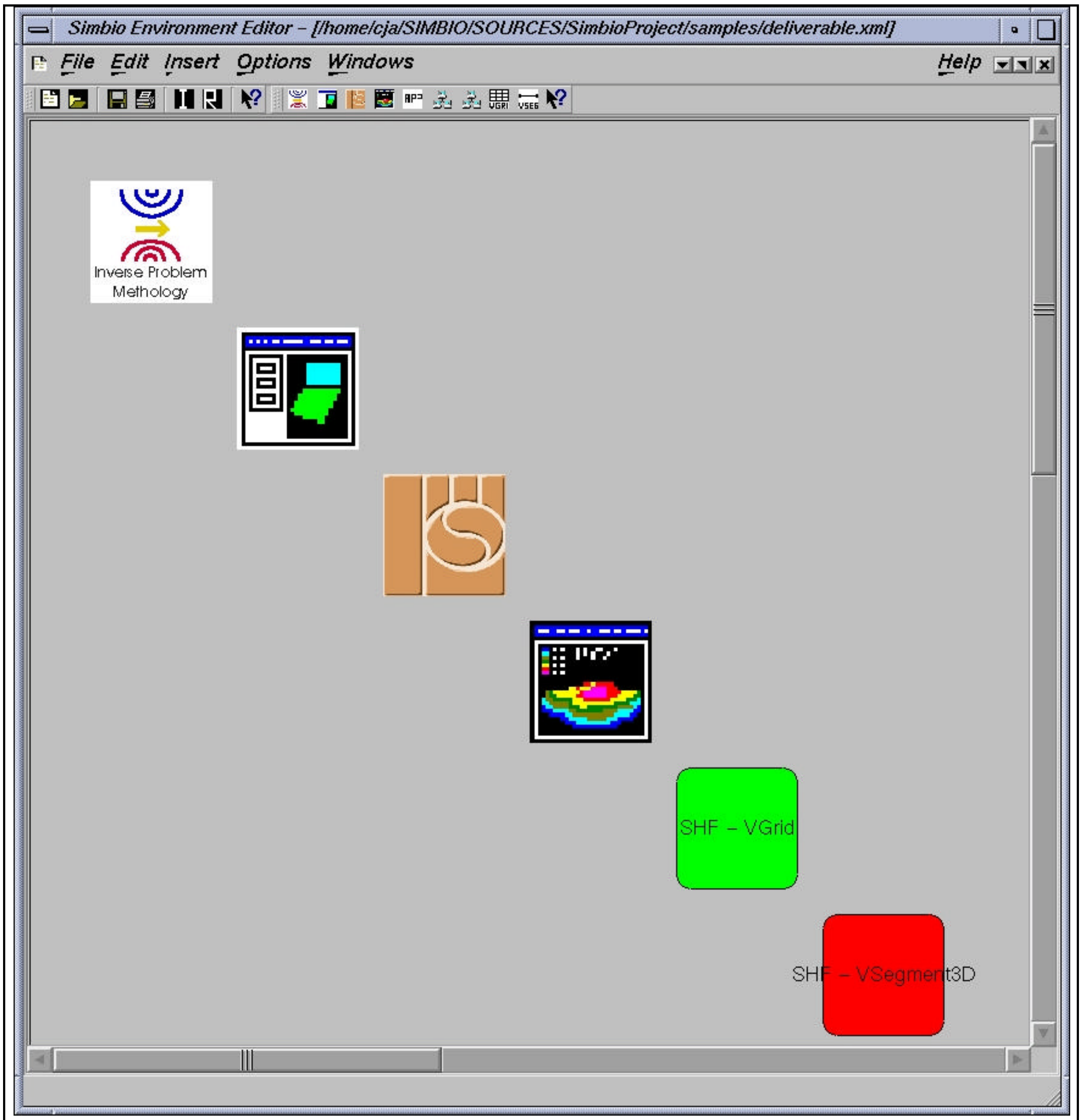

Simbio Environment Editor – [/home/cja/SIMBIO/SOURCES/SimbioProject/samples/deliverable.xml]
File
Edit
Insert
Options
Windows
Help
Inverse Problem Methology
SHF – VGrid
SHF – VSegment3D

Figure 1: SimBio Scenario Editor – Main Window

Each component is represented with a simple square and the component name.
Application icons instead of square are used when available such as Inverse Problem (IPM), Generis, PamSafe and Pamview applications in this example.
Selected component appears in red, others in green; when icon is used, cached icon is used instead of red.

The first toolbar (on the left) contains some tool buttons with important (often-used) functionality such as New, Open, Save, Print, Properties Run and Help.

The second toolbar (on the right) contains list of available components. It is exactly the same contents than Insert Menu.

File Functionality

Here is the list of all file menus available from this SimBio Scenario Editor.

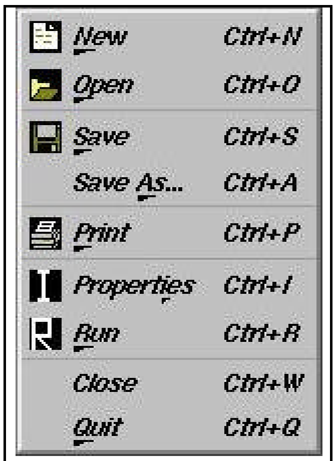


Figure 2: File Menu

- ♦ New: Create a new empty scenario. A new empty Scenario Window appears and becomes the one active. Next calls to Save, Save As, Print, Close, Edit menus and Insert menus will concern this new active window. Once this new scenario is created, the simulation properties dialog box appears and must be completed by the end-user.

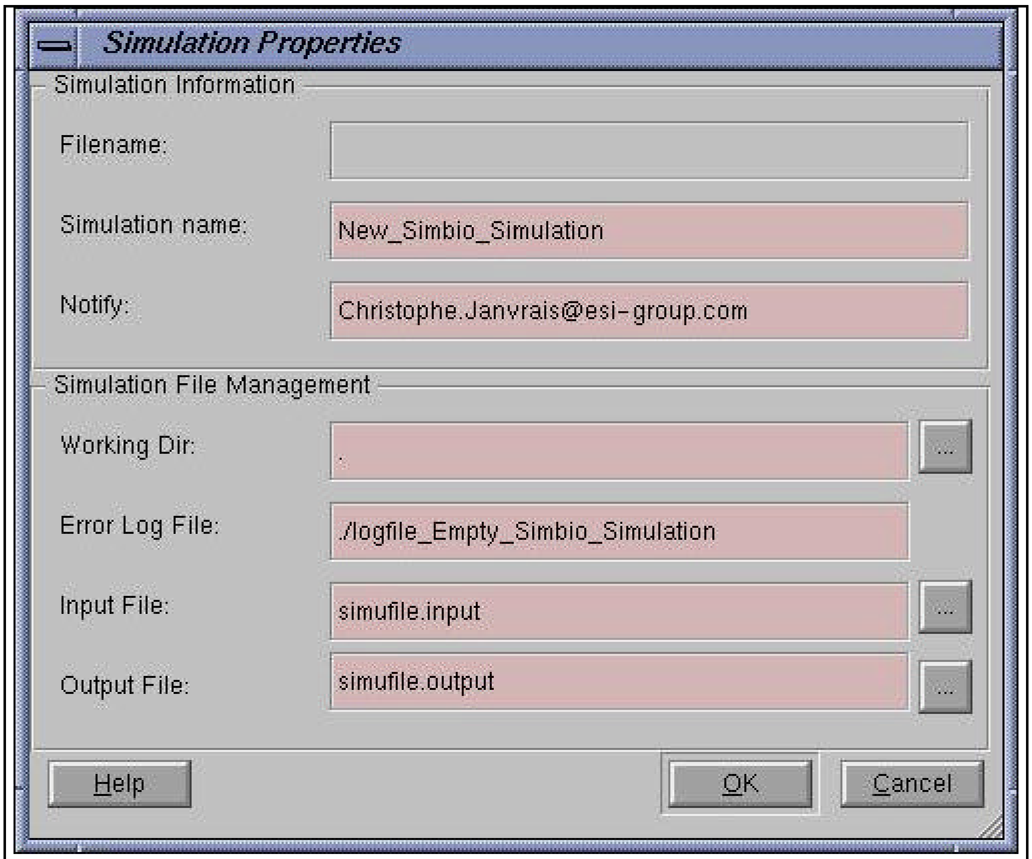
- ♦ Open: Open a file selector where user can select a scenario file. Then this file is loaded and a new window will appear to display its content. If the load operation failed, an error message is displayed and no new window is created.

- ♦ Save: Save the current XML scenario file. If no save has already been done, a name and location are asked and the new file is created, otherwise the current file is updated.

- ♦ Save As: A name and location are asked and the new scenario file is created.

- ♦ Print: Not yet implemented. We are currently thinking about that function: print the XML file or print the graphical part of the interface or something else. Still to be discussed depending on partners needs.

♦ Properties: Display the properties (filename, simulation name, notify address, working directory, error log file, input file and output file) of the current simulation request. See below. All these information are save in the scenario file.



Filename:
Simulation name: New_Simbio_Simulation
Notify: Christophe.Janvrais@esi-group.com
Working Dir: .
Error Log File: ./logfile_Empty_Simbio_Simulation
Input File: simufile.input
Output File: simufile.output

Figure 3: Simulation Request Properties Dialog

♦ Run: Call the SimBio Pilot to run the current simulation. If modifications have been done, save function is called before. Default called application is the one defined with environment variable called ‘SIMBIO_PILOT_PROG’, if no variable is defined, SimBio pilot ‘simbio-pilot’ is called and must be well defined in the current user path.
The only parameter to this application or script is the name of the XML Simulation Request file.

♦ Close: If modifications have been done, save function is called and the current window is closed, otherwise current window is simply close.

♦ Quit: Same than close function but applied to all windows of the editor. Once all windows are closed, the application is exiting.

Windows Functionality

Functions used to manage SimBio Scenario Editor windows.

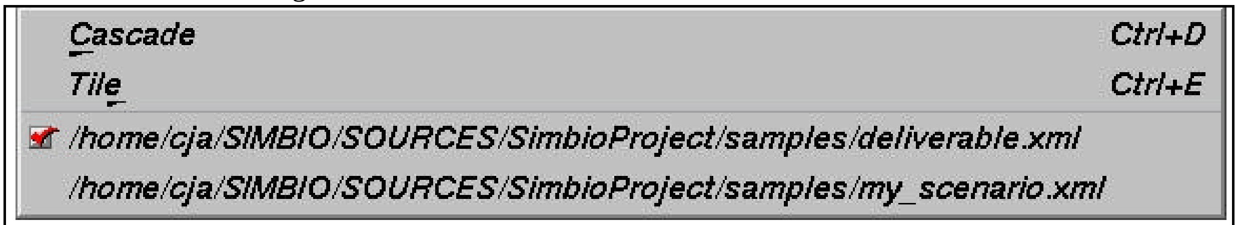

Cascade Ctrl+D
Tile Ctrl+E
/home/cja/SIMBIO/SOURCES/SimbioProject/samples/deliverable.xml
/home/cja/SIMBIO/SOURCES/SimbioProject/samples/my_scenario.xml

Figure 4: Windows Menu

- Cascade: All existing scenario windows are displayed one by one above the others.
- Tile: All existing scenario windows are displayed one by one beside the others.
- Window 1…n: The selected window is displayed in front of the others and becomes active. The red sign shows the current active window.

Help Functionality

Functions used to manage SimBio Scenario Editor help.

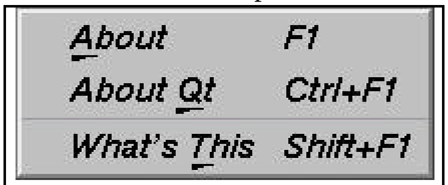


Figure 5: Help Menu

- About: Display a dialog with a very simple definition (name, short description, authors and version) of the SimBio Scenario Editor.
- About QT: Display a dialog with a simple definition remembering QT usage.
- What's this: The current icon becomes "?". Click on a menu or a button to display a short description of its functionality.

Edit Functionality

Functions used to manage applications in the current scenario.

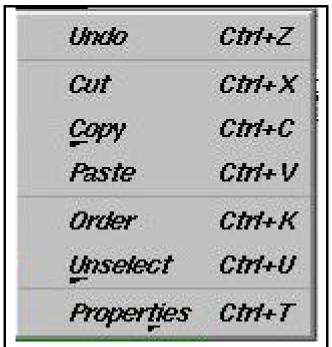
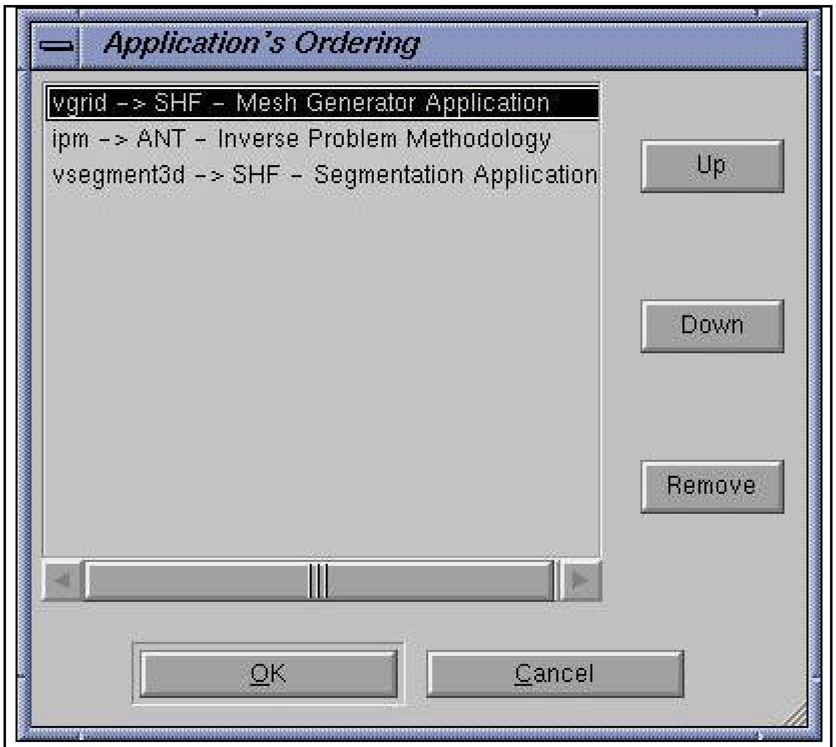


Figure 6: Edit Menu

- Undo: Not yet implemented. Should cancel the last operation done (only one level). Not really useful at the moment.
- Cut: Remove selected component from the current simulation but keep it in the SimBio Scenario Editor clipboard.
- Copy: Copy selected component from the current simulation to the SimBio Scenario Editor clipboard.
- Paste: Copy SimBio Scenario Editor clipboard to the current simulation.
- Order: To change order of applications in the current scenario. Can also be used to remove applications.



vgrid -> SHF - Mesh Generator Application
ipm -> ANT - Inverse Problem Methodology
vsegment3d -> SHF - Segmentation Application

Figure 7: Application Ordering Dialog Box

- Unselect: Remove component selection. In this case, Cut and Copy becomes no more available.
- Properties: Open the Properties Panel Dialog corresponding to the selected component.

Insert Application functionality

Here is the list of functions used to manage SimBio Scenario Editor components. The user could choose the components he wants to use during its simulation; he is also requested to set every parameter. Once the user selects one component, the corresponding property panel is displayed and needs to be validated. If the current component is already inserted in the current simulation, the operation is cancelled except if several calls to the same component is allowed.

For the moment, 9 applications are available:

1. ipm from ANT-Software,
2. Generis from ESI France,
3. PamSafe from ESI France,
4. Pamview from ESI France,
5. NeuroFEM from MPI,
6. Remote HeadFem from MPI and NEC,
7. Remote NeuroFEM from MPI and NEC,
8. vgrid from University of Sheffield and
9. vsegment from University of Sheffield.

Probably, only few SimBio applications will be added in the future.

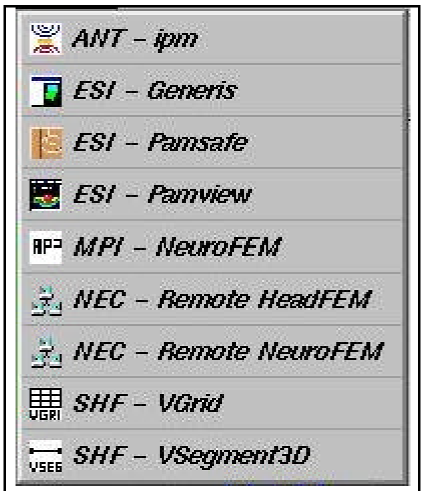
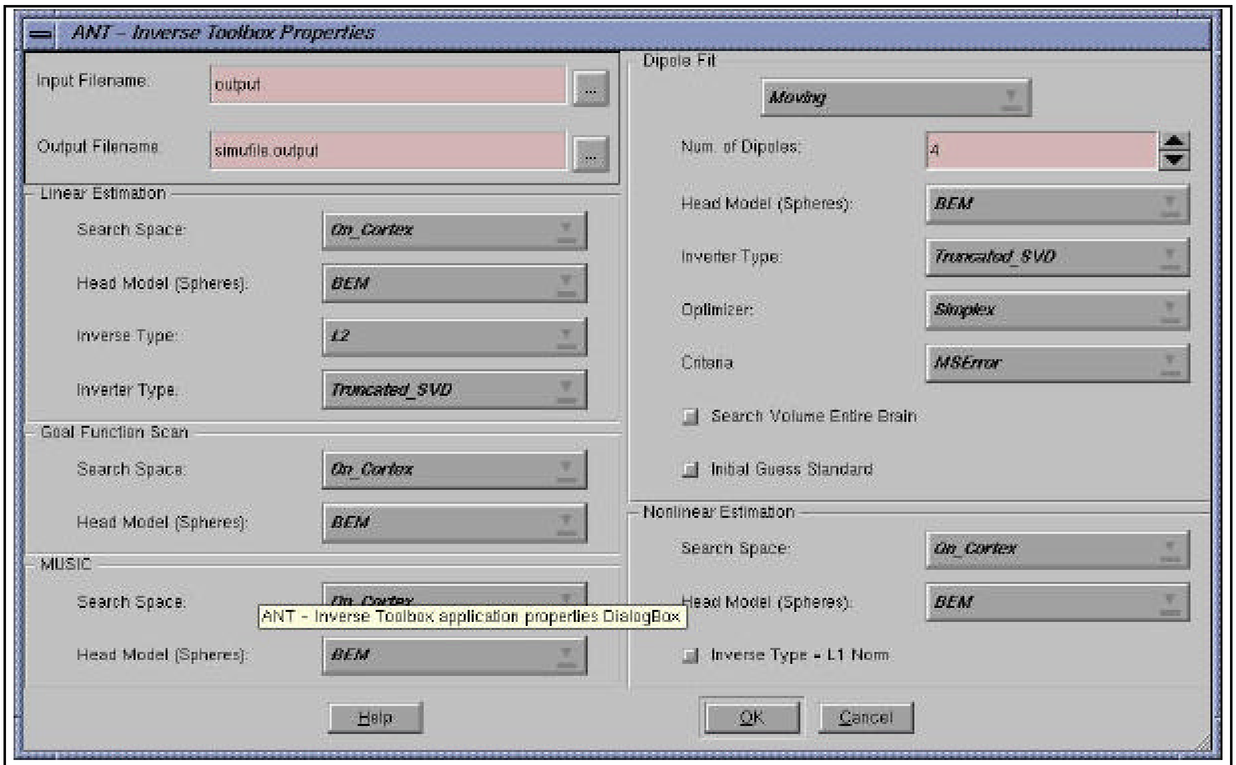


Figure 8: Insert Menu

In the next page we will see each Properties Dialog Box the user has to fill to define component parameters that correspond to the current status of development for each application.
See below images to have more information on this parameter that corresponds to the command line option of the described component.
Please have a look on each application user's manual to have more information on each parameter.
Depending on possible modification on each component/application, these dialog boxes could be modified in the future.


ANT – Inverse Toolbox Properties

Input Filename: output

Output Filename: simufile.output

Linear Estimation
- Search Space: On_Cortex
- Head Model (Spheres): BEM
- Inverse Type: L2
- Inverter Type: Truncated_SVD

Goal Function Scan
- Search Space: On_Cortex
- Head Model (Spheres): BEM

MUSIC
- Search Space: On_Cortex
- Head Model (Spheres): BEM

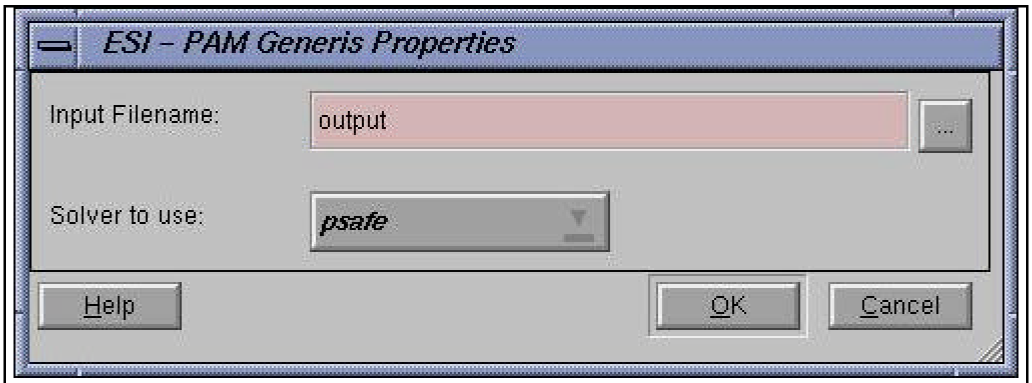
ANT - Inverse Toolbox application properties DialogBox

Dipole Fit
- Moving
- Num. of Dipoles: 4
- Head Model (Spheres): BEM
- Inverter Type: Truncated_SVD
- Optimizer: Simplex
- Criteria: MSError
- Search Volume Entire Brain
- Initial Guess Standard

Nonlinear Estimation
- Search Space: On_Cortex
- Head Model (Spheres): BEM
- Inverse Type = L1 Norm

Figure 9: ANT-Software - Inverse Toolbox (ipm) Properties Panel

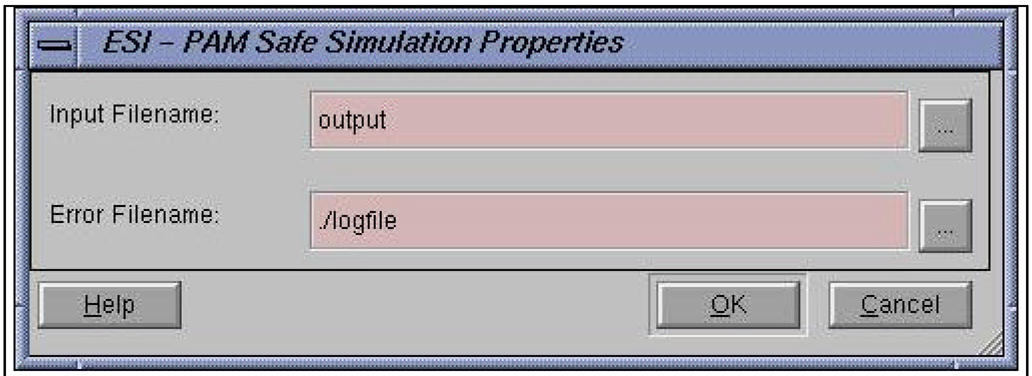
- This one must be updated in the nearby future.


ESI – PAM Generis Properties

Input Filename: output

Solver to use: psafe

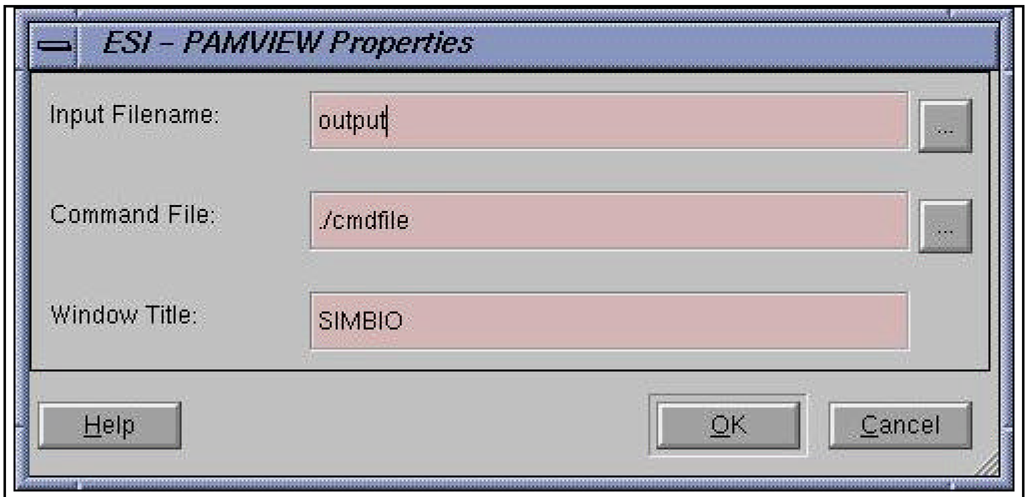
Figure 10: ESI - PAM Generis Properties Panel


ESI – PAM Safe Simulation Properties

Input Filename: output

Error Filename: ./logfile

Figure 11: ESI - PAM Safe Properties Panel

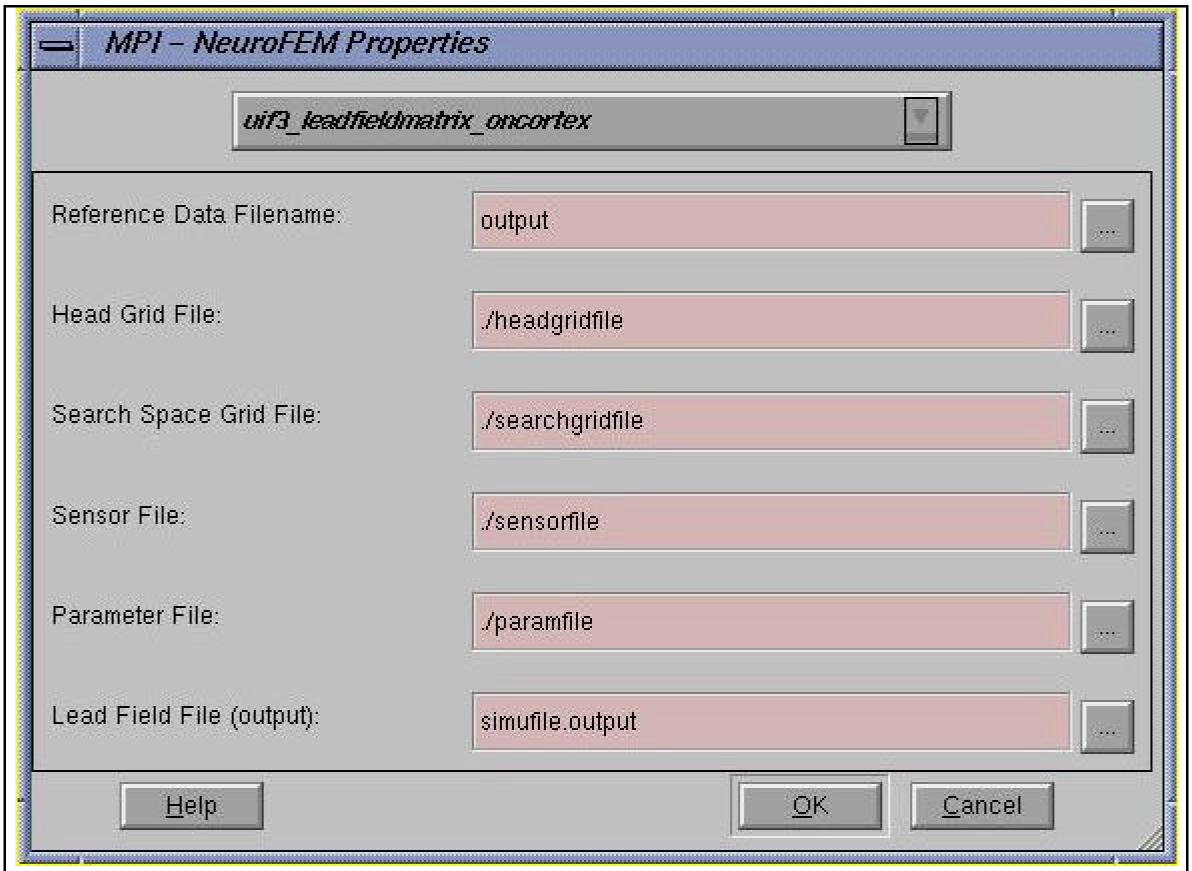


Input Filename: output

Command File: ./cmdfile

Window Title: SIMBIO

Figure 12: ESI - Pamview Properties Panel



Reference Data Filename: output

Head Grid File: ./headgridfile

Search Space Grid File: ./searchgridfile

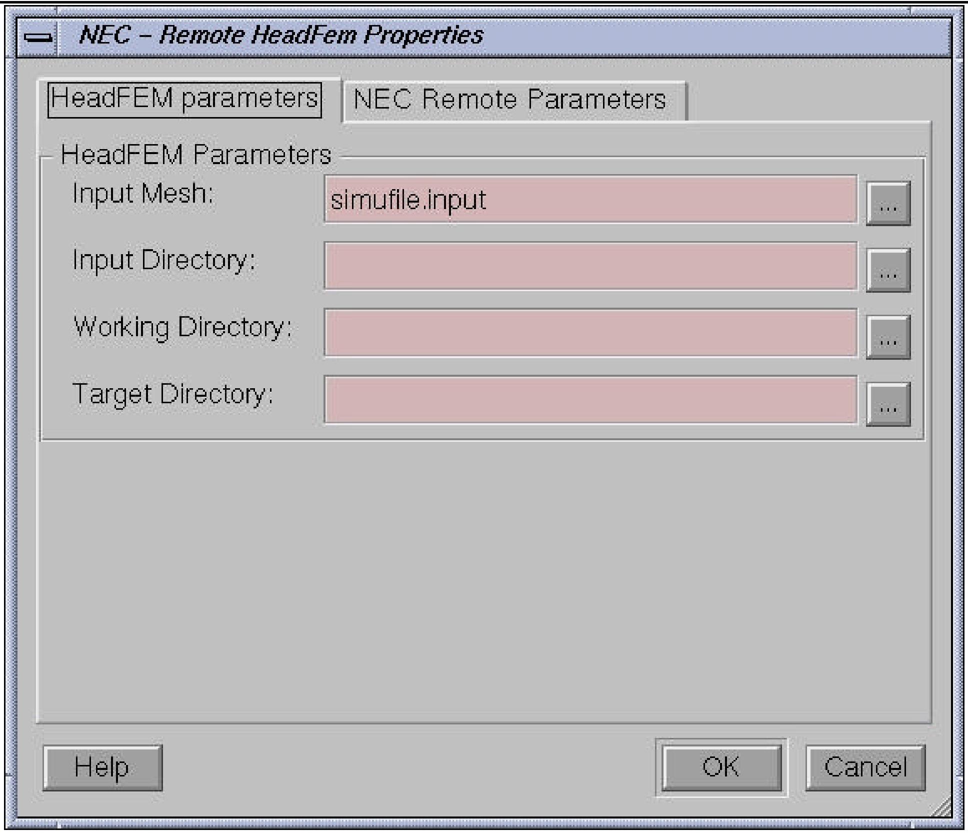
Sensor File: ./sensorfile

Parameter File: ./paramfile

Lead Field File (output): simufile.output

Figure 13: MPI - NeuroFEM Properties Panel

- Reference data filename will be removed in the nearby future.

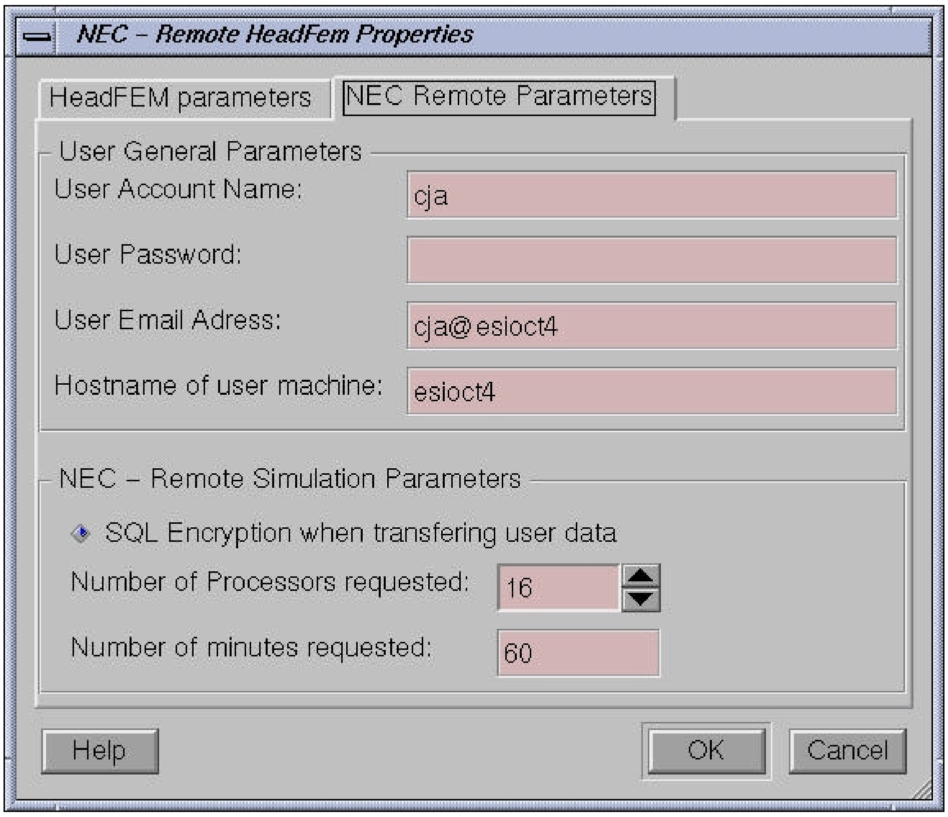

NEC – Remote HeadFem Properties

HeadFEM parameters | NEC Remote Parameters

HeadFEM Parameters

Input Mesh:	simufile.input
Input Directory:	
Working Directory:	
Target Directory:	

Help OK Cancel


NEC – Remote HeadFem Properties

HeadFEM parameters | NEC Remote Parameters

User General Parameters

User Account Name:	cja
User Password:	
User Email Adress:	cja@esioct4
Hostname of user machine:	esioct4

NEC – Remote Simulation Parameters

◆ SQL Encryption when transfering user data

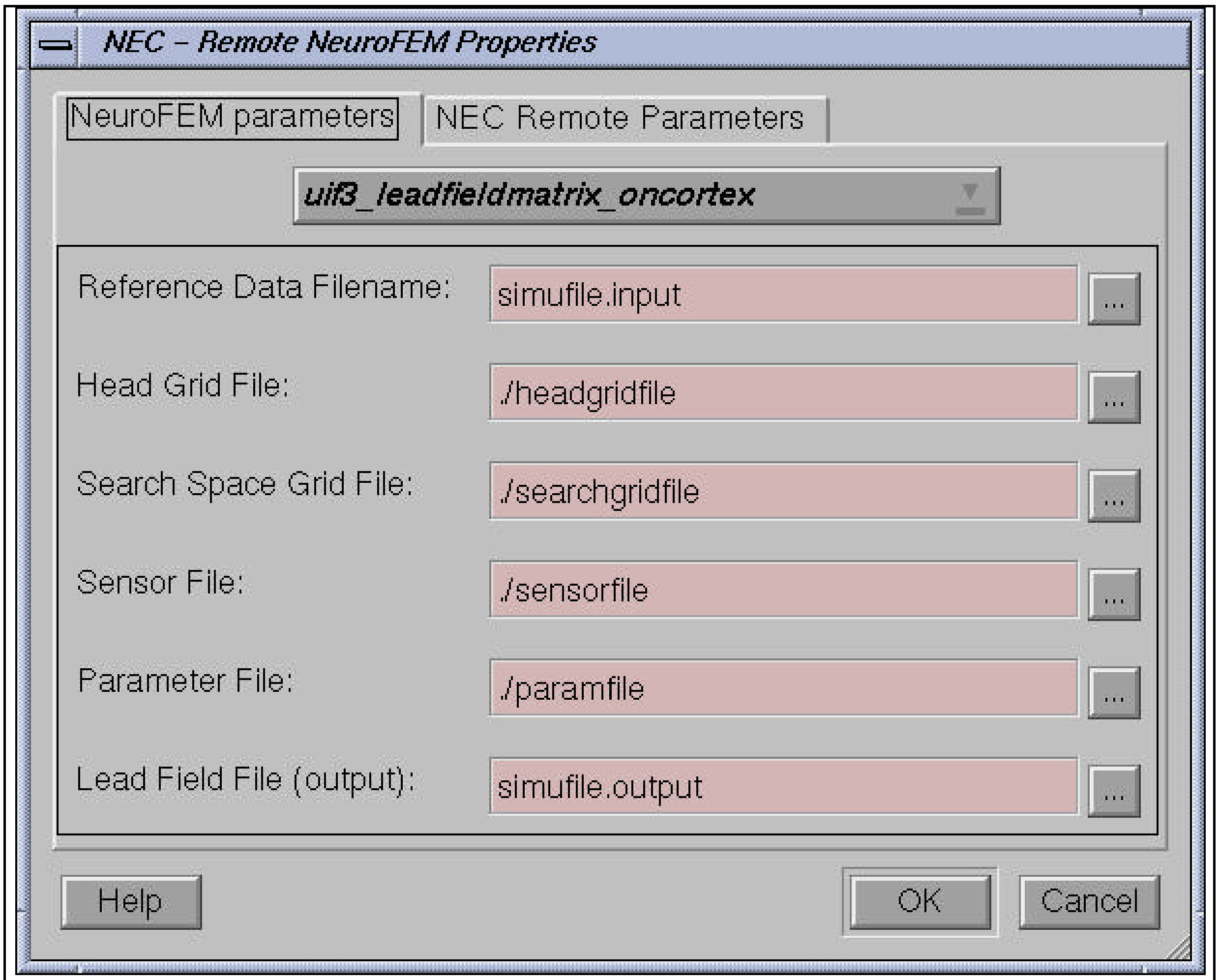
Number of Processors requested:	16
Number of minutes requested:	60

Help OK Cancel

Figure 14: MPI + NEC - Remote HeadFEM Properties Panel

- HeadFEM is a MPI product.
- Remote (“Remote compute server”) availability of HeadFEM was made by NEC.
- This dialog box show all parameters needed to connect to the “Remote compute server” developped by NEC that means account name, account password, user email adress, hostname, data encryption or not, number of processors requested and number of minutes

requested.


NEC – Remote NeuroFEM Properties

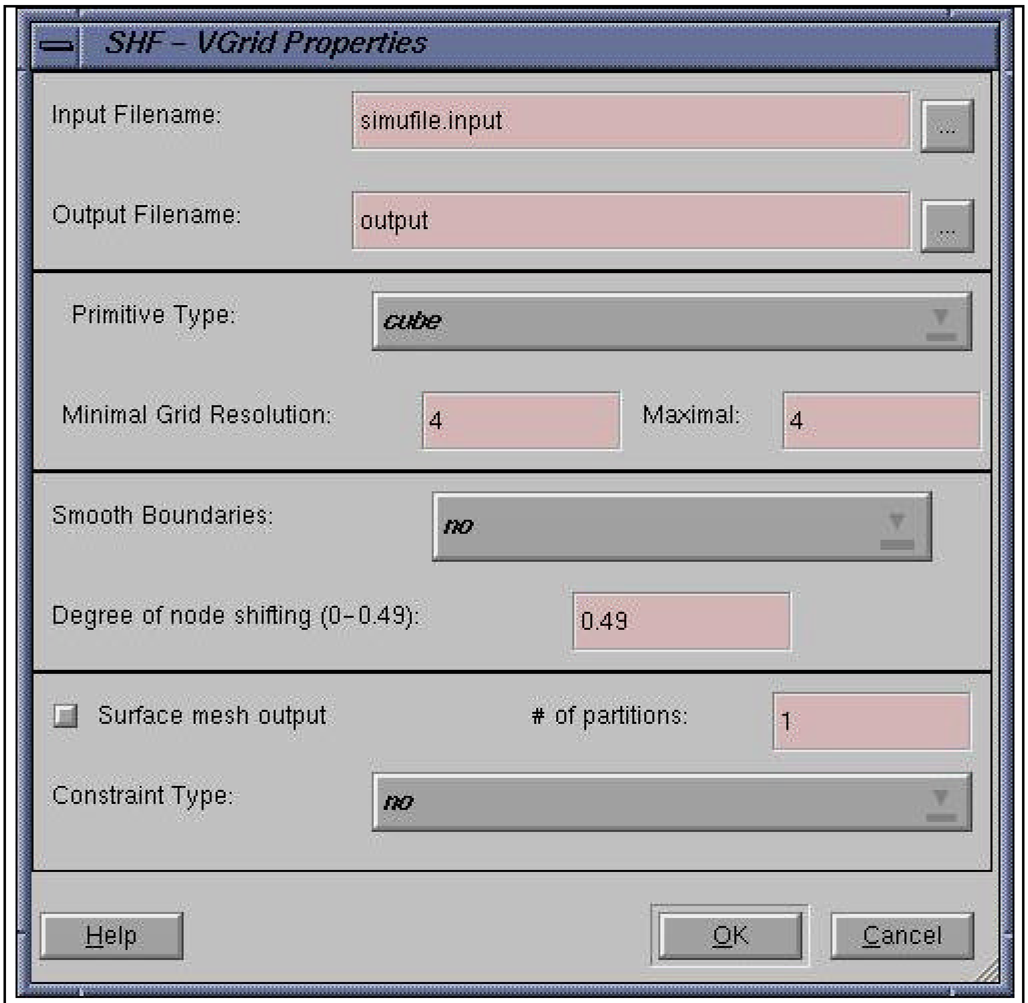
NeuroFEM parameters | NEC Remote Parameters

uif3_leadfieldmatrix_oncortex

Reference Data Filename:	simufile.input
Head Grid File:	./headgridfile
Search Space Grid File:	./searchgridfile
Sensor File:	./sensorfile
Parameter File:	./paramfile
Lead Field File (output):	simufile.output

Figure 15: MPI + NEC - Remote NeuroFEM Properties Panel

- NeuroFEM is a MPI product.
- Remote (“Remote compute server”) availability of NeuroFEM was made by NEC.
- Reference data filename will be removed in the nearby future.
- ***NEC Remote Parameters are exactly the same defined in HeadFem dialog box.***


SHF – VGrid Properties

Input Filename: simufile.input

Output Filename: output

Primitive Type: cube

Minimal Grid Resolution: 4 Maximal: 4

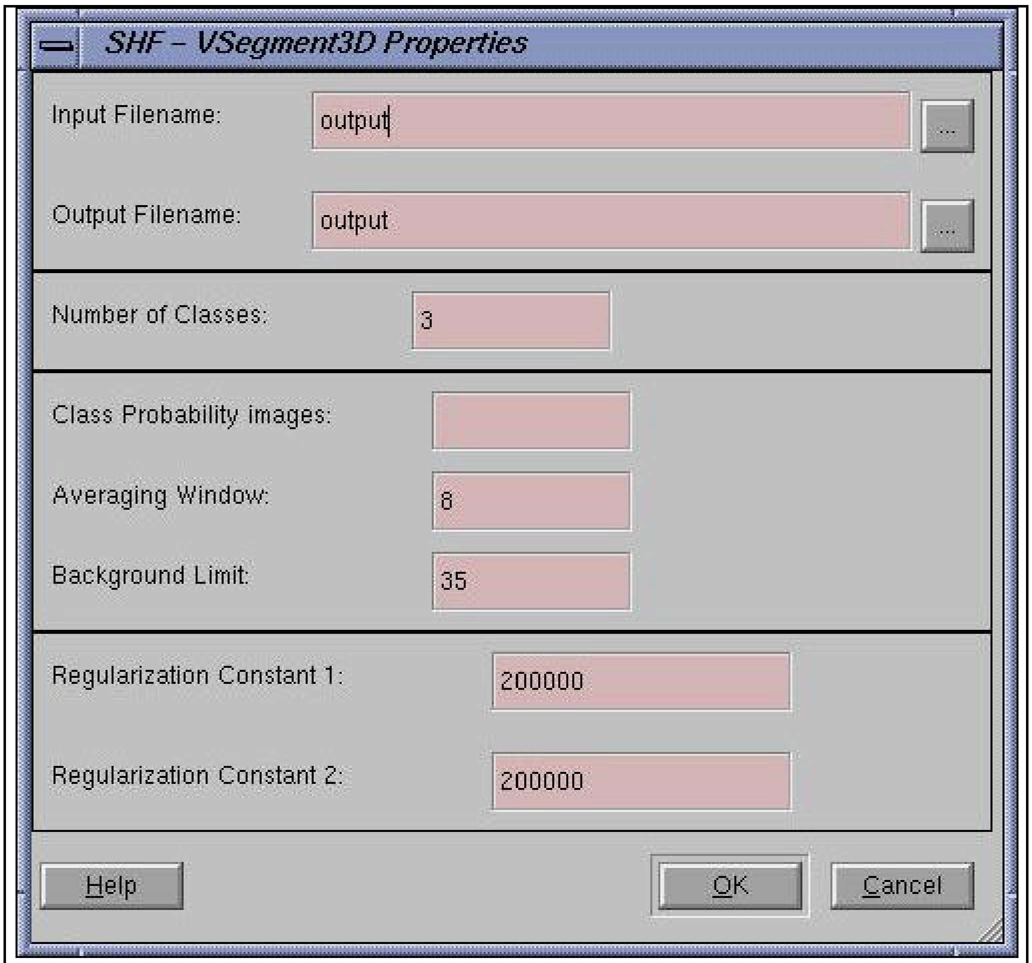
Smooth Boundaries: no

Degree of node shifting (0–0.49): 0.49

Surface mesh output # of partitions: 1

Constraint Type: no

Figure 16: Sheffield University - vgrid (Mesh Generator) Properties Panel



SHF – VSegment3D Properties

Input Filename:	output
Output Filename:	output
Number of Classes:	3
Class Probability images:	
Averaging Window:	8
Background Limit:	35
Regularization Constant 1:	200000
Regularization Constant 2:	200000

Figure 17: Sheffield University - vsegment (segmentation) Properties Panel

Options Functionality

Functions used to manage preferences in the SimBio scenario editor.

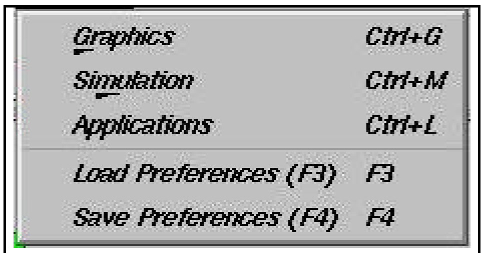
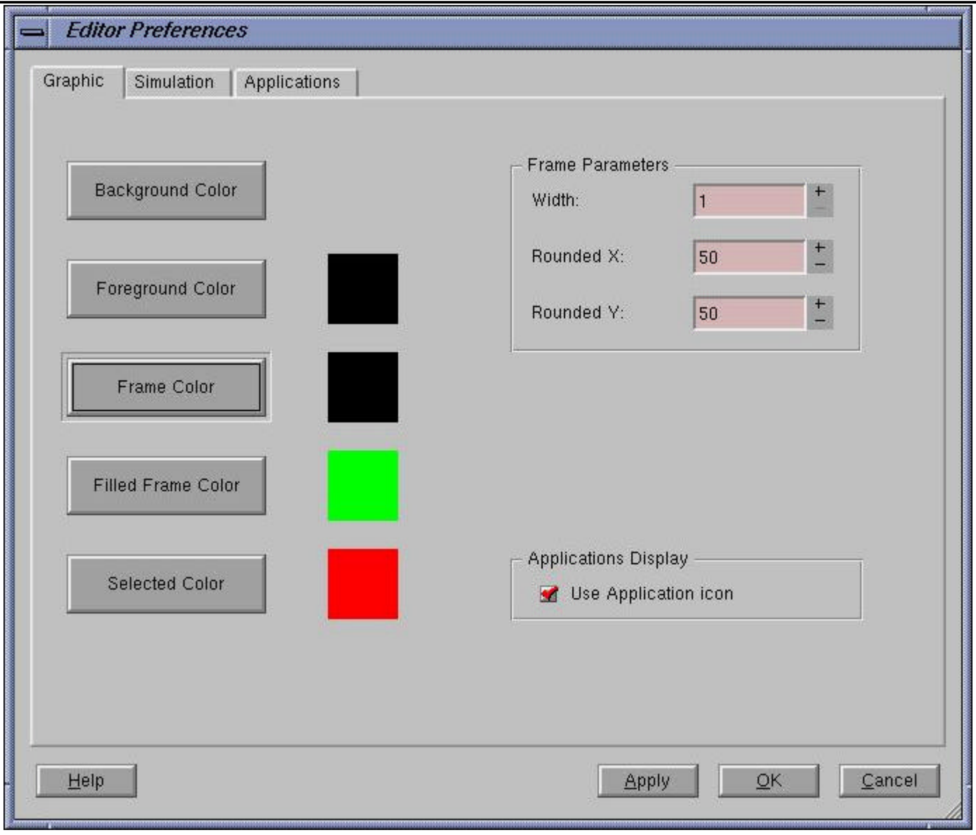


Figure 18: Options Menu

- Graphics: Allow to change some graphical parameters inside SimBio Editor.
- Simulation: Allow to define the default simulation parameters.
- Applications: Allow to manage applications/components available inside SimBio Editor.

Graphic

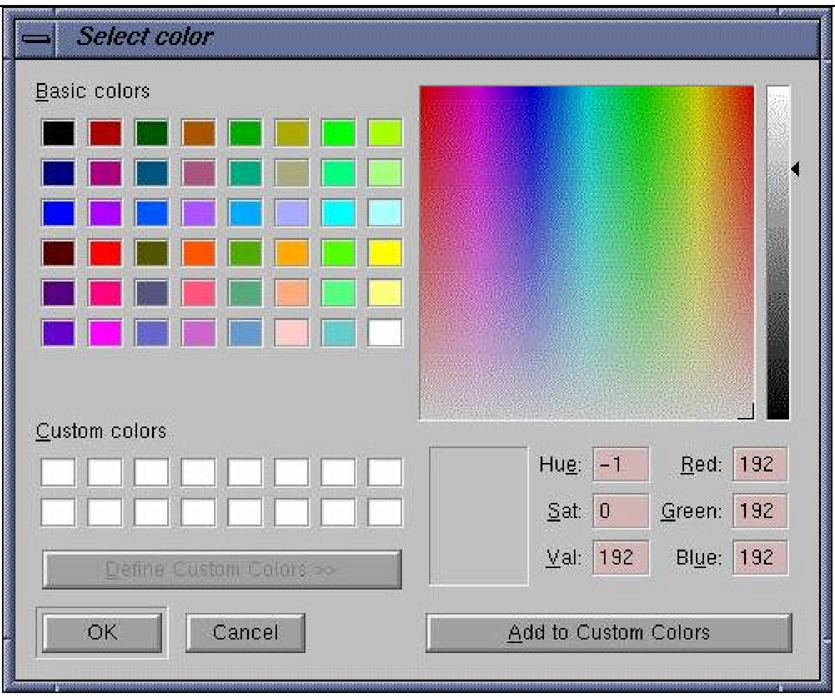
- User can modify colours used in the Simbio Editor main window.
  The following dialog box the open to allow colour selection.
- User can modify square shape used in the Simbio Editor main window.
- User could also select between square or application icons.



Width: 1
Rounded X: 50
Rounded Y: 50

Applications Display
✓ Use Application icon

Figure 19: Editor Graphic Preferences Properties Panel

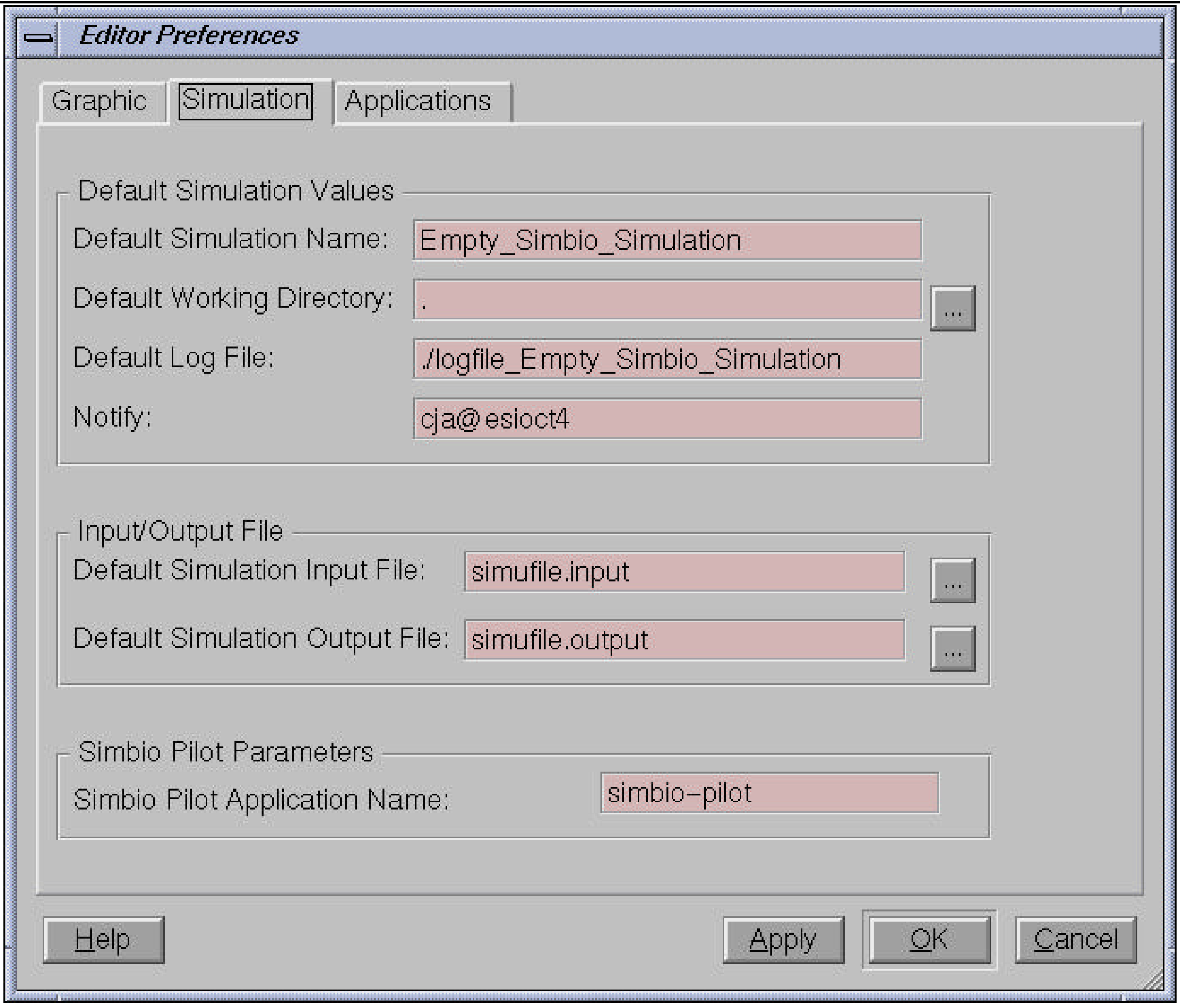


Hue: -1 Red: 192
Sat: 0 Green: 192
Val: 192 Blue: 192

Figure 20: Select Color Panel

Simulation

- User can modify all parameters used inside one simulation that means default name, default working directory, default log file, notify address, input and output files and also the name of the SimBio application pilot to be used to run the scenario.
- If user do not validate its choice with Apply or Ok buttons, the previous situation is restored.


Editor Preferences

Graphic | Simulation | Applications

Default Simulation Values

Default Simulation Name:	Empty_Simbio_Simulation
Default Working Directory:	.
Default Log File:	./logfile_Empty_Simbio_Simulation
Notify:	cja@esioct4

Input/Output File

Default Simulation Input File:	simufile.input
Default Simulation Output File:	simufile.output

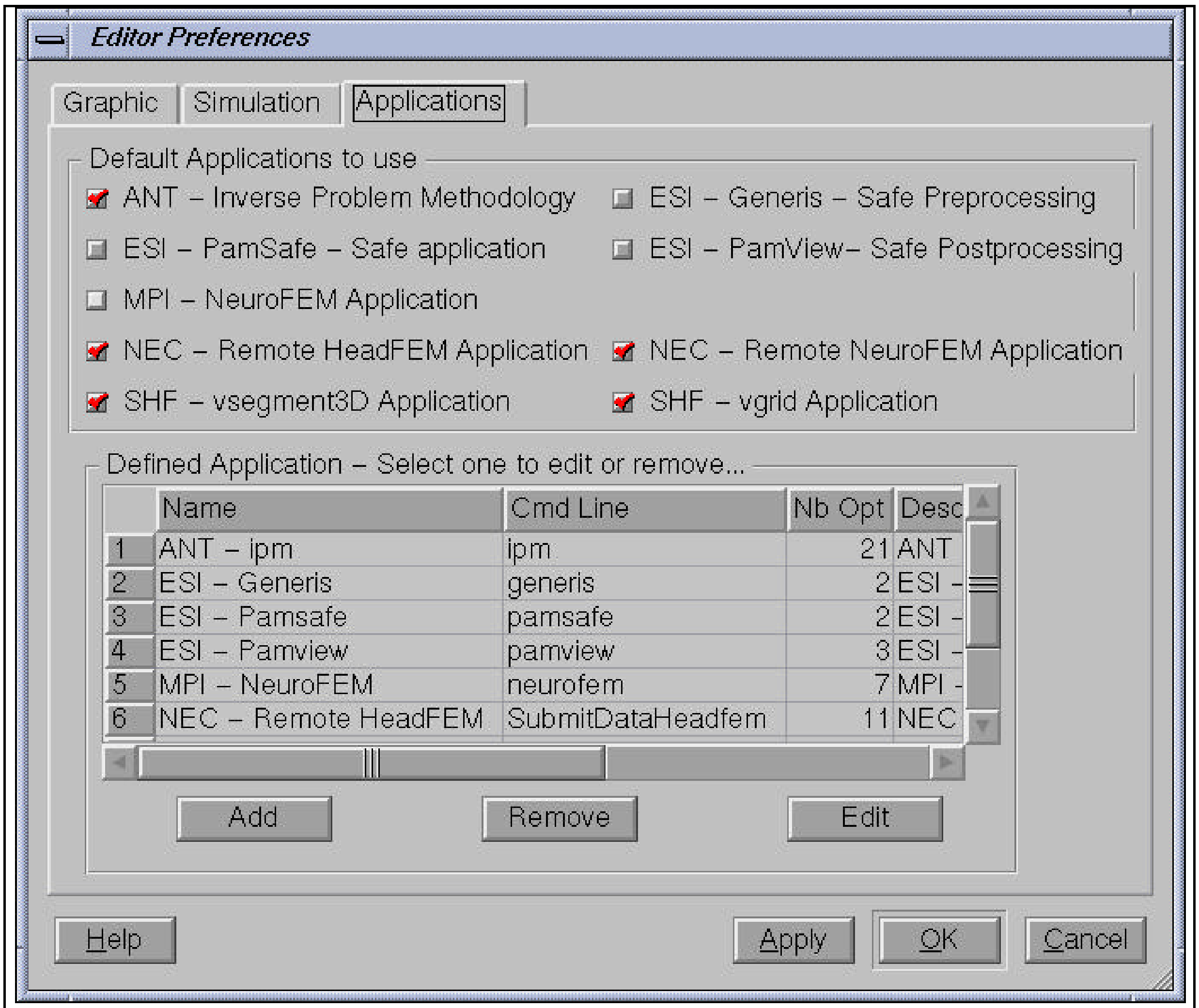
Simbio Pilot Parameters

Simbio Pilot Application Name:	simbio-pilot

Figure 21: Editor Simulation Preferences Properties Panel

Applications

- User can select the list of predefined applications he wants to use.
  If one is unselected, it will not appear in the toolbar and not in the Insert menu.
- Then the user could add its own application with its parameter (see next page).
- The user could also remove one added application; ***predefined applications could not been removed.***
- If user do not validate its choice with Apply or Ok buttons, the previous situation is restored.



Editor Preferences

Graphic | Simulation | Applications

Default Applications to use

- ✓ ANT – Inverse Problem Methodology
- ☐ ESI – Generis – Safe Preprocessing
- ☐ ESI – PamSafe – Safe application
- ☐ ESI – PamView– Safe Postprocessing
- ☐ MPI – NeuroFEM Application
- ✓ NEC – Remote HeadFEM Application
- ✓ NEC – Remote NeuroFEM Application
- ✓ SHF – vsegment3D Application
- ✓ SHF – vgrid Application

Defined Application – Select one to edit or remove...

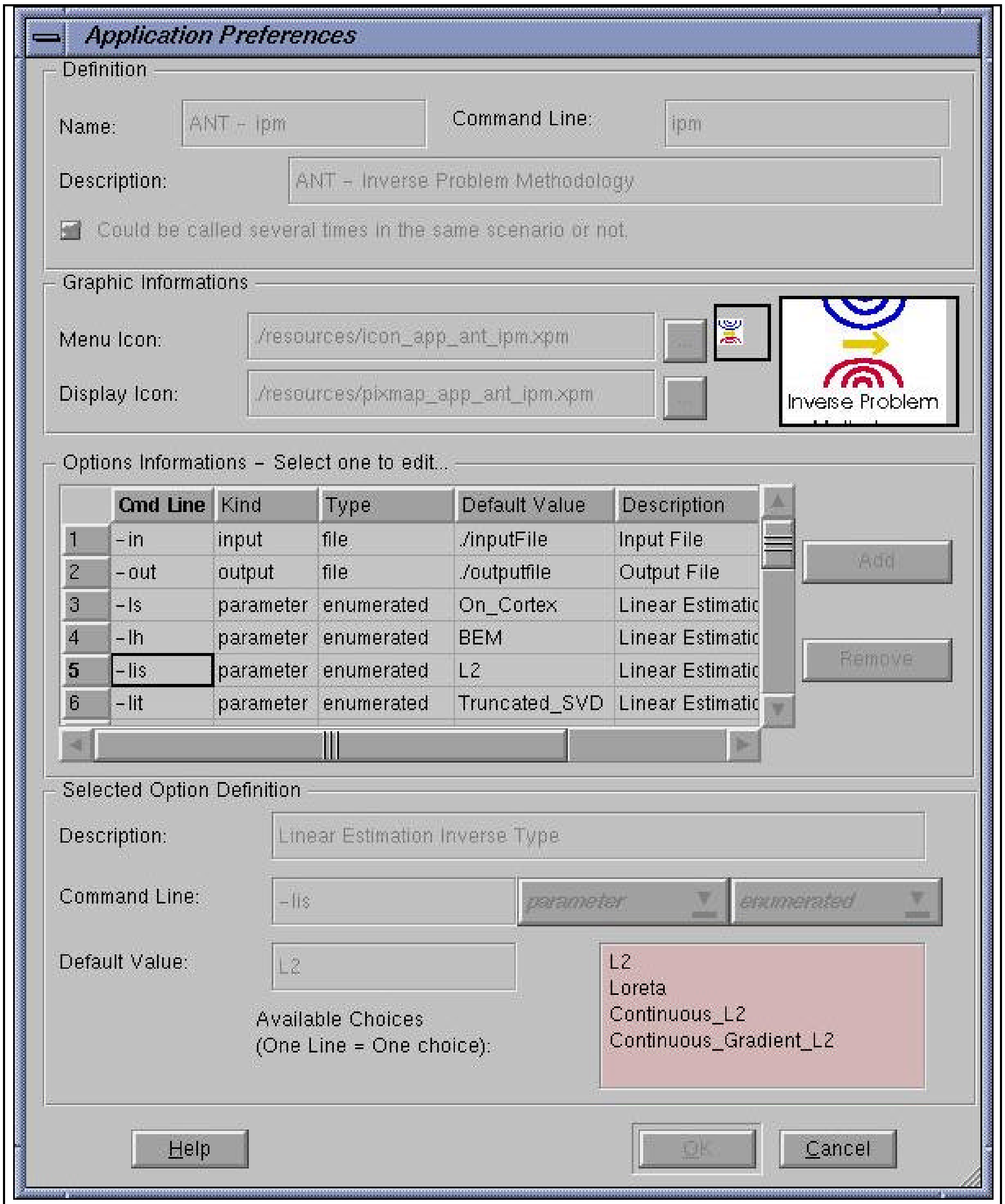
	Name	Cmd Line	Nb Opt	Desc
1	ANT – ipm	ipm	21	ANT
2	ESI – Generis	generis	2	ESI -
3	ESI – Pamsafe	pamsafe	2	ESI -
4	ESI – Pamview	pamview	3	ESI -
5	MPI – NeuroFEM	neurofem	7	MPI -
6	NEC – Remote HeadFEM	SubmitDataHeadfem	11	NEC

Figure 22: Editor Application Preferences Properties Panel

Add one application

When adding a new application the following dialog box appears.

- User must defined his new application, given a name, the command line to run it, a short description, icon to be used in menus, icon to be used in the graphic area and the list of all application options.
- When editing the scenario and this new application parameters (properties), an automatic dialog box will be generated using these information.
- Predefined application could only be read and no modification is allowed.



Name: ANT – ipm
Command Line: ipm
Description: ANT – Inverse Problem Methodology
Could be called several times in the same scenario or not.

Menu Icon: ./resources/icon_app_ant_ipm.xpm
Display Icon: ./resources/pixmap_app_ant_ipm.xpm

	Cmd Line	Kind	Type	Default Value	Description
1	-in	input	file	./inputFile	Input File
2	-out	output	file	./outputfile	Output File
3	-ls	parameter	enumerated	On_Cortex	Linear Estimatic
4	-lh	parameter	enumerated	BEM	Linear Estimatic
5	-lis	parameter	enumerated	L2	Linear Estimatic
6	-lit	parameter	enumerated	Truncated_SVD	Linear Estimatic

Description: Linear Estimation Inverse Type
Command Line: -lis parameter enumerated
Default Value: L2
Available Choices (One Line = One choice):
L2
Loreta
Continuous_L2
Continuous_Gradient_L2

Figure 23: Application Properties Panel

- Each application option must have a name, a description, a family (kind), a type and a default value. Input and output parameter could be used to automatically managed connection between following applications in the scenario.
  - Family must be selected between help, input, output and normal parameter.

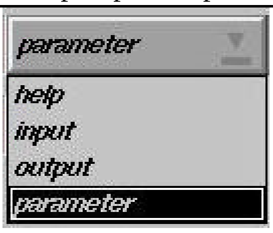


Figure 24: Parameter Family Selection

  - Type must be selected between string, file, directory, double (with or without range), integer (with or without range), boolean and enumerated.

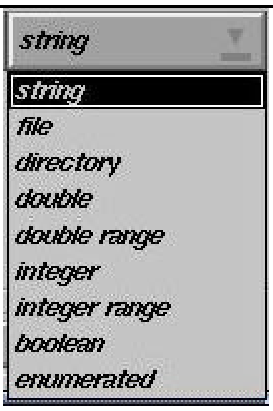
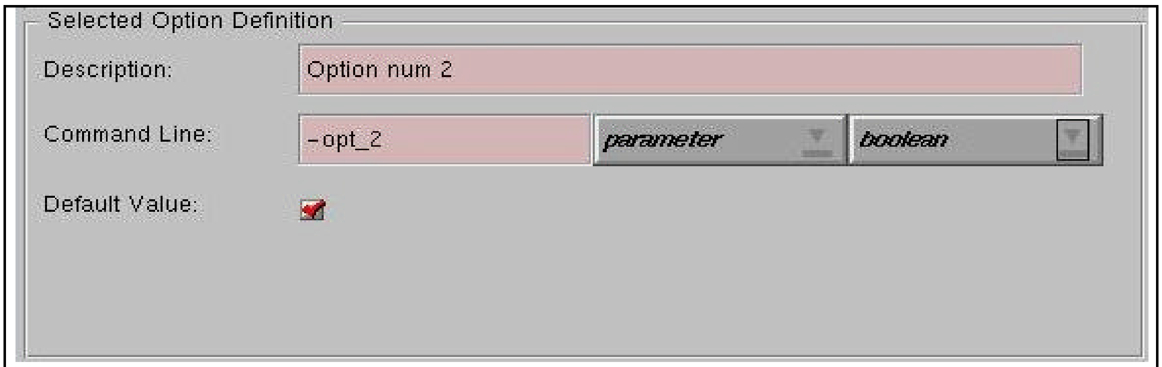


Figure 25: Parameter Type Selection

- Depending on the option type, the information to modify is slightly different. Moreover the generated settings parameter dialog box will be also different.
- Of course two applications with the same command line are refused.
- In one application, two options with the same command line are also refused.



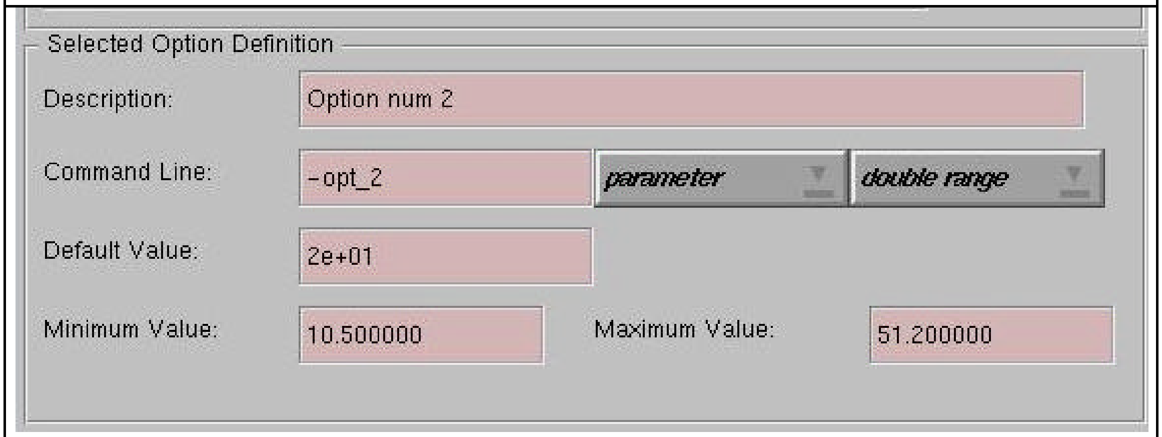
Selected Option Definition

Description: Option num 2

Command Line: -opt_2 | parameter | boolean

Default Value: ✓

Figure 26: Parameters to set when selected Boolean Option



Selected Option Definition

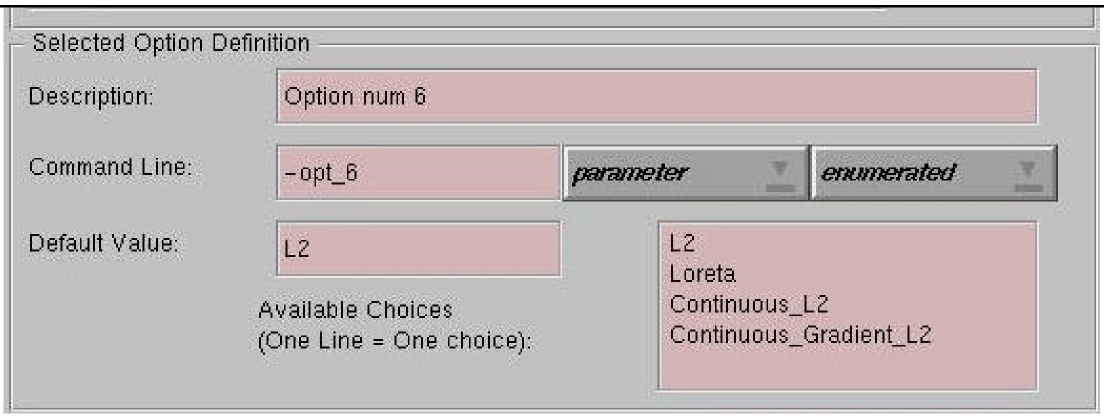
Description: Option num 2

Command Line: -opt_2 | parameter | double range

Default Value: 2e+01

Minimum Value: 10.500000 Maximum Value: 51.200000

Figure 27: Parameters to set when selected Double with Range option



Selected Option Definition

Description: Option num 6

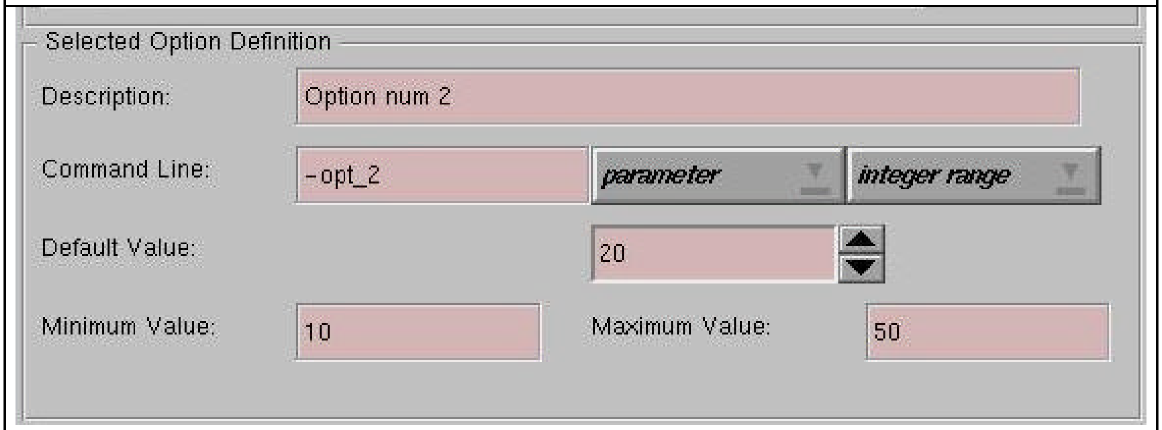
Command Line: -opt_6 | parameter | enumerated

Default Value: L2

Available Choices (One Line = One choice):

L2
Loreta
Continuous_L2
Continuous_Gradient_L2

Figure 28: Parameters to set when selected Enumerated option



Selected Option Definition

Description: Option num 2

Command Line: -opt_2 | parameter | integer range

Default Value: 20

Minimum Value: 10 Maximum Value: 50

Figure 29: Parameters to set when selected Integer with Range option

Adding new components

Introduction

To be really generic, users must be able to easily add new components inside the SimBio Scenario Editor.
To do such as thing, we propose to deal with this issue in the following way:

1. Adding new components by source code modification.
2. Reading components definition inside configuration file (XML format too).

For the moment, step 1 is completely available using SimBio editor sources codes.
The SimBio Scenario Editor contains a fixed number of components and we have defined a list of steps to do to add a new component. This ‘adding manual’ has already been tested successfully when adding last components (ESI and NEC remote at least).
Step 2 is also currently available but will probably needs some improvement in the next few months.

Modifying editor sources code

For the moment, to add new component by source code modification you have to follow the next 14 steps:
You can also have a look on source code and search for vgrid for instance to see how this component has been added.
Please use variables name, files name, functions name, …, which contain the name of the component you add to be able to find it again easily in the source code.

1. Create properties dialog box using QT designer. See examples in “designer/*.ui”. You can also created by hand but it will be more difficult.
2. Add references into Makefile and designer/Makefile.
3. Create a new simple icon for menus (see “application?.xpm” in “resources/”).
4. Add "#include "application?.xpm" (or the name you choose for your icon) in “SimbioWindows.cpp”.
5. Add menu button in “SimbioMainWindow::createSimulationMenu”.
6. Add toolbar button in “SimbioMainWindow::createSimulationToolbar”.
7. Add function “SimbioMainWindow::addApplication?” in “SimbioWindows.hpp”.
8. Add function “SimbioMainWindow::addApplication?” in “SimbioWindows.cpp”.
9. Add function “SimbioDocumentWindow::addApplication?” in “SimbioWindows.hpp”.
10. Add function “SimbioDocumentWindow::addApplication?” in “SimbioWindows.cpp”.
11. Add new application parameters in “SimbioSimulation::initializeKnownApplications”.
    Be careful, don't forget to update “NB_KNOWN_APPLICATIONS” in this function.
12. Add new type of application (EN_ApplicationKind) in “SimbioApplication.hpp”.
13. Add new case in “SimbioApplication::setType”.
14. Add new case in “SimbioApplication::updateDialogFromApplication”.
    Be careful, don't forget any widget.
15. Add new case in “SimbioApplication::updateApplicationFromDialog”.
    Be careful, don't forget any widget.

Using Scenario Editor preferences

See previous paragraph called Options Management / Add One Application.

WP6 Scenario Pilot – User’s Manual

Description

The SimBio-Pilot executes the SimBio requests. Normally, as mentioned above this program will be started automatically by the SimBio Editor, however, if the user has an appropriate XML file, then it can be started on the command line as: *simbio-pilot file.xml.*

Functionality

The SimBio-Pilot uses the Apache Xerces program to parse the XML request file. It then extracts the individual commands which are part of the complete request and executes them sequentially in the order defined by the XML request file. Once the request sequence is completed an-email message is sent to the user containing a status report on each individual request. In addition, the status messages are written to the log file specified by the user in the XML request file.

Installation

The user must first install all the applications from the other workpackages that he/she wants to use. Then they install the editor and the pilot. Then if the user wants to run jobs remotely, they need to install the CORBA package mentioned above and finally the executables that manage the communication to NEC's compute server.
The SimBio pilot distribution will contain only one executable file called simbio-pilot.

Status

Finished.

WP6 Remote Site - User’s Manual

General Overview

As already described above the SimBio partners agreed to restrict the SimBio environment to a simplified 2-way model. The first part of this model constitutes the local environment.
Here all non-simulation tools are installed and controlled by our SimBio Editor. The other part of this model is the so called remote or compute server site.
Many discussions have been held about the way local and remote tools should interact within medical environments (cf. D6a). Finally we agreed on using the middleware layer CORBA to handle the required data exchanges and authorization steps. Since the use of CORBA imposes a client/server logic onto our environment, we have to install server daemons on the compute server site, which permanently listen for incoming SimBio requests. Apart from very few exceptions, the vast majority of parallel systems are under the control of management programs. One central part of these suites is the job scheduler.
The SimBio server has to interact directly with the scheduler for starting simulation requests and receiving information about the current job status (queued, running, terminated) . Since the server has to do these actions in name of all legal SimBio users, it needs additional permissions for their execution.
However, it is very unlikely that external computing centers would allow the installation of such daemons. Furthermore, the interaction possibilities with classical batch queuing systems, e.g., NQS, are rather limited. Thus, it is only feasible to provide a comfortable and user friendly compute server environment on systems that are fully under our control.
The negative impact on SimBio partners is that it's only possible to use the NEC's PC-cluster for their simulation automatically. If they require other platforms instead, they will get no further support from the "generic environment". However, if partners would install compatible clusters, they could straightforwardly port our compute server related software to them.

Introduction

Depending on the local security environment under which a user is working, the simulation part consists of either one or two client server pairs.
The first client program is responsible for transferring to the target machine the description of the simulation, the so called request descriptor (RD), along with the corresponding mesh/grid file.
On the target machine a servant listens for matching incoming requests. It checks the validity of the RD and, if all security aspects are successfully matched, stores it in a database.
The status of the transmission, success or error value, is returned to the client program.
On those sites that allow for callbacks, an additional server program is launched in the local environment of the user. Its purpose is to process the incoming result data from the simulation and to store it in a certain directory.
The server on the simulation site launches an appropriate client for handling the transfer. If security restrictions on the user site prohibit the automatic exchange, users are in charge of ftp-ing the data on their own. Since big time differences may occur between the initiation of a request and it's completion, an e-mail will provide information on the termination status of the request to the user.

Implementation Design

Request Initiation

The Pilot invokes the client program after having successfully generated the mesh/grid file. It simply passes the name of the grid file and additional control values as command line parameters to the executable. The client tries first to transfer the RD to the server. Currently the RD consists of the following IDL entries:

```
struct SimReq_ {
      Account user;
      Account password;
      Path email;
      FileName mesh;
      Path workdir;
      Path targetdir;
      short time;
      short procs;
};
```

Listing 1: RD’s Idl Representation

The call might fail due to several reasons. On the one hand, the complete compute server service may not be active. Here, the client is unable to connect to the naming service or the actual SimBio server due to hardware/network problems.
Another possibility would be a rejection of the service due to invalid access permissions.
While in the first case the users have to postpone their actions, in the latter one they could check their user/password pairs and re-try or ask the SimBio service administrator for further information.

Request Storing

The server program running on a system in the parallel compute server environment enters some data from the RD into a relational database (currently MySQL). The delivered user and password entries reused to validate the permission for inserting into a table. Furthermore the values for requested time and procs are checked whether they are in allowed range. If all relevant parameters are matching the requirements, the actual insertion will be done. After the successful completion of this first transmission phase, the client starts to send the (large) binary file. The servant in charge receives this file chunk by chunk and stores it into the targetdir. If no problems occurred a special flag is updated in the corresponding line of the database table, indicating the completeness of the simulation request. At this point the client program terminates and the established network connection shuts down.

Request Execution

The job scheduling system is instrumented to check periodically the contents of the SimBio request database. If it detects a simulation request ready for execution and the required resource contingents are available, it will automatically schedule the request for a batch queue. Every user gets an e-mail acknowledgment at starting and termination time of the job. In those cases where automatic transfer of result data is possible, the scheduler will invoke the corresponding client program. The request will be taken out of the request table and shifted to an accounting table. The information stored here could be taken for various things, e.g., statistics on machine usage or preparation of invoices.

Security

For the data-exchange between user site and the compute server, we use the SSL plug-in from CORBA. This gives as currently a 128-bit encryption security. If user decide that the sensitivity of their data is low, e.g., test or junk data, encoding and decoding phases can be omitted for performance issues.

Status of Prototype

The first integrated simulation program is the HEAD-FEM FE code. It expects a mesh file, originally generated by the vgrid utility and later partitioned with the DRAMA tool, as input parameter. Both, the partitioning program and the simulation code run in parallel on the same number of processors, which allows to do just one resource allocation. A predefined batch script is passed to the scheduler that fetches the mesh file and invokes the DRAMA tool on it.
If this could be successfully completed, the actual simulation is carried out.

- ✓ HeadFem -- finished.
- ✓ NeuroFem -- almost finished.
- ✓ PAM -- not yet started.

Release Notes

SimBio Scenario Editor

- 17.04.2002: Version 2.0 fixes the following bugs and errors:
  1. Adding 5 new SimBio Applications inside SimBio Editor.
  2. Finishing possibility to add new component manually.
  3. Correcting several errors from partners remarks.

- 15.01.2002: Version 1.0 fixes the following bugs and errors:
  1. Settings correct parameters for each SimBio Applications inside SimBio Editor.
  2. Adding preferences management.
  3. Possibility to add new component manually.
  4. Correcting several errors from partners remarks.
      1. New Binary version made on Linux 2.2, RedHat 6.2

- 24.04.2001: Version 0.1.1 fixes the following bugs and errors:
  1. Adding new application called uif3, Inverse Toolbox application, with corresponding properties panel.
  2. ‘raw data’ used for testing was removed.
  3. ‘Memory Fault’ core dumped with ‘New’ menu in QT static version.
  4. ‘Memory Fault’ core dumped when inserting a vgrid application within an empty simulation.
  5. Call to SimBio Pilot is now done using environment variable called ‘SIMBIO_PILOT_PROG’ instead of calling directly a script. If this variable does not exist, default value 'simbio-pilot’ is called. The parameter is still the scenario XML file.

- 15.04.2001: Beta Version 0.1.0 is completed. The SimBio Scenario Editor incorporates the following features:
  1. It contains 4 applications (rawdata, vgrid, vsegment and drama-sim). Each application has its own properties panel that allows user to set application parameters.
  2. Following functions are currently working: New/Open/Save/Save As, Properties/Run, Close/Quit, Cut/Copy/Paste, Deselect/Properties and Insert application1 to 4. It is able to manage several scenarios at the same time and all windows manipulation (tile, cascade and so on) from QT are available.
  3. Following functions that are not currently implemented:
     - Print: nothing done for the moment.
     - Undo: not yet finished.
  4. Binary versions are available for the following operating systems:
      2. SGI IRIX 6.5 with QT Dynamic or QT static.
      3. SUN OS 5.5 has been tested once but have to be redone.

SimBio Pilot

- 25.04.2002: Version 0.2.3 fixes some bugs in the logfile output.
- 17.02.2002: Version 0.2.0 restructures the logfiles.
- 11.04.2001: Version 0.1.1 fixes a bug that can occur due to certain incompatibilities between versions 18-12-2000 and 24-10-2000 of the 'simbio-request.dtd' files.

- 01.04.2001: Beta Version 0.1.0 is completed. The SimBio-Pilot incorporates the following features:
  * Reads XML files covered by the 'simbio-request.dtd' version 18-12-2000 or 24-10-2000.
  * Executes the user's request in exactly the order given.
  * Creates a log file containing a list of all requests along with any error messages.
  * Notifies the user via e-mail when the entire request sequence has finished.
  * Binary versions are available for the following operating systems:
    4. SGI IRIX 6.5.
    5. Linux 2.2.

  (At the moment only dynamically linked versions are available for SGI systems.)

Future Planning (future releases)

SimBio Scenario Editor

First of all this editor will be distributed to partners involved in the project that they can evaluate it. Depending on their comments, the following list of priority could be modified.
Anyway here are future developments, in order of priority; we would like to emphasis on in the near future:

1) (high) Adding work package ST4.2 application, called uif3 that is not yet fully achieved.
2) (high) Improving automatic filenames for input and output parameters.
3) (low) Adding new components if asked by partners.
4) (medium) Changing application menus (toolbar) by using 16x16 'icons'. Needs to get icon for each component that is part of the SimBio Scenario Editor.
5) (low) Changing application drawing by using 100x100 'icons'. Needs to get icon for each component that is part of the SimBio Scenario Editor.

SimBio Pilot

In cooperation with the other project partners who will be evaluating the environment, we will consider their feedback and make any necessary changes accordingly. Furthermore, we will complete the integration of NeuroFem and PAM if there is sufficient interest from the other project partners.

“Remote compute server” Development

In the final year we will integrate the missing simulation applications into our "Remote compute server" environment.

Feedback

The WP6 Editor, pilot and “Remote compute server” are now almost finish; only few functionalities are missing.

For questions, comments and suggestions please contact:

Mr Christophe Janvrais ESI Software 99 rue des Solets SILIC 112 94513 Rungis Cedex Phone: +33 (0)1-41-73-58-00 Christophe.Janvrais@esi-group.com	Mr Falk Zimmermann C&C Research Laboratories NEC Europe Ltd. Rathausalle 10 53757 St. Augustin +49-2241-92520 falk@ccrl-nece.de

We are grateful for any feedback and interest.

Annex 1: Preferences DTD[1] file: 'simbio-editor-config.dtd'

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ............................................................... -->
<!-- SimBio Editor Config DTD ...................................... -->
<!-- ............................................................... -->
<!--
PURPOSE:
    This DTD was developed within the SimBio project.
    It is intended to define the Simbio Editor environment.
CHANGE HISTORY:
    The list of changes appears at the end of the DTD.
-->

<!-- A Simbio Editor Config consists of graphic options, default simulation
            parameters, pilot, list of used default applications and new application list.
    Additionally, each Simbio Editor Config request an ID which should be universially unique.
-->

<!ELEMENT SimBio-Editor-Config (graphic-options?,default-simulation?,pilot?,default-applications?,application-list?) >
<!ATTLIST SimBio-Editor-Config id   ID  #REQUIRED>

<!-- Graphical parameters used by Simbio Editor -->
<!ELEMENT graphic-options (background?,foreground?,frame?,filled-frame?,selected?) >

<!-- Background color used when displayed applications scenario tree -->
<!ELEMENT background (color) >
<!-- Foreground color used when displayed applications scenario tree -->
<!ELEMENT foreground (color) >
<!-- Frame color used when displayed applications scenario tree -->
<!ELEMENT frame (color) >
<!-- Frame parameters (width and rounded size) used when displayed applications scenario tree -->
<!ATTLIST frame
      width CDATA #REQUIRED
      xrnd CDATA #REQUIRED
      yrnd CDATA #REQUIRED>
<!-- Color used to fill frame when displayed applications scenario tree -->
<!ELEMENT filled-frame (color) >
<!-- Color used to display selected application -->
<!ELEMENT selected (color) >

<!-- Default simulation parameters used by Simbio Editor -->
<!ELEMENT default-simulation (name,working-dir?,log-file?,notify?,input?,output?) >

<!-- Default Name: (Simulation | option) -->
<!ELEMENT name (#PCDATA) >
<!-- Default Working Directory -->
<!ELEMENT working-dir (#PCDATA) >
<!-- Default Log File -->
<!ELEMENT log-file (#PCDATA) >
<!-- Default Notify Address -->
<!ELEMENT notify (#PCDATA) >
<!-- Default Input Filename -->
<!ELEMENT input (#PCDATA) >
<!-- Default Output Filename -->
<!ELEMENT output (#PCDATA) >

<!-- Where is located the simbio pilot used to run the scenario? -->
<!ELEMENT pilot (#PCDATA)>


<!-- Which default application do we want to use? -->
<!ELEMENT default-applications (#PCDATA) >
<!-- Do we use ipm?, -->
<!--    generis?,pamsafe?,pamview?, -->
```

[1] Document Type Definition

```
<!--    neurofem? -->
<!--    remote HeadFEM?, remote NeuroFEM? -->
<!--    vgrid?,vsegment3d?, -->
<!ATTLIST default-applications
        ipm (true|false) #IMPLIED
        generis (true|false) #IMPLIED
        pamsafe (true|false) #IMPLIED
        pamview (true|false) #IMPLIED
        neurofem (true|false) #IMPLIED
        remote-headfem (true|false) #IMPLIED
        remote-neurofem (true|false) #IMPLIED
        vgrid (true|false) #IMPLIED
        vsegment3d (true|false) #IMPLIED>

<!-- List of new applications to use -->
<!ELEMENT application-list (application*) >

<!-- New application definition -->
<!ELEMENT application (command,menu-icon?,display-icon?,description,stdin?,stdout?,stderr?,options-list*) >
<!-- Could be multiple -->
<!ATTLIST application
        name CDATA #REQUIRED
        multiple (true|false) #IMPLIED>

<!-- Application command line -->
<!ELEMENT command (#PCDATA) >
<!-- Application icon used in menu and toolbar -->
<!ELEMENT menu-icon (#PCDATA) >
<!-- Application icon used when display -->
<!ELEMENT display-icon (#PCDATA) >
<!-- Application short description -->
<!ELEMENT description (#PCDATA) >

<!-- List of new options to use -->
<!ELEMENT options-list (option*) >

<!-- Option definition -->
<!ELEMENT option (comments,range?,enumeration?,default-value?) >
<!-- Option type -->
<!ATTLIST option
        name CDATA #REQUIRED
        family (input|output|help|parameter) #REQUIRED
        type (string|file|directory|boolean|enumerated|double|integer) #REQUIRED>

<!-- Option short description -->
<!ELEMENT comments (#PCDATA) >
<!-- Option default value -->
<!ELEMENT default-value (#PCDATA) >
<!-- Option Range in case of integer or double -->
<!ELEMENT range (#PCDATA) >
<!ATTLIST range
                min CDATA #REQUIRED
                max CDATA  #REQUIRED>
<!-- Option Enumeration list in case of enumerated -->
<!ELEMENT enumeration (#PCDATA) >

<!-- Color definition  -->
<!ELEMENT color (#PCDATA) >
<!ATTLIST color
                red CDATA #REQUIRED
                blue CDATA #REQUIRED
                green CDATA #REQUIRED>

<!-- Standard input definition  -->
<!ELEMENT stdin (#PCDATA)>

<!-- Standard output definition  -->
<!ELEMENT stdout (#PCDATA)>
```

```
<!-- Standard error definition  -->
<!ELEMENT stderr (#PCDATA)>

<!-- ............................................................... -->
<!-- Change history ................................................ -->

<!--
#10-05-2001: janvrais (Christophe.Janvrais@esi-group.com)
# -  initial version
-->
```

Annex 2: XML Preferences Example File

```
<?xml version="1.0" encoding="UTF8"?>

<!DOCTYPE SimBio-Editor-Config SYSTEM "simbio-editor-config.dtd">
<SimBio-Editor-Config id="Config1">
        <graphic-options>
                <background>
                        <color red="0" green="0" blue="0"/>
                </background>
                <foreground>
                        <color red="255" green="0" blue="0"/>
                </foreground>
                <frame width="2" xrnd="15" yrnd="25">
                        <color red="0" green="255" blue="0"/>
                </frame>
                <filled-frame>
                        <color red="0" green="0" blue="255"/>
                </filled-frame>
                <selected>
                        <color red="255" green="255" blue="255"/>
                </selected>
        </graphic-options>
        <default-simulation>
                <name>NewSimulation</name>
                <working-dir>.</working-dir>
                <log-file>./simbio-req.log-R1</log-file>
                <notify>Christophe.Janvrais@esi-group.com</notify>
                <input>simufile.in</input>
                <output>simufile.out</output>
        </default-simulation>
        <pilot>simbio-pilot.csh</pilot>
        <default-applications ipm="true" vgrid="true" drama-sim="true" vsegment3d="false">
        </default-applications>
        <application-list>
                <application name="vgrid2" multiple="true">
                        <command>vgrid2</command>
                        <description>Very nice application without bugs to ...</description>
                        <stdin>stdin-mesh</stdin>
                        <stdout>stdout-mesh</stdout>
                        <stderr>stderr-mesh</stderr>
                        <options-list>
                                <option name="-fin" type="file" family="input">
                                        <comments>Input File Parameter</comments>
                                </option>
                                <option name="-fout" type="file" family="output">
                                        <comments>Output File Parameter</comments>
                                </option>
                                <option name="-str" type="string" family="parameter">
                                        <comments>String Parameter</comments>
                                        <default-value>toto</default-value>
                                </option>
                                <option name="-str2" type="string" family="parameter">
                                        <comments>String2 Parameter</comments>
                                        <default-value>toto2</default-value>
                                </option>
                                <option name="-in" type="file" family="input">
                                        <comments>File Parameter</comments>
                                        <default-value>ds2t_t1_a1.seg</default-value>
                                </option>
                                <option name="-vald" type="double" family="parameter">
                                        <comments>Double Parameter</comments>
                                        <default-value>44.0</default-value>
                                </option>
                                <option name="-valdr" type="double" family="parameter">
                                        <comments>Double Range Parameter</comments>
                                        <range max="9.0" min="2.0"/>
                                        <default-value>5.0</default-value>
                                </option>
                                <option name="-vali" type="integer" family="parameter">
                                        <comments>Integer Parameter</comments>
                                        <default-value>77</default-value>
                                </option>
                                <option name="-valir" type="integer" family="parameter">
                                        <comments>Integer Range Parameter</comments>
                                        <range max="100" min="10"/>
                                        <default-value>45</default-value>
                                </option>
                                <option name="-bool" type="boolean" family="parameter">
                                        <comments>Boolean Parameter</comments>
                                        <default-value>1</default-value>
                                </option>
                                <option name="-enum" type="enumerated" family="parameter">
                                        <comments>Enumerated Parameter</comments>
                                        <enumeration>titi,toto,tutu</enumeration>
                                        <default-value>toto</default-value>
                                </option>
                                <option name="-enum2" type="enumerated" family="parameter">
                                        <comments>Second Enumerated Parameter</comments>
                                        <enumeration>aaa,bbb,ccc</enumeration>
                                        <default-value>aaa</default-value>
                                </option>
                        </options-list>
                </application>
        </application-list>
</SimBio-Editor-Config>
```

Annex 3: XML Simulation Request Example File

This simulation request called ‘New_Simbio_Simulation‘ is made of following application: ipm, generis, pamsafe, pamview, vgrid and vsegment3d; each application contains an important number of parameters (called option).

```
<?xml version="1.0" encoding="UTF8"?>

<!DOCTYPE SimBio-Request SYSTEM "simbio-request.dtd">
<SimBio-Request id="New_Simbio_Simulation">
        <user-environment>
                <working-dir>.</working-dir>
                <log-file>./logfile_Empty_Simbio_Simulation</log-file>
                <notify>cja@esioct4</notify>
        </user-environment>
        <request-sequence>
                <request>
                        <command>ipm</command>
                        <command-line-options>
                                <option name="-in">simufile.input</option>
                                <option name="-out">output</option>
                                <option name="-ls">On_Cortex</option>
                                <option name="-lh">BEM</option>
                                <option name="-lis">L2</option>
                                <option name="-lit">Truncated_SVD</option>
                                <option name="-gs">On_Cortex</option>
                                <option name="-gh">BEM</option>
                                <option name="-ms">On_Cortex</option>
                                <option name="-mh">BEM</option>
                                <option name="-num">4</option>
                                <option name="-df">Moving</option>
                                <option name="-dh">BEM</option>
                                <option name="-di">Truncated_SVD</option>
                                <option name="-do">Simplex</option>
                                <option name="-dc">MSError</option>
                                <option name="-igs">0</option>
                                <option name="-sveb">0</option>
                                <option name="-itl1">0</option>
                                <option name="-nls">On_Cortex</option>
                                <option name="-nlh">BEM</option>
                        </command-line-options>
                </request>
                <request>
                        <command>generis</command>
                        <command-line-options>
                                <option name="-f">output</option>
                                <option name="-sol">psafe</option>
                        </command-line-options>
                </request>
                <request>
                        <command>pamsafe</command>
                        <command-line-options>
                                <option name="-input">./inputFile</option>
                                <option name="-logfile">./logfile</option>
                        </command-line-options>
                </request>
                <request>
                        <command>pamview</command>
                        <command-line-options>
                                <option name="-file">./inputFile</option>
                                <option name="-cmdf">./cmdfile</option>
                                <option name="-title">SIMBIO</option>
                        </command-line-options>
                </request>
                <request>
                        <command>vgrid</command>
                        <command-line-options>
                                <option name="-in">./inputFile</option>
                                <option name="-out">output</option>
                                <option name="-elem">cube</option>
                                <option name="-min">4</option>
                                <option name="-max">4</option>
                                <option name="-smooth">no</option>
                                <option name="-shift">0.49</option>
                                <option name="-surface">0</option>
                                <option name="-np">1</option>
                                <option name="-constraint">no</option>
                        </command-line-options>
                </request>
                <request>
                        <command>vsegment3d</command>
                        <command-line-options>
                                <option name="-in">output</option>
                                <option name="-out">simufile.output</option>
                                <option name="-nc">3</option>
                                <option name="-win">8</option>
                                <option name="-lim">35</option>
                                <option name="-lambda1">200000</option>
                                <option name="-lambda2">2.00E+06</option>
                        </command-line-options>
                </request>
            </request-sequence>
        </SimBio-Request>
```

Annex 4: Corresponding DTD file: 'simbio-request.dtd'

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ............................................................... -->
<!-- SimBio Request DTD ........................................... -->
<!-- ............................................................... -->


<!--
PURPOSE:
    This DTD was developed within the SimBio project.  It is intended
    to define a user request on the environment.

CHANGE HISTORY:
    The list of changes appears at the end of the DTD.
-->

<!-- A SimBio request consists of a "user-environment" element and
    a "request-sequence" element. Additionally, each SimBio request
    an ID which should be universially unique.
-->

<!ELEMENT SimBio-Request (user-environment,request-sequence) >

<!ATTLIST SimBio-Request id   ID  #REQUIRED>

<!-- The "user-environment" consists of a working directory, a log file,
    possibly stderr and stdout files and a an e-mail address of the
    user to be notified when the request has completed.
-->

<!ELEMENT user-environment (working-dir,log-file,notify?) >
<!ELEMENT working-dir (#PCDATA) >
<!ELEMENT log-file (#PCDATA) >
<!ELEMENT notify (#PCDATA) >

<!-- The request sequence consists of one or more requests that are to be
    executed in the order in which they appear in the file.
-->

<!ELEMENT request-sequence (request+) >

<!-- Each request contains all the information needed to run the
    specified command.
-->


<!ELEMENT request (command,stdin?,stdout?,stderr?,command-line-options*) >

<!ELEMENT command (#PCDATA) >

<!ELEMENT command-line-options (option*) >

<!ELEMENT option (#PCDATA) >
<!ATTLIST option
            name   CDATA  #REQUIRED
            type   CDATA  #IMPLIED
            separator (none|space) "space">


<!ELEMENT stdin (#PCDATA)>
<!ELEMENT stdout (#PCDATA)>
<!ELEMENT stderr (#PCDATA)>

<!-- ............................................................... -->
<!-- Change history ............................................... -->
```

```
<!--
#25-04-2002: kohring
#		Made the notify element optional.
#05-03-2002: kohring
#		Corrected a spelling error in the option ATTLIST.
#21-02-2002: kohring
#		Added the "separator" attribute to the option element to allow a choice
#		of separators between the option name and the option value when placed
#    on the command line.
#18-12-2000: kohring
#		Ammended the "stdin, stdout and stderr" definitions to be
#		consistent with the definition for "logfile"
#07-12-2000: janvrais
#    Replaced 'request*' with 'request+' in 'request-sequence' definition.
#    This prohibits one from generating XML files without a request.
#24-10-2000: kohring
# -  initial version
-->
```