An Open Domain-Extensible Environment for Simulation-Based Scientific Investigation (ODESSI)

Adnan M. Salman, Allen D. Malony, and Matthew J. Sottile

Dept. Computer and Information Science, University of Oregon, Eugene, OR 97403 USA
{adnan,malony,matt}@cs.uoregon.edu

Abstract. In scientific domains where discovery is driven by simulation modeling there are found common methodologies and procedures applied for scientific investigation. ODESSI (Open Domain-extensible Environment for Simulation-based Scientific Investigation) is an environment to facilitate the representation and automatic conduction of scientific studies by capturing common methods for experimentation, analysis, and evaluation used in simulation science. Specific methods ODESSI will support include *parameter studies*, *optimization*, *uncertainty quantification*, and *sensitivity analysis*. By making these methods accessible in a programmable framework, ODESSI can be used to capture and run domain-specific investigations. ODESSI is demonstrated for a problem in the neuroscience domain involving computational modeling of human head electromagnetics for conductivity analysis and source localization.

1 Introduction

Computational science is now accepted as an important approach for scientific investigation, broadly considered equivalent in its discovery power to theoretical and experimental science. It is typically conducted through mathematical modeling and scientific simulation, leveraging access to advanced, high-performance computers (HPC) to run computational experiments (simulations) that seek to model reality in various domains. The evolution of computational science reflects both a growing need for computational power and increased sophistication of simulation methodology. Early concerns were on access to sufficient HPC resources, motivating research in parallel computing, computational grids, and large-scale storage. More recent research work in computational portals and workflows attempts to simplify resource access as well as provide programming support for coordinating simulation and analysis tasks. With computational horsepower becoming more ubiquitous, there is now growing interest in enhancing the discovery process of scientific investigations. In general, how productivity in computational-based science can be improved in practice will depend greatly on software environments that raise the level of investigation creation, execution, and management.

In scientific domains where discovery is driven by simulation there are common methodologies and procedures. An environment that can capture the shared standard practices and support their reuse across domains could improve productivity in scientific investigation creation and application. Methods such as parameter studies and tuning, optimization, uncertainty and sensitivity analysis, are generally used across simulation fields. Application of these methods in simulation studies typically require executing the

G. Allen et al. (Eds.): ICCS 2009, Part I, LNCS 5544, pp. 23–32, 2009.

simulation many times with different input parameter sets and data files. The environment could capture the common scientific methods in modules that can be contextualized for domain-specific use. The modules would hide the details of backend execution (implemented by the environment infrastructure), while providing an interface for their programming as part of an investigation workflow. The environment could also support other aspects of scientific investigations, including the management of input and output data, the specification of parameters, the post-processsing of results, and the generation of reports. The benefit is to provide a high level of service and automation to the computational scientist to enhance their work throughput and management.

In this paper we describe our research work to create and apply an environment for supporting scientific investigation called ODESSI (Open Domain-extensible Environment for Simulation-based Scientific Investigation, pronounced "Odyssey"). The environment will facilitate the representation and automation of scientific studies by capturing shared methods for experimentation, analysis, and evaluation used in simulation science in a framework that can be programmed and specialized for domain investigations. ODESSI will be demonstrated for scientific studies in the neuroscience domain involving computational modeling of the human head.

Section §3 describes the ODESSI objectives and design. The development of the ODESSI prototype is discussed in §4. ODESSI was inspired by our prior ICCS work [18,20] on computational modeling of human head conductivity. Section §5 outlines the domain problem in human brain science we are investigating and shows how ODESSI is applied to improve scientific productivity in this domain. Section §6 concludes with a discussion of research issues and outline of future research directions.

2 Related Work

The general theme of the ODESSI approach is to manage complexity in domain-specific scientific investigations by providing a programmable framework with high-level services for domain contextualization and use. Problem solving environments (PSE) are a traditional approach to addressing domain-relevant concerns by incorporating all the mathematical, algorithmic, and computational features necessary to solve a targeted class of science or engineering (S/E) problems [1,2]. The main goal of a PSE is to increase the productivity of scientists by letting them describe a problem and its solution in terms of the S/E concepts and use a highly-functional, integrated set of capabilities for modeling, analysis, and visualization. PSEs have been developed for partial differential equations (PDE) [3], linear algebra [4], chemistry [5], and other S/E areas. However, the traditional PSE approach has three important drawbacks: 1) it is difficult to create a new PSE, 2) PSEs are not developed to be reused, and 3) PSEs are hard to extend with new capabilities or new science methods.

One response to strict PSE design is to identify domain-level functionality that is common across related fields and build software tools that can be applied in developing computational science environments [6]. Scientific development environments take this idea further by offering rich components for data management and analysis, in a programming framework for scientific applications. For example, SCIRun [7] is a powerful environment for interactive computational science which has been used to create

integrated problem solving environments in biomedical science [8]. ODESSI complements these directions by abstracting common simulation-based scientific methods in reusable components, providing a cross-domain framework for scientific investigation.

Grid computing and workflow systems research take a different tact by focusing on how to allocate and coordinate the use of computational resources (both systems and software/tool components) to create and run scientific applications such as GridLab[15]. Grid-enabled workflow systems such as Pegasus[9], Triana [10], and Kepler [11] are powerful tools being applied in computational science projects. However, their support for multi-experiment simulation workflows is still rudimentary and is not easily programmed for cross-domain use or execution on non-grid platforms. Web-based portals (e.g., the NEES [12] and BIRN [13] portals) and environments such as ViroLab [16] address some of these issues by offering higher-level S/E services (e.g., analysis, data management, simulation) while hiding backend complexity. The ability to abstract and reapply scientific methods for new scientific investigations or new scientific domains in these environments though is not supported well.

On the other hand, there are wealth of toolkits for scientific methods used in simulation. The DAKOTA toolkit [14] provides several optimization algorithms, uncertainty quantification, and parameter estimation. The Portable, Extensible Toolkit for Scientific Computation (PETSc) [17] is a suite of data structures and routines for the scalable (parallel) PDE-based scientific applications. The important aspect of these systems is their embodiment of a known scientific methodology in a programmable form. The idea behind ODESSI's approach is to provide a high-level scientific development framework that parameterizes and configures scientific methods for domain specialization.

3 ODESSI Requirements and Design

The goal of ODESSI is to provide a productive environment that assists domain scientists in the development and application of their computational investigations. To this end, the main requirements are:

1. To support common types of scientific methods used in simulation-based science.
2. To provide a programmable framework to contextualize methods for domain use.
3. To insulate the scientist from concerns of HPC resource usage, allowing them to focus on the process aspects of the domain investigation.
4. To provide record of simulation experiment for evolving scientific investigations.

The ODESSI environment shown in Fig. 1 was designed to support these requirements. The key concept of the ODESSI approach is the capture of standard procedures to conduct and analyze (simulation-based) scientific experiments in a modular, extensible, and reusable form. We call these procedures *scientific methods* and think of the methods as generating a set of simulation experiments to run. Common scientific methods include parameter studies, comparative analysis, optimization, sensitivity analysis, and uncertainty analysis. These methods are the basis upon which activities such as verification and validation, parameter tuning, and simulation-based experimentation are built for domain application. These processes that integrate different methods are the foundation of domain scientific investigations. A *scientific investigation* is a domain-specific

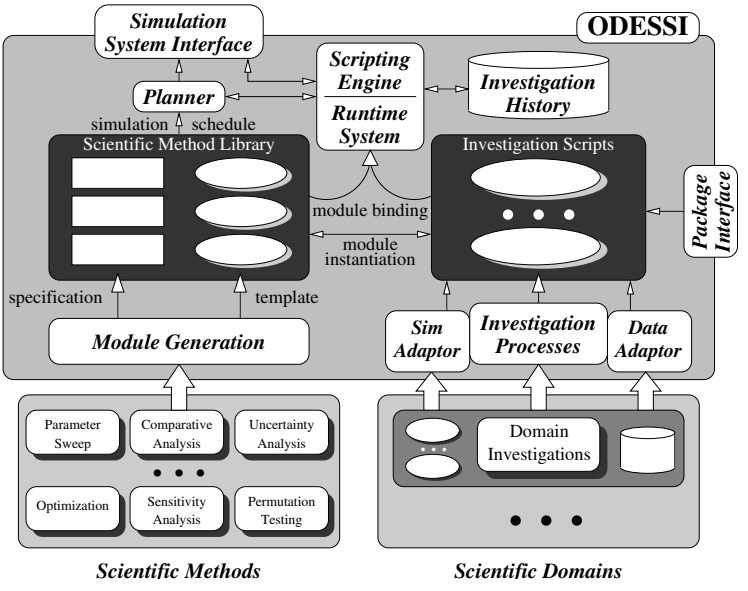

Fig. 1. Architecture and components of ODESSI framework

discovery process that applies one or more scientific methods in its lifetime. It defines the simulation codes to use, the input data files, and post-simulation analysis and visualization. If ODESSI can capture key scientific methods in easy-to-use modules, the level of productivity in the development and execution of scientific investigations may increase. We will focus our discussion on this aspect of the design.

Logically, ODESSI represents methods internally as *modules* consisting of two parts: a *specification* and a *template*. The specification identifies the context necessary for the execution of the modules, including the simulation program to be run and parameters. The template is the software construction of the module with abstract classes for operation of the specific scientific method. In this respect, the template embodies the method procedures for the generation of domain simulation experiments. A module is instantiated by an investigation script, setting the specification context and initializing the module state. When a method module is executed, it generates an experiment schedule that is passed to the ODESSI planner.

It is the responsibility of the ODESSI *planner* to conduct the necessary simulations on behalf of the invoked method. It is possible multiple methods are concurrently active, each with its own planner. The planner interfaces with the external simulation system to run a simulation experiment. It determines which experiments to execute based on the specified simulation schedule. If a method uses information from earlier experiments to determine future experiments, its module uses a dynamic schedule which is applied within the planner. The planner attempts to optimize schedules by interrogating the ODESSI *investigation history* to determine when simulation experiments have previously been conducted. A record is maintained in the ODESSI investigation history of every completed simulation experiment, containing complete metadata for the investigation and method specification.

4 ODESSI Development

In this section we describe the implementation of the conceptual design described in section §3. ODESSI is based on a set of Python objects that implement components shown in Fig. 1. These objects form a set of interacting threads that cooperate during the execution of the investigation script. An investigation script imports and instantiates the necessary objects that implement scientific methods that it uses, and these in turn interact through message passing operations once started. Simulation programs are invoked through the Python system interface.

An investigation script consists of three main sections. In the first section the user specifies the simulations under investigation and their initial input parameter values. The simulation specifications are used to create a simulation manager object that can be used to request simulation solutions. In the second section, the scientific investigation methods are customized, instantiated and launched for execution. Each scientific investigation method is executed in a separate thread. Each scientific method object derives from a base *Planner* object, adding the method-specific functionality that it is intended to provide. The third section is concerned with post-processing the results.

The ODESSI implementation uses messaging to communicate between threads. Due to the potential for long execution times during simulation runs, it is preferable to use an asynchronous execution model to allow threads to execute independently without blocking. Threads in ODESSI are based on the process and messaging model from the Erlang language [19] implemented by the Python "candygram" package. Each process has a message mailbox into which messages from other processes are delivered and later extracted and handled by the receiving process in the order of delivery.

ODESSI provides two entities that can be instantiated in the investigation script: the *Simulation Manger* entity and the *Scientific Investigation Method* entity. The Simulation Manager controls and manages the execution of a simulation. It acts as a server that provides solutions given sets of input parameters. Each instance of a simulation manger controls a single simulation. Multiple simulations can be controlled by multiple instances of the simulation manager. A simulation manager object can serve multiple requests from different threads. Scientific Investigation Methods provide the procedures that occur across scientific domains such as optimization and sensitivity analysis.

4.1 Scientific Investigation Methods

Scientific investigation methods are the common procedures that are used in several scientific domains. ODESSI currently supports optimization based on a parallel simulated annealing algorithm, simplex search, parameter studies and linear regression based sensitivity analysis. ODESSI can easily be extended with more methods and procedures. A scientific method gets executed in its own thread spawned by the main thread The scientific method is implemented as a module of four classes described below:

- A *specification* class providing a template to customize the investigation procedure.
- An *interface* class providing an interface for the user to instantiate and execute the scientific method. The interface class is parameterized with an instance of the method specification class and a simulation manager object. An interface object provides methods to start and stop execution and access results.

- The *scientific method logic* class implements the algorithm or the procedure of the scientific method and must inherit from the planner base class.
- The *planner* abstract class that all scientific method logic classes must inherit from. This class defines the interactions between the scientific method and the execution manager, cleanly separating scientific method logic from communication code.

4.2 Simulation Manager

The simulation manager class is the user interface to the execution manager class. A simulation manager object is created in the main thread.A simulation object and other optional parameters are passed in at initialization. At instantiation, the simulation manager spawns a thread and starts the execution manager. The simulation manager class currently provides methods to control execution and measure execution timing.

- *The execution manager:* The execution manager manages and controls a single simulation. It acts as a server that provides the simulation response given input parameters. The execution manager is an internal class and is not modified by the user. It is instantiated and run in its own thread by the simulation manager, and the planner base class requests solutions from the execution manager. Threads request solutions from the execution manager by delivering messages to its mailbox. In handling the request the simulation manager employs a dynamically sized pool of workers for simulation execution and a solution database manager.
- *Workers:* Each worker in the pool of an execution manager corresponds to the execution of a simulation on a resource. Workers interact with a simulation either through a very simple modification to the main function of the simulation code or through a wrapper around the unmodified simulation. The communication protocol between ODESSI workers and the simulation is very simple. Messages are defined for starting, stopping, parameterizing, and retrieving results from simulations.
- *The database manager:* The database manager implements the investigation history component of ODESSI and manages a repository where simulation solutions can be obtained. Solutions are associated both with the simulation code which produced them and the input parameters necessary for the run. This allows for both provenance tracking of simulation results and performance enhancement by avoiding redundant computations when the output already exists in the database.

5 ODESSI Application

ODESSI was inspired by our research in human neuroscience where we are developing computational models of human head electromagnetics for use in dynamic brain analysis [18,20,21]. The main goal of our research is to estimate the locations of the active brain regions given measured electroencephalogram (EEG) recordings. Called the *source localization* problem, its accurate solution will provide an opportunity to analyze cortex dynamics at high temporal and spatial resolution. The source localization problem has two parts:

1. *Forward problem*: given electrical sources (e.g., cortex dipoles), tissue geometries and conductivities, determine head volume and scalp electrical potentials.
2. *Inverse problem*: given an accurate forward solution, find optimal sources to match measured scalp potentials.

An accurate forward solver requires knowledge of the head tissues geometry (obtained from MR or CT images) and their conductivities. To determine the conductivities of the head tissues, we must solve the conductivity inverse problem. Here, a small current is injected in a subject's head and the response is measured on the scalp using electrical impedance tomography (EIT) technology. A search for optimal conductivities parameters can then be performed using the forward simulation compared to the measured potentials. Once the conductivities are found for an individual, a distributed dipole linear inverse solver can be built for EEG localization [22].

There are several challenges in this research. From the start, the source localization problem is ill-posed, since EEG measurements are made on (up to) 256 sensors and there may be thousands of cortex dipoles active. In addition, there are multiple sources of measurement error and modeling uncertainties that ultimately contribute to the accuracy of the solution as well as the performance. Measurement errors include the quality of MR/CT images, electrode and dipole registration, injected current level, and the EEG electrode data. These errors lead to modeling inaccuracies which propagate uncertainty in the solution results and also can affect computational efficiency. These include discretizing the PDEs, adjusting the computational grid resolution, and accurately segmenting the head tissues. Further, selection of parameters and modeling algorithm in the forward and inverse solvers also influence the final result.

How can we understand the quality of our source localization solutions and their use in dynamic brain analysis when dealing with multiple sources of measurement error and modeling uncertainties in constructing head models? Our desired scientific investigations involve computational processing to generate candidate models, as well as verification and validation to determine the effects of uncertainty and the robustness of solutions. In general, these objectives are shared with other scientific domains. In the following we outline the use of ODESSI in conducting several analyses from our domain, in particular showing the results from sensitivity analysis studies.

Conductivity Modeling. In our previous work we developed an inverse solver based on parallel simulated annealing algorithm to estimate the head tissue conductivities by solving the conductivity inverse problem. With ODESSI we were able to set up the problem and adjust the optimization parameters by only interfacing with the optimization method module. ODESSI made it trivial to experiment with different optimization objective function and different optimization algorithms.

PDF Solver Tuning. Our forward model is based on solving the time-dependent Poisson equation and considering the steady state solution as the static solution. The convergence of the forward solver depends, on two parameters. The *time step*, which controls the speed of reaching the steady state, and the convergence *tolerance* which specifies the level of accuracy. We used ODESSI to tune these parameters by performing a parametric study for different current injection pairs and different sets of conductivities.

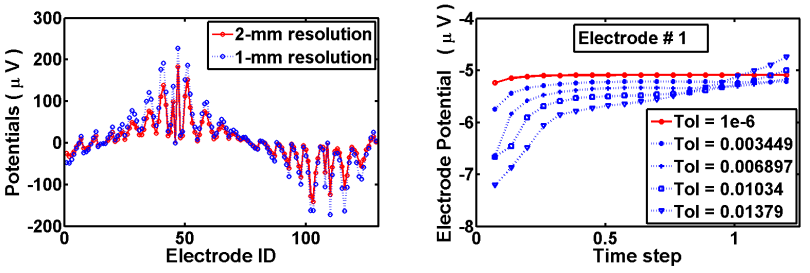

Fig. 2. Left, the potentials at the electrodes using $1mm$ resolution geometry vs. $2mm$ resolution. Right, tuning the forward-solver convergence parameters (time-step vs. tolerance).

Figure 2 shows a sample from this study. As the tolerance increases the solution fails to converge. For very small time step, the solver terminates prematurely.

Geometry Resolution Error. The geometry of the head tissues is obtained from imaging such as MRI or CT scans. Geometry obtained from high resolution ($1mm$) MRI captures more details about the head tissues, such as wholes in the skull. However, the computational time is significant. We use a high resolution image to construct lower resolution geometry by eliminating every other plane from the high resolution image. Then we used ODESSI to evaluate the error caused by this approximation. RDM and MAG metrics are used to compare between the solutions obtained using the two geometries. Our results show that the average RDM is about .8 and the average MAG is about .1. Therefore, the $2mm$ geometry can be used for visualization and testing. However, we have to use the high $1mm$ resolution to obtain accurate conductivity reconstruction. Here ODESSI allows us to experiment with the metrics for comparison.

Sensitivity Analysis. We further applied ODESSI for regression analysis to study how the uncertainty in each electrode potential can be apportioned to uncertainties in the inputs. In this analysis, we only considered the head tissue conductivities. A multivariate sample of 1000 points of the head tissue conductivities is generated. The conductivity of each head tissue is sampled from the normal distribution with mean equal to the average accepted value from the literature and the standard deviation is chosen such that the distance between the mean conductivity and the lower and upper bounds is about 3 standard deviations. Then the potentials at the electrodes are computed for each sample and a multiple regression fitting is performed on the standardized conductivities and electrode potentials. The standardized regression coefficients (SRC or β) quantify the effect caused by changes in the model independent variables from their mean values in terms of standard deviations.

Figure 3 shows distributions of the electrodes sensitivity due to changes in tissue conductivities. Positive β coefficients (SRC) correspond to electrodes near the current source while negative β coefficients correspond to electrodes near the sink. From the distributions we see that the potentials at all electrodes are insensitive to variation of the CSF tissue. This can be reasoned to the fact that the CSF tissue size is small and the variation in its conductivity is small. The second important observation is that the potentials are sensitive to changes of the brain conductivity. This observation contradicts

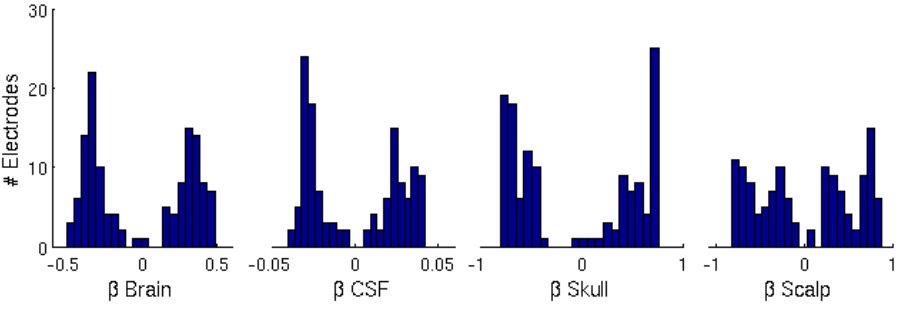

Fig. 3. Distribution of the electrode's sensitivity due to changes in tissue conductivities

the belief that most of the current will be shunted in the scalp. Therefore, we believe it is possible to explore the brain with EIT technology. We explain this sensitivity due to the fact that the brain is a big tissue and the wholes in the skull –which our forward solver captures– allow the current to go through the skull (contrary to spherical models). The third observation is that the potentials are highly sensitive to changes in the skull and scalp conductivities as expected, since the current sources are on the scalp.

Identifying and ranking the sensitivity of the electrodes due to model input variables are very important in our research. For instance, the conductivity of the CSF tissue can be considered homogeneous and be fixed at the literature accepted value. Also, the contributions of the electrodes potentials in computing the objective function can be weighted based on their sensitivity in the conductivity inverse problem.

Without the ODESSI framework, this detailed sensitivity analysis would not have been possible. We parameterized the sensitivity testing template in ODESSI and interfaced the simulation code. Once configured, the investigation required thousands of simulations to generate the results. These simulations were fully automated by ODESSI.

6 Conclusion and Future Work

ODESSI provides a framework for constructing scientific investigations by instantiating common methods for simulation-based analysis with domain-specific context. From a productivity perspective, ODESSI enables a (potentially large) number of simulations generated from a domain investigation processes to be run in a systematic and automated way. Extending our earlier ICCS research to allow investigation of uncertainty in brain conductivity and source modeling absolutely depended on this capability. The sensitivity results presented demonstrate the investigative power that can be achieved with relatively simple ODESSI programming.

However, the ODESSI concept extends further to domain support for data management (e.g., EEG and MRI data), results processing (e.g., statistics, data mining), visualization (e.g., plotting, 3D graphics), and meta-analysis. The key idea is how ODESSI can capture domain information to parameterize and contextualize common methods in these areas for high-level use. Our immediate goal is the development of the investigation history database to track the provenance of simulation experiments.

Acknowledgement

This research is supported by the Department of Defense, Telemedicine Advanced Technology Research Center (TATRC), grant number W81XWH-07-2-0092.

References

1. Gallopoulos, S., Houstis, E., Rice, J.: Computer as Thinker/Doer: Problem-solving Environments for Computational Science. IEEE Comp. Sci. and Eng. 1(2), 11–23 (1994)
2. Rice, J., Boisvert, R.: From Scientific Software Libraries to Problem-solving Environments. IEEE Computational Science and Engineering 3(3), 44–53 (1996)
3. Akers, R., Kant, E., Randall, C., Steinberg, S., Young, R.: Scinapse: A Problem-solving Environment for Partial Differential Equations. IEEE Comp. Sci. and Eng. 4(3) (1997)
4. Petitet, A., Casanova, H., Whaley, R., Dongarra, J., Robert, Y.: A Numerical Linear Algebra Problem-solving Environment Designer's Perspective. In: SIAM Annual Meeting (1999)
5. Extensible Computational Chemistry Environment, `http://ecce.emsl.pnl.gov/`
6. Cuny, J., et al.: Building Domain-Specific Environments for Computational Science: A Case Study in Seismic Tomg. Int. J. of Super. App. and High Speed Comp. 11(3) (1997)
7. Parker, S., Johnson, C.: SCIRun: A Scientific Programming Environment for Computational Steering. ACM/IEEE Conference on Supercomputing (1995)
8. MacLeod, R., et al.: SCIRun/BioPSE: Integrated Problem Solving Environment for Bioelectric Field Problems and Visualization. IEEE Int. Symp. on Biom. Img. 1, 640–643 (2004)
9. Deelman, E., et al.: Pegasus: Mapping scientific Workflows onto the Grid. In: Dubitzky, W., Schuster, A., Sloot, P.M.A., Schröder, M., Romberg, M., et al. (eds.) GCCB 2006. LNCS (LNBI), vol. 4360, pp. 11–20. Springer, Heidelberg (2007)
10. Churches, D., et al.: Programming Scientific and Distributed Workflow with Triana Services. Concurrency and Computation: Practice and Experience 18(10), 1021–1037 (2006)
11. Ludäscher, B., et al.: Scientific Workflow Management and the Kepler System. Concurrency and Computation: Practice and Experience 18(10), 1039–1065 (2006)
12. NEES Cyberinfrastructure Center, `http://it.nees.org/`
13. BIRN - Biomedical Informatics Research Network, `http://www.nbirn.net/`
14. The DAKOTA Project, `http://www.cs.sandia.gov/DAKOTA/software.html`
15. GridLab, `http://www.gridlab.org/`
16. ViroLab, `http://www.virolab.org/`
17. PETSc: Portable, Extensible Toolkit for Scientific Computation, `http://www.mcs.anl.gov/`
18. Salman, A., Turovets, S., Malony, A., Tucker, D.: Computational Modeling of Human Head Conductivity. In: Sunderam, V.S., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2005. LNCS, vol. 3514, pp. 631–638. Springer, Heidelberg (2005)
19. Armstrong, J.: Programming Erlang: Software for a Concurrent World (2007)
20. Salman, A., Malony, A., Turovets, S., Tucker, D.: Use of Parallel Simulated Annealing for Computational Modeling of Human Head Conductivity. In: ICCS 2007, pp. 86–93 (2007)
21. Salman, A., Malony, A., Turovets, S., Tucker, D.: On the Role of Skull Parcellation in the Computational Modeling of Human Head Conductivity. EIT, pp. 16–18 (June 2008)
22. Pascual-Marqui, R.: Review of Methods for Solving the EEG Inverse Problem. Int'l. Journal Bioelectromagnetics 1, 75–86 (1999)