A Grid-Enabled PSE for Physical Simulation: Experiences on GridPSi

Zhou Jun
Graduate School of Science and Engineering, Shizuoka University, Japan
zhoujun@cs.inf.shizuoka.ac.jp

Yukio Umetani
Computer Science Department, Faculty of Information, Shizuoka University, Japan
umetani@cs.inf.shizuoka.ac.jp

Abstract

As the power of computational Grids increases, there is a corresponding need for better usability in PSE community. In this paper, we describe our experiences on design of project GridPSi, a Grid-enabled PSE for physical simulation. GridPSi provides a rich suite of functions that facilitate the composition of computational scripts to automate physical simulations by design domain-specific templates for a verity of research fields. The generic functionality provided by it includes problem modeling, mesh file conversion, computational script generation, solution scheme specification and access to Grid resources. GridPSi offers a simple interface that is intuitive to users in the human-PSE interactions and ways of thinking. Furthermore, we developed higher-level graphical components named PDE wizards that build upon the core functions to guide users through problem modeling process with minimum effort. By using Unicore as Grid middleware, we are lowering the barriers to entry for users wishing to exploit Grid technologies.

1. Introduction

Grid computing environments have increasingly gained attention in recent years. Advances in Grid technologies as well as a better awareness of the needs of computational researchers motivate the development of Grid-enabled problem-solving environment, which will become increasingly important as a way of doing science. Although Grid Computing [1] is not a very new concept nowadays, it still presents a lot of challenges for the researchers in physical simulation filed who wish to use it to conduct their PDE-based sophisticated simulations. In general, these researchers cannot be expected to keep abreast of the rapid development of Grid technologies, so an easy-to-use environment is needed to allow them to take full advantage of the power of today's computational Grids without having detailed knowledge of the underlying Grid infrastructure.

One of the technical issues that must be addressed in order to realize the potential offered by the Grid in PSE research field is the ease with which it can be exploited by the end-user. Ideally, from the perspective for the researcher, the adoption of Grid technologies would not mean abandoning the tools, computing environments, or ways of both thinking and working that represent their daily practice. It is important to understand that integration with existing scientific computing environments is essential to promote the uptake of the Grid technologies and usage of PSEs. Another issue is the clear work assignment distribution between the end-user environment and target global resources. Undoubtedly, the current Internet is not ideal for remote interactive physical simulations since huge volume of data transmission will be involved during the interactions. We envision a future Internet, as advances in hardware and network bandwidth will make the real-time simulation become realistic, but in order for an efficient communication on current Internet, PSE developers have to carefully think about what part of work should be finished locally while what part should be delivered to the remote solving systems for execution when designing a Grid-enabled problem-solving environment.

The GridPSi project is a research program with the purpose of demonstrating a high-level design framework of Grid-enabled PSE for physical simulation field. GridPSi is a client-server based framework that provides non-computer professionals a natural environment in which they can utilize global high performance computing resources to rapidly prototype their research ideas in an intuitive and straightforward way.

The reminder of this paper is organized as following. In Section 2, we present an overview of the GridPSi system, with descriptions of system main features as well as our design and implementation considerations. Section 3 shows the GridPSi system architecture. Section 4 describes our practice on

interfacing GridPSi to the computational Grids. In Section 5, we use a simple case study to demonstrate running a physical simulation by using GridPSi. We make brief conclusions and discuss the future work in Section 6.

2. System overview

The GridPSi is designed to provide researchers in physical simulation field an easy-to-use environment for problem modeling, applying numerical schemes and visualizing simulation results as well as access to Grid computing resources. From an end-user's point of view, the GridPSi is a client-server based system.

2.1. Problem-solving environment

We agree with the definition of PSE given in [2]: "A PSE is a computer system that provides all the computational facilities necessary to solve a target class of problems. These features include advanced solution methods, automatic or semiautomatic selection of solution methods, and ways to easily incorporate novel solution methods. Moreover, PSEs use the language of the target class of problems, so users can run them without specialized knowledge of the underlying computer hardware or software."

For our research focus we assume the end-users of GridPSi are often computer literate but not trained computer experts. They are specialists in their own research fields and have the need to access computational Grids to solve their problems that require high performance computing resources.

2.2.Requirements for GridPSi

There are many well-known design principles [3] that PSE developers agree. By adhering to these principles, the following characteristics greatly influenced design of our system.

Problem domain-specific. The PSE should allow researchers to concentrate on their discipline without to care about too much computer science issues. It also needs to eliminate any superfluous information that might confuse a user in simulating process, and make a given physical simulation more domain-specific from the beginning.

Graphical and visual. The use of graphics and visuals can enhance the usability of the PSE. The graphical user interface of PSE should not only assist users in performing simulations in a straightforward way but also prevent them from "making errors".

Open and adaptive. The PSE must be able to tailor interchangeable, plug-and-play components and legacy code into use easily, and integrate itself into state-of-the-art software packages in a novel way.

Integrated and consistent. Many problems and their solution strategies are extremely heterogeneous: in models, methods and machines. The PSE must be designed to manage these issues in an integrated way, so that the user is presented with a predictable and consistent PSE [4].

2.3.System main features

The GridPSi is a new generation (in progress) of problem-solving environment for physical simulation that comes with many features of a modern software application.

Intuitive simulating environment. GridPSi client brings in a rich user-friendly GUI that guides a user through simulating process in a natural and straightforward way. It automates the processes of building simulation model and specifying solution details with a few mouse clicks and keyboard inputs. Besides, a user does not need to explicitly construct any matrices nor vectors in GridPSi system. What he needs to do is to input necessary experimental parameters, select simulating condition types by feeding only the right-side values or variables to equations predefined in abstracted forms.

The realization of such a simulating environment is mostly contributed to a family of high-level abstracted, domain-specific graphical components named PDE wizards, which are in turn, built on the core functions of the GridPSi client. In general, the client package is implemented in a three-layered architecture with the client GUI on the top, PDE wizard family and other functional packages in the middle, and core functions at the bottom. The working mechanism is the under-layered components providing services for the upper-layered components.

Dynamical generation of computational scripts. GridPSi client allows a user to dynamically generate computational script after finishing problem modeling and solution scheme specification. This script will be delivered to GridPSi server for translation and execution. The Script Generator family is in charge of generating computational scripts, which is comprised of a set of domain-specific Script Generators residing in the middle layer of the client package. A Script Generator takes serialized Java objects constructed by the under-layered components as arguments to create a structured binary file on which all definitions and specifications of a given simulation are recorded. In post-analysis process, a computational script can be modified according to the user's needs by either being loaded into a PDE wizard or in a text-based editor.

There are three reasons that we decided to implement this method. One is to utilize a legacy

package PSILAB [5], which has been proven to be reliable and efficient for solving PDE based problems. Another is to reduce the one-way network traffic from GridPSi client to a remotely distributed GridPSi server. Since the problem modeling and specification are the most error-prone processes in physical simulation, we argue that they should be finished locally. The last is that GridPSi server can be integrated with our selected Grid middleware seamlessly without any modification.

Grid resource accessibility. To lower coding effort and the barriers to entry for users wishing to exploit Grid technologies, we selected Unicore [6] as Grid middleware to feature our system Grid resource accessibility. The Java-based GridPSi can be integrated with Unicore gracefully while Unicore will take care of all the underlying security and network issues. A typical physical simulation based on GridPSi can be constructed into a Unicore job containing only five Unicore tasks.

Platform independence. For the improvement of system adaptability and integration with Unicore, both the GridPSi client and GridPSi server are implemented as Java applications to eliminate platform dependency.

Real-time visualization. This feature is still being under development. We are investigating use of the VTK [7] (Visualization ToolKit) to build our own visualization package for users to perform Grid based real-time visualization.

3. System architecture

In this section, we describe the architecture of GridPSi system as well as the functionality of core components consisting of the GridPSi package. The architecture of a GridPSi system comprising one GridPSi client and a total of three GridPSi servers running on local, remote and Grid computing systems respectively is shown in Figure 1.

3.1. GridPSi client

The GridPSi client can run on any Java–enabled user workstations or PCs possibly somewhere on the Internet, which is the interface to the end-users who use it to connect GridPSi server. It is designed to be "programming once, running everywhere", and covers the whole aspects of PDE based physical simulation except the solving process.

GridPSi client is implemented in a three-layered architecture with the client GUI on the top, PDE wizards, Script Generators, Solution Specification and Post-analysis packages etc in the middle and core functional components at the bottom. The working mechanism is the under-layered components provide services to the upper-layered components, components at the same layer can communicate with each other. Figure 2 shows the GridPSi client architecture.

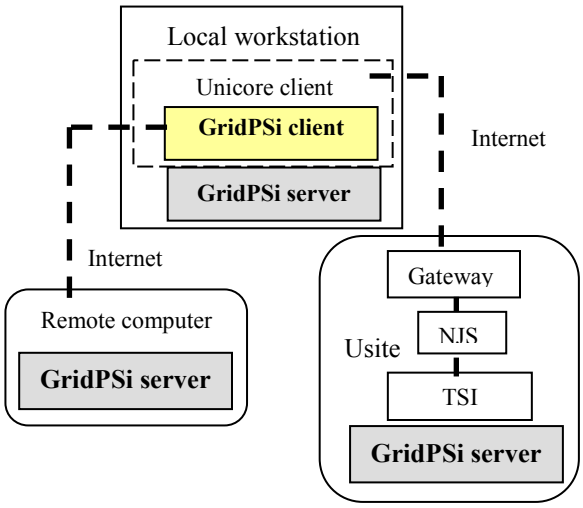

Figure 1. **The architecture of a GridPSi client connecting to GridPSi servers running on local, remote and Grid computing systems**

3.1.1. Client GUI. The client GUI is for the user to perform file operations, connect to the GridPSi server and invoke the under-layered graphical components, such as individual PDE wizard, post-analysis editor and visualization package. It also contains a problem domain specification component that consists of a PDE wizard selection dialog which categorizes physical simulations in a domain-specific manner, and two dialogs for collecting spatial dimension and time-in/dependent information of target problems. This component is the entrance to initialization of a new simulation.

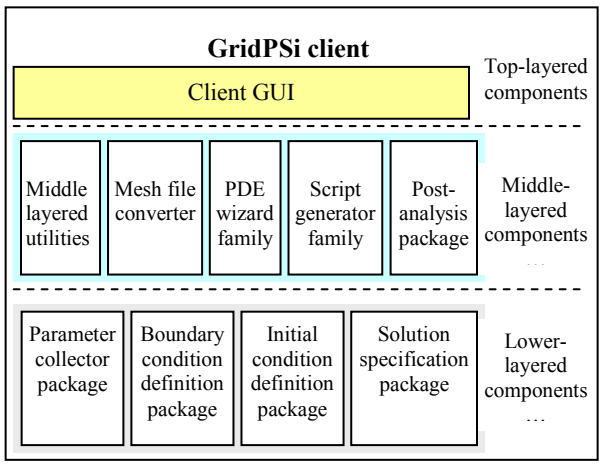

Figure 2. **GridPSi client architecture**

3.1.2. Middle-layered components. The middle-layered components are bridges between end-users and lower-layered components, which are high-level abstracted for the improvement of system modularity and extensibility while hiding the implementation details from users as much as possible. The main components of this layer are listed as below.

Mesh File Converter**.** We intentionally passed over the design of our own geometry construction and mesh generation package in the beginning of development of the GridPSi project, but took an approach to design a mesh file converter to convert mesh files generated by other mesh generation toolkit. All of us are smart enough to know that we are not smart as others in their research fields, and consuming is always cheaper than doing by oneself. Currently, we use Gmsh [8] to construct geometry and generate mesh since it is a mature public domain FEM mesh generator. The Mesh File Converter can not only convert all the first order mesh files generated by Gmsh to the format we need, but also extract the sub-domains and boundary regions into Java vector objects during mesh file conversion. These objects will be automatically passed to the later simulating steps for use.

PDE Wizard Family. PDE wizards are graphical components equipped with their own GUIs to assist users in defining PDE problems, applying boundary and initial conditions, and specifying solution details etc. Since PDE based problems can be characterized by a collection of specifications having a particular structure, and these specifications themselves, for example the governing equations can be predefined in templates by the lower-layered components given the spatial and temporal properties are specified. In physical simulation practice, many modeling processes do have some characteristics in common. Take boundary condition for example, despite the meanings are quite distinct in different discipline, but it can be summarized in only three types, namely Dirichlet, Generalized Neumann and Mixed. Likewise, they can also be predefined as templates by the low-layered components with each boundary condition type being designed as a serialized Java object, and represented in an abstracted form in the boundary condition definition dialog.

Developing PDE wizards does not add too much coding effort since most under-layered components can be shared and most classes can be inherited. But the pay back is a user-friendly interface with high-level abstraction and a problem-specific simulation from the beginning. In addition, A PDE wizard ensures that a successor simulation step can be invoked only if all predecessors have been completed successfully and all necessary data sets are defined or available. Currently some PDE wizards such as Structural Mechanics wizard, Electrostatics wizard are functional in our system while others are under development.

Script Generator Family. The reasons that we decided to develop the Script Generators have been well explained in Section 2. We would like to focus on the structure and implementation of the generated scripts here. The computational script is organized in sections with each section beginning with a keyword, which can be regarded as a high-level abstracted and concise language for expressing generic features that every PDE-based simulation possesses. The contents of a computational script are composed jointly by a domain-specific PDE wizard and predefined templates.

Since the Post-analysis Package has not been finished yet, it will not be described here.

3.1.3. Lower-layered components. The lower-layered components are core functional components in GridPSi client package. To accomplish efficient inheritance and well-designed templates for creating new components, the class hierarchy, function separation and component composition etc are carefully designed. Since the functionality of most packages in this layer can be well self-explained by their names, we will not give too much description here.

Parameter Collector Package serves as collecting parameters in three categories, constants such as material properties of experimental object(s), variables defined by a user during problem modeling, scalars for holding results that might interest the user. These defined parameters are passed to the corresponding components to construct coefficients and/or unknowns of the discrete governing equations and boundary condition equations etc.

Boundary Condition Definition Package provides both methods and GUI for a user to define and apply boundary conditions on his simulation model. The user's definitions together with predefined equation templates are used to construct serialized objects that will be passed to a script generator to write the boundary condition section in the computational script. This package is also responsible for checking definition incompleteness and errors.

Initial Condition Definition Package functions almost in the same way as Boundary Condition Definition package does.

Solution Specification Package allows users to specify solution details such as integration schemes, iteration steps and numerical method etc. This package collects user specified solution information for building the scheme section of the computational script.

Equation Template Provider Package provides predefined templates, such as equation templates, pre-constructed matrices and vectors etc. Some of the templates might be general for all components while

others are domain, dimension and time specific.

3.2. GridPSi server

The GridPSi server provides services for GridPSi clients no matter running on local, remote or Grid machines. It is designed to manage received jobs from GridPSi client, translate received computational scripts into high-level executable language and solve a given problem with user specified PDE solver, as well as send solved results back to clients upon request. The components of GridPSi server are shown in Figure 3.

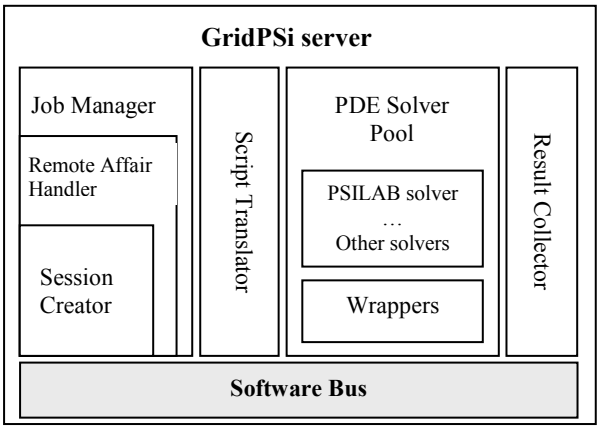

Figure 3. **GridPSi server architecture**

Job Manager serves as a central controller for receiving computational scripts and mesh files from GridPSi client, coordinating components to compute and control solving process and dispatching computed results back to GridPSi client.

Script Translator is in charge of translating a computational scripts into high-level executable programming language while checking errors before execution.

Solver Pool is a collection of PDE solvers. The solver part is designed to make use of some existing mature and legacy numerical packages. Since most legacy codes are not implemented in Java language, the wrapper collections are responsible for wrapping them into Java components.

Result Collector is used to save and collect computed results.

Software Bus is the underlying "glue" to integrate individual components of the server.

4. Access to computational Grids

In this section, we describe our practice on integration the GridPSi with Unicore to access computational Grids.

4.1. Construct Unicore site

In order to test running real Grid-based physical simulations through GridPSi, we constructed a Unicore site in our laboratory. The Unicore Gateway, NJS and TSI were installed on a Sunw Ultra-60 Sparc machine running Solaris 8 operating system.

We set up our own certificate authority first by using OpenSSL, which can issue necessary certificates for our research purpose. Configuring a Unicore site is not very difficult as long as one carefully reads the installation guides. The main point is to edit the property files of Unicore Gateway and NJS, Incarnation Database and TSI files etc in accordance with corresponding ports, hardware and software information of the machines on which the Unicore components are installed. Here, we only introduce a lesson that we have learned. That is, remember to use the "nohup" in front of the command when starting a TSI in a command line driven environment. Otherwise, the TSI will automatically disconnect from the NJS after the Xlogin session finished.

Since the mesh files and computational scripts of GridPSi are both text-based and there is a well-known line-break character problem that often happens when transferring text-based files from Windows operating system to Unix-alike operating systems, we selected a Unicore client installed on a machine running Windows XP operating system to test client-server communication in a heterogeneous environment. The purposes behind such a deployment are to figure out solution to the above problem and test the robustness of our system.

4.2. Integrate with Unicore

The GridPSi server has been integrated with Unicore gracefully. We only need to define three functions of GridPSi server, namely *psitrans, psicomp* and *psigo* as software resources and provide corresponding paths to these functions in the TSI-execution section of the Incarnation Database file.

The main purpose of programming GridPSi client as a plug-in to Unicore client is for real-time visualization. Since the visualization package of our system has not been finished yet, we postponed the plug-in programming. The current work style is simply delivering the computational script and mesh file of a simulation from the Unicore client to the Unicore site on which GridPSi server is running.

5. A simulating scenario

In this section, a case study on simulation of average distribution of vertical movements at the

bridge area of a three-dimensional piano soundboard is examined to demonstrate using GridPSi to conduct physical simulations on computational Grids.

5.1. Simulation background

The material properties of this piano soundboard are shown in Table 1.

Table 1. **Material properties of the soundboard**

	X axis	Y axis	Z axis
Dimension (mm)	935	935	6
Young's Modulus ($Kg/mm \bullet s^2$)	1.1564×10^7	9.0160×10^5	4.9980×10^5
Shear Modulus ($Kg/mm \bullet s^2$)	7.546×10^5	3.332×10^4	7.154×10^5
Density (kg/mm^3)	3.9×10^{-7}	3.9×10^{-7}	3.9×10^{-7}
Poisson's Ratio	$Pxy = 0.37$, $Pyz = 0.43$, $Pxz = 0.47$		

The governing equations for three-dimensional elastic stress analysis are as below.

Equilibrium

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xz}}{\partial z} + \frac{\partial \tau_{xy}}{\partial y} + F_x = \rho \frac{\partial^2 u}{\partial t^2} + R \frac{\partial u}{\partial t}$$

$$\frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} + \frac{\partial \tau_{xy}}{\partial x} + F_y = \rho \frac{\partial^2 v}{\partial t^2} + R \frac{\partial v}{\partial t}$$

$$\frac{\partial \sigma_z}{\partial z} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{xz}}{\partial x} + F_z = \rho \frac{\partial^2 w}{\partial t^2} + R \frac{\partial w}{\partial t}$$

Where σ_x, σ_y and σ_z are the non-zero axis stress components, τ_{xy}, τ_{yz} and τ_{xz} are the shear stress components, F_x, F_y and F_z are body forces, per unit volume; u, v and w are small displacements along axes, ρ is density, R is damping constant and t is time.

In this case study, body forces are ignored. The experimental methods and conditions are shown in Table 2, while the experimental parameters involved are shown in Table 3.

Table 2. **Experimental methods and conditions**

Type	Method / Conditions
Discretization Method	Finite Element Method (FEM)
Boundary Conditions	Fixed Dirichlet on the four sides Neumann on the top and bottom
Initial Conditions	No displacement at initial state
Numerical Method	Preconditioned Conjugate Gradient (PCG)

Table 3. **Experimental parameters**

Parameter	Expression
Frequency	fq = 1000 (Hz)
Damping Constant	R = 1000(Kg/s)
Amplitude of external stress	fext = 1/100
Time interval	$\Delta t = 4.17 \times 10^{-7}$ (s)
Bridge Mask	Kmask=max(0,sign(1,4.57- abs(x))) × max(0,sign(1,4.79 - abs(y)))
Stress distribution on the bridge area	Sz = sin (2πt×fq) × fext × Kmask
Variable being integrated on top	Wk = w × Kmask (where *w* is the unknown vertical displacement)

5.2. Problem modeling

In GridPSi client, each PDE based simulation is treated as an independent project for which a workspace will be created after a PDE wizard is selected. A project log file will be generated at the same time, which records all the information about reloading this project for quick modification. A typical problem modeling in GridPSi follows the steps below.

Choosing a PDE wizard. After the *new project* menu item of the client main menu is clicked, the PDE wizard selection dialog is loaded for selection. Since the spatial and temporal characteristics of a problem directly influence the governing equations and solution methods of a simulation, the selected wizard first asks the user to provide dimensional and time in/dependent properties for his project, which will be used by it to invoke suitable components and disabling unnecessary graphical items. Then the main framework of the wizard is displayed. In this example, a Structural Mechanics Wizard is selected. Figure 4 shows the PDE wizard selection dialog loaded into the client main framework.

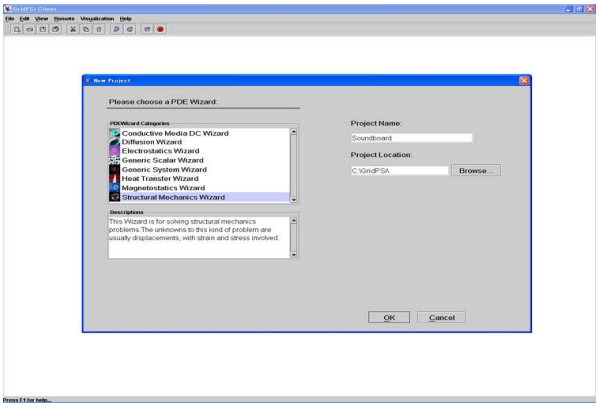

Figure 4. **The PDE wizard selection dialog loaded into the client main framework**

Constructing geometry and mesh. A user should construct the geometry and mesh by using Gmsh in advance. In this step, the PDE wizard requires a user to provide the mesh file of his project for conversion. After the mesh file is specified, the Mesh File Converter is invoked which first checks the correctness and consistency of the mesh file (e.g. the mesh file must be an uncorrupt mesh file in Gmsh format, and if the dimensional flag is 3D, the mesh file must be a 3D mesh file, otherwise an error message will be reported). Then the mesh file in GridPSi format is converted. In this example, the soundboard is meshed with a total of 1179 tetrahedron elements. One sub-domain and six boundary regions are extracted during conversion. The PDE wizard framework with meshed soundboard displaying inside is shown in Figure 5.

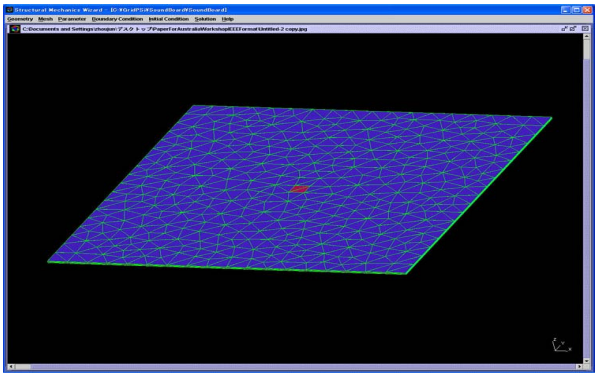

Figure 5. **The PDE wizard framework with meshed soundboard displaying inside**

Defining parameters. In this step, we need to define the parameters of material properties shown in Table 1 and experimental parameters shown in Table 3, as well as declare a scalar named "Wkoma" to hold the computed result at each iteration step. Figure 6 shows the Material Property Definition dialog.

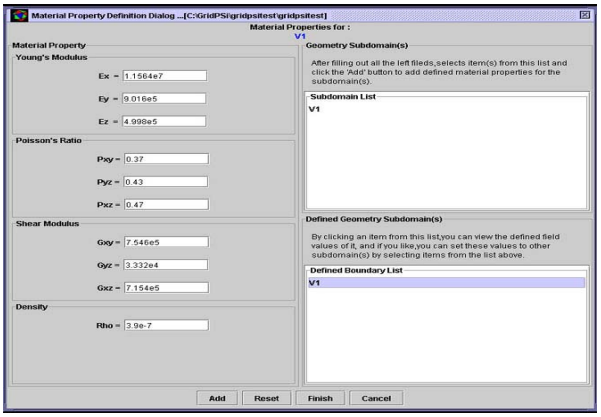

Figure 6. **Material property definition dialog**

Applying Boundary Conditions. The Dirichlet, Generalized Neumann and Mixed boundary condition expressed in abstract forms can be selected. Same boundary condition applied on different boundary regions can be defined by multiply selecting items from the boundary region list.

In our example, boundary conditions applied on the four sides are defined by selecting Dirichlet condition with zeros as input values in each field, and press "add" button after corresponding boundary regions being multiply selected. Boundary condition applied on the bottom is defined in the same way as above except for Neumann condition is selected instead. As for Neumann condition applied on the top, the definition is essentially alike except using the defined variable "Sz" as the field input for stress along Z-axis. Figure 7 shows the Boundary Condition Definition dialog.

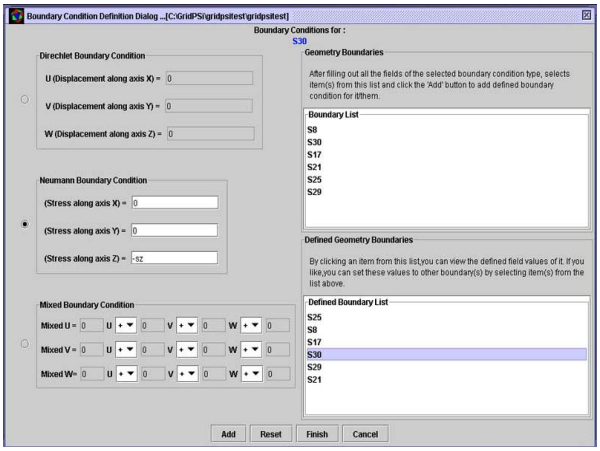

Figure 7. **Boundary condition definition dialog**

Applying Initial Condition. Initial condition is applied for time-dependent simulations only. Since this simulation is time dependent, we supply the values of beginning time, ending time and time step in the time definition panel, and zeros for each field in the condition definition panel of the Initial Condition Definition dialog that will not be illustrated with figure.

5.3. Solution scheme specification

The user can specify solution details such as iteration steps, numerical method and integration scheme etc in the multiple paged solution specification tabbed pane during this step. In this example, we select the PCG as numerical method with iteration steps up to 500 and integrate the defined variable "Wk" at the top region of this soundboard.

The governing equations of this example mentioned before are pre-formulated in the following forms with matrices and parameters predefined in templates.

Formulated equations

$$\rho\frac{\partial^2 u}{\partial t^2}+R\frac{\partial u}{\partial t}-Fx=div\left(A_u\cdot\cdot grad(u)+A_v\cdot\cdot grad(v)+A_w\cdot\cdot grad(w)\right)$$

$$\rho\frac{\partial^2 v}{\partial t^2}+R\frac{\partial v}{\partial t}-Fy=div\left(B_u\cdot\cdot grad(u)+B_v\cdot\cdot grad(v)+B_w\cdot\cdot grad(w)\right)$$

$$\rho\frac{\partial^2 w}{\partial t^2}+R\frac{\partial w}{\partial t}-Fz=div\left(C_u\cdot\cdot grad(u)+C_v\cdot\cdot grad(v)+C_w\cdot\cdot grad(w)\right)$$

where *Au, Av, ... Cw* are predefined matrices below.

$$A_u=\begin{pmatrix}\frac{E_x}{h}\left(1-p_{yz}{}^2\frac{E_y}{E_z}\right) & 0 & 0\\ 0 & G_{xy} & 0\\ 0 & 0 & G_{xz}\end{pmatrix}\quad\cdots\quad C_w=\begin{pmatrix}G_{xz} & 0 & 0\\ 0 & G_{yz} & 0\\ 0 & 0 & \frac{E_z}{h}\left(1-p_{xy}{}^2\frac{E_x}{E_y}\right)\end{pmatrix}$$

$$h=1-p_{xy}{}^2\frac{E_x}{E_y}-p_{yz}{}^2\frac{E_y}{E_z}-p_{xz}{}^2\frac{E_x}{E_z}-2p_{xy}p_{yz}p_{xz}\frac{E_x}{E_z}$$

5.4. Computational script generation

After finishing the processes above, The PDE wizard calls a structural mechanics script generator to generate the computational script for this soundboard example. Figure 8 shows the screen shoot of a small portion of the generated computational script.

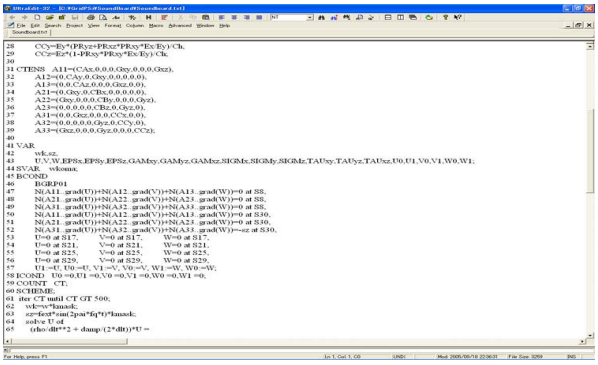

Figure 8. **Screen shoot of a small portion of the generated computational script**

5.5. Server execution

To demonstrate the Grid-based scenario, a GridPSi server running on our Usite is selected for execution.

Prepare a new job containing five tasks with clear execution order in Unicore client and then submit it to the Usite for execution. These tasks include an import task for transferring the computational script and mesh file to the Uspace, three command tasks for calling the functions of GridPSi server and an export task to retrieve the results back. Due to the page limitation of this paper, the computed results will not be shown in a Table, which can be obtained upon request. The job preparation and execution status are shown in Figure 9.

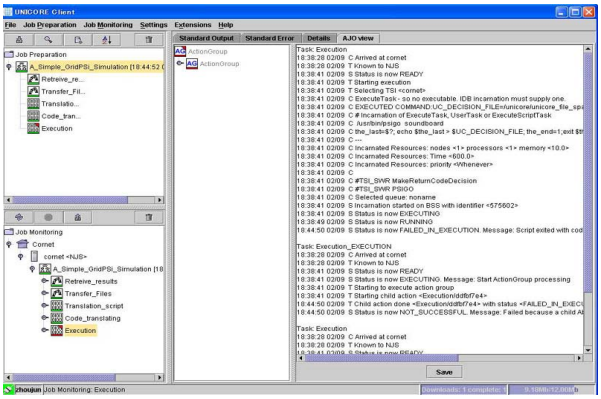

Figure 9. **Job preparation and execution status**

6. Conclusions and future work

The GridPSi described in this paper represents a novel attempt at creation of a Grid-enabled user-friendly PSE for physical simulation in Grid computing environment. In summary, we list three features that we believe the GridPSi demonstrates its strong point of development of the next generation of PSE.

High-level abstraction makes simulating process more specific and fairly straightforward.

Predefined templates greatly reduce users' work.

Open and adaptable architecture can tailor both the innovative technologies and legacy code into use easily.

Our future work will involve the design of post-analysis package, Unicore client plug-in programming and the design of more domain-specific PDE wizards

7. References

[1] Ian Foster, The Grid: Computing Without Bounds, Scientific American Digital, April 2003.

[2] E. Houstis and J.R. Rice, Future Problem-Solving Environments for Computational Science, computational science, mathematics and software, 2002, pp.93 – 114.

[3] Report of workshop on problem solving environments and scientific IDEs for knowledge, information and computing (SIDEKIC98), 1998, NM, USA.

[4] G. von Laszewski, I. Foster, et al, Designing Grid-based Problem Solving Environments and portals, 34th Hawaii International Conference on System Science, 2001.

[5] http://www.hitachi-ul.co.jp/PROD/PSILAB.

[6] http://unicore.sourceforge.net/unicoreatsf.html.

[7] http://public.kitware.com/VTK.

[8] http://www.geuz.org/gmsh.