PSE Park: a framework to construct Problem Solving Environments

Hiromichi Kobashi[1)], Shigeo Kawata[1)], Yasuhiko Manabe[2)], Masami Matsumoto[3)], Hitohide Usami[4)]

1) Graduate School of Engineering, Utsunomiya University (7-1-2 Yohtoh Utsunomiya 3218585)
2) Numazu National College of Technology (3600 Ooka Numazu 4108501)
3) Yonago National College of Technology (4448 Hikona-cho, Yonago 6838502)
4) Tamagawa K-12 & University (6-1-1 Tamagawagakuen, Machida 1948610)

In this paper, we introduce a new framework for Problem Solving Environment (PSE) construction that is PSE Park, which enables us to construct PSEs easily. The PSE Park consists of four engines; PIPE Server, Core, Registration and Console. A PSE designed and constructed in the PSE Park consists of Cores. The PIPE Server manages the Cores based on the core map, which expresses the connection of the Cores for a specific PSE. The output of each Core is retrieved and merged by the PIPE Server. All outputs of the Cores are saved and easily reused. If a selected Core has been deployed in another site, the PIPE Server manages the communication between the Cores. Therefore, the Core developers do not need to take care about the communication between the Cores in the PSE Park. The Core is independent from programming language, because each Core is executed individually as a process in the PSE Park. The Cores are registered by using the Registration engine, and users access the engines via the Console. We applied the PSE Park to develop a PSE for partial differential equation based problem. This PSE helps a part of simulation steps. There are six Cores to construct this example PSE. By using this PSE, users can execute a PDE-based simulation and get a detailed document about the simulation. We believe that the concept of the PSE Park, that is a framework for PSE development, presents a significant new direction in the world of problem solving environment.

***Key Words** : Computer assisted simulation, PSE, PDE*

1. Introduction

We believe that there are two movements around computing world: polarization in computing pattern and cloud computing. Computing power and network bandwidth are still growing, and they have been influencing absolutely not only computer engineers but also ordinary people. Many people have mobile phones and they access to the Internet to send or receive mails or to check their schedule. Those mobile phones have really powerful CPUs and network interfaces.

However, recently the high performance is no longer the best. PC, games and mobile phone markets have this remarkable tendency. Netbook is the example of this tendency. "Low performance but reasonable" is the concept of Netbook. This is why the performance of CPU was enough high in these few years and people are interesting in other features such as easy to use or low price. "Universal Design" is included in this tendency.

We call this tendency as polarization. Polarization is ahead in some areas and behind in others. Simulation area is the latter case. There are many research projects including national projects for high performance simulation such as super computer or high speed network development. On the other hand, there are few projects for low-end simulations: for example, simple simulations such as energy efficiency in a room or a house or education for students or so. Energy saving is one of the hot topics for many people. The needs of simple simulations or computations increase for our better life. However, there is no infrastructure or framework for easy-to-start simulations.

Cloud computing may provide one way for low-end simulation. Cloud computing is the new computing environment. In Cloud computing, software, middleware, OS and even computer are provided as services. Amazon, one of most famous company in cloud computing, provides middleware (Simple Storage Service or Elastic Compute Cloud) for users at a low price. We believe that more people will be interest in simulations for better life. For example, many sensors will be joined to computer network and user will make it easy to create input data such as room size or

current temperature. Computing resources will be available on users' demand via network in cloud computing.

In such cloud computing environment, we believe that PSE (Problem Solving Environment) is required for simulations. PSE is 'a system that provides all the computational facilities necessary to solve a target class of problems. It uses the language of the target class and users need not have specialized knowledge of the underlying hardware or software' (E. Gallopoulo et al.(1) , E. Houstus et al.(2)). PSEs help and accelerate to simulate what they want to know. The 1st generation PSEs are PSILAB (Y. Umetani et al.(3), C. Konno et al. (4)), ELLPACK (J. Rice et al. (5)) and NCAS (C. Boonmee et al. (6)(7), T. Teramoto et al. (8), S. Kawata et al. (9)(10)). These are libraries or program generation systems for scientific simulations. PSILAB and ELLPACK are the black box type of program generation system and NCAS is the white box type of program generation system. In the 2nd generation, PSE have extended the research areas including lower layer of computing environment. In Grid computing (11) environments, which are similar to cloud computing, many PSEs have been created and developed because of its complexes. Cactus Code (G. Allen et al.(12)) is a PSE for Grid computing for the simulation of relativity theory and IRIS Explore and SCIRun is the data flow management PSE (C. E. Goodyer et al. (13)). NAREGI-PSE is the portal to Grid computing environment (S. Kawata et al. (14), H. Kanazawa et al. (15)). PSE for Particle Image Velosimetry (PIV) is the PIV Virtual Laboratory (Y. Kadooka et al. (16)).

However, these PSEs are the domain specific environments. PSE itself is useful, though it is difficult to construct or develop a PSE. As the 3rd generation PSE may include a meta PSE, which enables us to construct PSEs easily. In the near future many users may access the cloud computing to start simulations for their hobby, better life, etc; one user may have his/her own PSEs as his/her simulation environments.

We developed a framework, which makes it easy to construct PSEs for many kinds of domains. We call it the PSE Park. Users, who want to simulate problems, access to the PSE Park, construct their PSEs, and execute simulations for their purposes.

In this paper, we describe about the PSE Park in Section 2. We construct an example PSE for PDE (partial differential equation) -based problem on the PSE Park. Section3 presents an adaptation example of the PSE Park to this PDE-based PSE. In Section 4, we conclude this paper.

2. PSE Park

A PSE construction framework, that is, the PSE Park is described in this section. The PSE Park makes it easy to construct PSEs on a cloud computing environment. Fig. 1

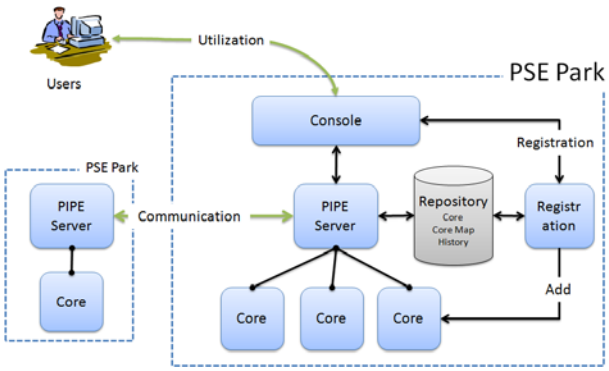


Fig. 1 PSE Park Architecture

presents the architecture of PSE Park. The PSE Park provides functions as Cores, which are required to construct PSEs. The PIPE Server in the PSE Park handles the Cores and constructs PSEs. The PIPE Server can communicate with another PIPE Server and use Cores easily. Users access to the Console in the PSE Park and construct/use a PSE or registered Cores and a Core Map.

The PSE Park has following the characteristics:

- Module Base

In cloud computing environment, we believe that operations and functions are highly modularized than ever to use computers effectively. Our PSE Park also provides functions to construct module-based PSEs, and each function is supported by a Core. Users, who construct PSEs, connect the Cores for their PSEs. In scientific simulation research areas, discretization methods like finite difference method are still being improved, and many researchers have requirements, that they want to try new methods quickly. In the PSE Park, if a user wants to try another method or to develop a new method, the user just only replaces the Core which is in charge of the specific method.

- Transparent access to Cores

PSE Park provides a transparent access to Cores. This means that users can access to Cores regardless of the Core's location. In the cloud computing, user does not recognize which machine he/she uses. The PSE Park handles the communication among Cores instead of users.

- Verification and Validation

Users, who develop the PSE Park Cores, can register their Cores in the PSE Park. The PSE Park has many kinds of Cores and manages them. The PSE Park provides users many choices. However, this is sometimes unkind for entry users. The PSE Park provides some typical stencils called Core Map for each target domain. Cores in the Core Map are verified and validated to provide high reliability. Expert users also can use the core map but easily modify as they want. For example, an expert user may try a new method or know a better way to solve a problem. In this case, if a new core is verified, a renewed core map should be registered as a validated core

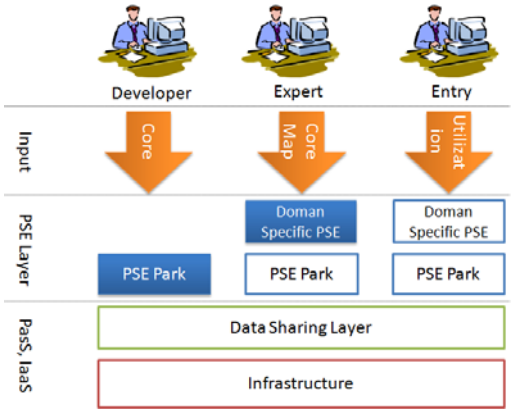


Fig.2 PSE Park Users

map.

We assume three types of users in the PSE Park; Developer, Expert and Entry-level user. (see Fig. 2)

Developers provide Cores to the PSE Park. They contribute to the PSE Park for further improvement and expansion of Cores registered or of the PSE Park itself. Experts provide and register the PSE Park Core Map. Their contribution is the same as traditional PSE developers'. They construct typical PSEs in each specific domain by using Cores, which are provided by Developers. Entry-level users can run simulations easily by using PSEs, which are constructed and registered by Experts.

We expect that Data Sharing Layer and Infrastructure in Fig.6 are provided by cloud computing environment. They are called as Platform as a Service: PaaS and Infrastructure as a Service: IaaS in cloud computing. The machines where PSE Park is running are provided from IaaS. Data in the PSE Park is shared by using PaaS between machines in PSE Park. We do not describe about them in this paper, because they are the function of cloud computing not PSE Park.

We describe each PSE Park's engines below:

2.1 PIPE Server

The PIPE Server is the main engine of the PSE Park. The PIPE Server manages and handles Cores and constructs PSEs by connecting Cores.

Users write Core Maps and pass them to the PIPE Server. The PIPE Server connects Cores based on the Core Map. In a Core Map, the relations of Cores are described, and the PIPE server constructs a PSE.

Fig. 3 shows an example of a simple Core Map expression. The Core Map expresses a PSE, which has Start Core, Stop Core and another four Cores in this example case. The PIPE Server executes the Cores between the Start Core and the Stop Core. In this Core Map, the PIPE Server executes the Core A first, and the output of the Core A is passed to the Core B as its input. If a Core has multiple inputs like the Core D, then the PIPE Server merges outputs of the Core B and the

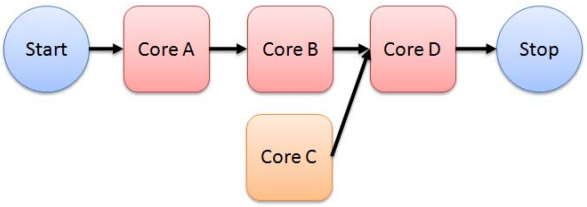


Fig. 3 Core Map example

Core C. The Core D is executed with the input, which are the merged output of them. The PIPE Server stores each output, and the output can be easily reused.

2.2 Core

Core is executed as a function of PSE. Cores collaborate with each other by the PIPE Server. In PSE Park, Core is executed as a process and is not compiled at execution time. This means that Core is independent from language, and users can implement Cores by using their well-known language. This makes it easy to develop Cores for developers or experts, and to register Cores into PSE Park for many users.

The communication between a Core and a remote Core is handed by the PSE Park, so that users do not need care about it. It is usual in cloud computing environment that all Cores, which user wants to use, are not located on the same site. If users should prepare a communication function, communication handling tends to be difficult. The PSE Park handles the communication among Cores.

2.3 Registration

Cores are registered into the PSE Park by the Registration engine. For registration of Core, Core itself and the definition of INPUT/OUTPUT are required. Core Map can be also registered with the Registration engine. After the registration, user can use the Cores and the Core Map.

The Registration engine saves an execution history of each Core, and the history and the Core map registered can be reused easily. The Core and Core map managements are handled by the Registration engine.

2.4 Console

The Console provides an interface to the PSE Park functions. Users can create Core Maps namely PSEs themselves or reuse previously constructed PSEs, which are registered Core Maps. The Cores and Core Maps are registered from the Console.

3. PSE for PDE based problems in PSE Park

We constructed a PSE for PDE-based problems in the PSE Park. In this chapter, we describe an adaptation example of the PSE Park to an example PDE-based PSE.

3.1 PSE for PDE

There are five steps in simulation, "problem discovery", "program designing / writing", "program execution", "aggregate execution result" and "discussion". If an engineer

or researcher wants to run simulations in his target problem, he/she should study not only his/her problem itself but also how to use computer, how to write program or how to execute program. For this reason, execution of simulation may be difficult even in his/her specialty domain.

The PSE helps a part of simulation steps, ”program designing / writing” or ”program execution”. Users can easily create a program on the PSE Park, and then they can simulate and solve the target problem on the PSE created on the PSE Park.

The target of this example PSE is PDE-based problems. This PSE generates a simulation program from “equations”, “boundary conditions”, “simulation domain”, “initial conditions” and so on. This PSE is a White Box system, which shows intermediate results, and a document of the target problem is automatically created.

3.2 Cores in PSE for PDE based problems

The Cores in this PSE are followings:

- Input Equation Core

This core helps input target problem. Users describe their problems by using this core. PDEs themselves, the initial conditions, etc. are the inputs of this PSE.

- Difference Method Core

A difference method is defined in this Core. For example, an explicit method or the SOR method or another one is selected or specified.

- Equation Manipulation Core

This Equation Manipulation discretizes given equations. All intermediate results are saved and user can check them if needed.

- Program Generation Core

This Core creates a program by using the discretized and manipulated equations, the initial conditions, the boundary conditions, etc.

- Program Execution Core

This Core executes the generated program. This Core compiles the program generated or links libraries if needed.

- Document Generation Core

This Core creates a document from the intermediate results of the Cores.

3.3 Adaptation

We have developed a PSE for PDE based problems by using Cores which we presented in 3.2. This PSE is focused on a 2-dimensional heat conduction phenomenon. The Core Map of this PSE is shown in Fig. 4.

3.3.1 Input Equation Core

Input Equation Core passes the inputs defined by users to the Equation Manipulation Core. The parameters, which users set in this PSE are “computation target in equation”, “computation domain of simulation”, “equation should be solved”, “physical quantities point (“in between point” or “on the point”)”, “initial condition of simulation” and “boundary condition of simulation”.

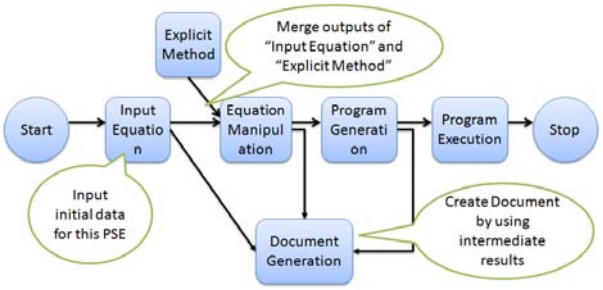


Fig. 4 Core Map for PSE for PDE based problems

The initial conditions of this result are following:

Equation;

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \quad (1)$$

Computation Domains;

$$0.0 \leq x \leq 1.0,\ 0.0 \leq y \leq 1.0 \quad (2)$$

Initial Values;

$$0.0 \leq x \leq 1.0,\ 0.0 \leq y \leq 1.0 \quad (3)$$

$$0.0 \leq x \leq 1.0,\ 0.0 \leq y \leq 1.0 \quad (4)$$

Boundary Condition;

Periodic boundary in both x and y axis

3.3.2 Explicit Method Core

In this PSE, we adapted an explicit method as the difference method. The Difference Method Core generates discretized equations for each physical quantities on specified points. In the Explicit Method Core, the discretized equations are followings:

$$\frac{\partial u}{\partial t} = \frac{u_{l+1} + u_l}{\Delta t} \quad (5)$$

$$\frac{\partial^2 u}{\partial t^2} = \frac{u_{l+2} - 2u_{l+1} + u_l}{\Delta t^2} \quad (6)$$

$$\frac{\partial u}{\partial t} = \frac{u_{l+1} - u_{l-1}}{2 * \Delta t} \quad (7)$$

$$\frac{\partial^2 u}{\partial t^2} = \frac{u_{l+2} - 2u_l + u_{l-2}}{4 * \Delta t^2} \quad (8)$$

$$\frac{\partial u}{\partial t} = \frac{u^{j+1} + u^j}{\Delta j} \quad (9)$$

The PSE Park performs the discretizations for the basic PDEs based on the discretization method specified.

If user wants to use the SOR method, that is, one of implicit methods, user should exchange this Explicit Method Core to the SOR method Core

3.3.3 Equation Manipulation Core

Equation Manipulation Core receives outputs from both of the Input Equation Core and the Difference Method Core (Explicit Core). In fact, the PIPE Server merges outputs of the two Cores and passes them to the Equation Manipulation Core.

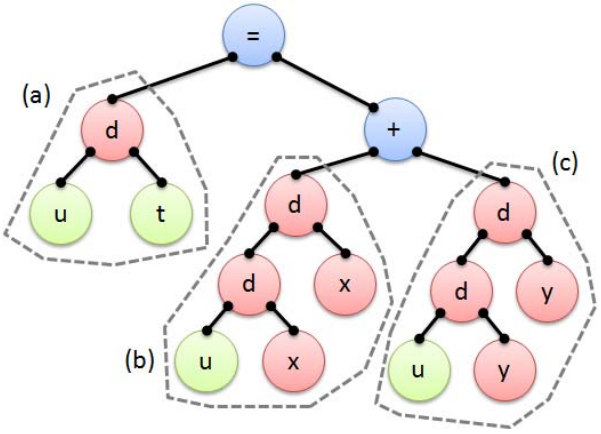


Fig. 5 Tree of equation

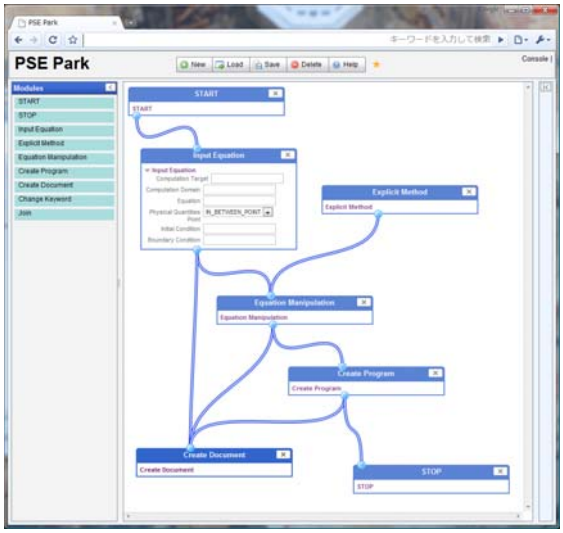


Fig. 6 Snapshot of Console

In the Equation Manipulation Core, the discretized equations are manipulated to obtain the new values for the target dependent variables.

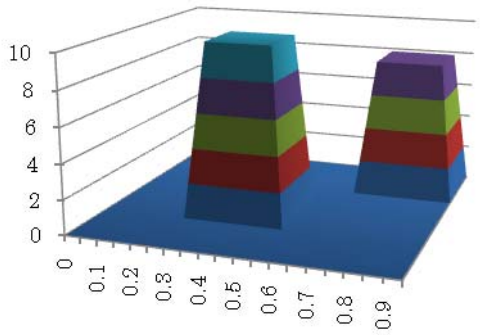
Fig. 5 shows the tree of equation. In the Equation manipulation Core, the obtained equation is converted to a tree. The tree has "=" as its Root. Operators like "+", "-", "*" and "/" are set as inner nodes. Values which are located on both sides of the operator are set as children of the operator node. Sub trees of the tree in Fig. 5 are replace to trees which are indicated in Explicit Method Core. In this Example, we use equation (9) for sub tree of Fig. 5 (a) and equation (6) for sub tree of Fig. 5 (b) and (c).

3.4 Simulation Results

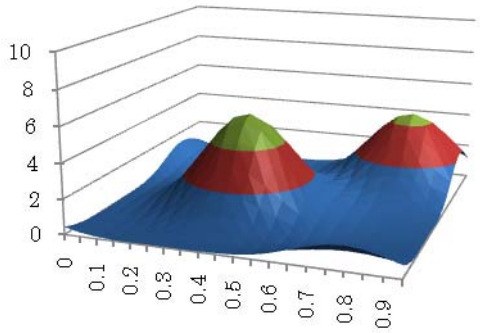
We can see the Core Map of this adaptation through Console in Fig. 6.

We show simulation results in Fig. 7. The simulation program is generated by the PSE for PDE-based problem, and the PSE was generated by the PSE Park.

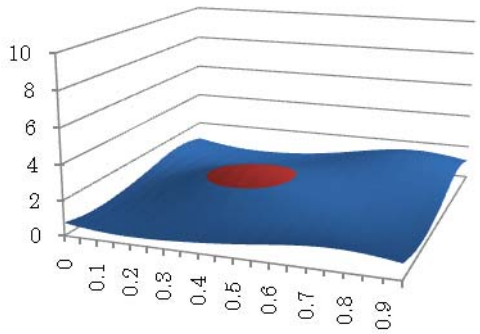
We put two peaks of heat quantity in this simulation. We can observe that these two are merged each other and heat quantity comes to constant in all field.



(a) step = 0



(b) step = 50



(c) step = 1200

Fig. 7 Simulation Results

3.5 Evaluation of PSE Park

We constructed a PSE by using the PSE Park. If the PSE Park is not available, users should construct the PSE by users. In this case users have to prepare the functions for the PSE (these kinds of Cores), and study how to handle processes and data in Cloud computing. In order to use cloud computing efficiently, users need to obtain know-how. It is quite difficult for most of entry users to study them.

In the PSE Park, users do not have to prepare functions and study how to use cloud computing. Functions are provided by Cores, and users can choose cores, which they want to use. The PIPE Server in the PSE Park handles processes and data

access instead of users. These are advantages of using PSE Park.

4. Conclusions

In this paper we developed and presented the PSE Park. The PSE park is a PSE construction framework, that is a kind of meta PSE. By using the PSE Park, a user can construct PSEs easily and perform simulations for his/her target problems. The PSE Park supports PSE developers and engineers, as well as low-end users for computer simulations. The PSE Park may meet users' requirements for problem solving: program generation, easy modification of programs, PSE development, automatic documentation supply and simulation execution support on cloud computing systems. The concept of the PSE Park, that is a framework for PSE development, presents an important new direction in the world of problem solving environment.

References

1) E. Gallopoulo, E. Houstis and J. Rice, "Computer as thinker/doer: problem-solving environments for computational science", Computational Science & Engineering, IEEE Volume 1, Issue 2, Summer 1994, pp. 11 - 23

2) E. Houstis, E. Gallopoulos, R. Bramley and J. Rice, "Problem-Solving Environments for Computational Science", Computing in Science and Engineering, vol. 4, no. 3, July-Sept. 1997, pp. 18-21

3) Y. Umetani, M. Tsuji, K. Iwasawa, H. Hirayama , "DEQSOL- A Numerical Simulation Language for Vector/Parallel Processors", Proc. of IFIP WG 2.5 Working Conference on Problem Solving Environments for Scientific Computing, 1985, pp. 147-164

4) C. Kon'no, M. Saji, N. Sagawa and Y. Umetani, "Advanced implicit solution function of DEQSOL and its evaluation", Proceedings of 1986 ACM Fall joint computer conference, pp. 1026 – 1033

5) J. Rice, R. Boisvert, Solving elliptic problems using ELLPACK, Springer-Verlag New York, Inc. New York, NY, USA, 1985

6) C. Boonmee and S. Kawata, "Computer-Assisted Simulation Environment for Partial-Differential-Equation Problem: 1. Data Structure and Steering of Problem Solving Process", Transactions of JSCES, Paper No. 19980001, 1998

7) C. Boonmee and S. Kawata, "Computer-Assisted Simulation Environment for Partial-Differential-Equation Problem: 2. Visualization and Steering of Problem Solving Process", Transactions of JSCES, Paper No. 19980002, 1998

8) T. TERAMOTO, T. NAKAMURA, S. KAWATA, S. MATIDE, K. HAYASAKA, H. NONAKA, E. SASAKI and Y. SANADA, "A Distributed Problem Solving Environment (PSE) for Partial Differential Equation Based Problems", Transactions of JSCES, Paper No.20010018, 2001

9) S. Kawata, H. Usami, Y. Hayase Y. Miyahara, M. Yamada, M. Fujisaki, Y. Numata, S. Nakamura, N. Ohi, M. Matsumoto, T. Teramoto, M. Inaba, R. Kitamuki, H. Fuju, Y. Senda, Y. Tago and Y. Umetani, "A problem-solving environment (PSE) for distributed computing" , Int. J. High Performance Computing and Networking, Vol. 1, No. 4, 2004, pp. 223-230

10) S. KAWATA, M. INABA, H. FUJU, H. SUGIURA, Y. SAITOH, T. KIKUCHI, "Computer-Assisted Liaison among Modules in a Distributed Problem Solving Environment (PSE) for Partial Differential Equation Based Problems", Transaction of JSCES, Paper No.20050029, 2005

11) I. Foster and C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, The Elsevier Series in Grid Computing, 2004

12) G. Allen, W. Benger, T. Goodale, H. C. Hege, G. Lanfermann, A. Merzky, T. Radke, E. Seidel, J. Shalf, "The Cactus Code: A Problem Solving Environment for the Grid", High-Performance Distributed Computing, 2000. Proceedings. The Ninth International Symposium, 2000, pp. 253-260

13) C. E. Goodyer1 and M. Berzins, Solving Computationally Intensive Engineering Problems on the Grid using Problem Solving Environments Software Environments and Tools. Springer, 2006

14) S. Kawata, H. Usami, Y. Hayase, Y. Miyahara, M. Yamada, M. Fujisaki, Y. Numata, S. Nakamura, N. Ohi, M. Matsumoto , T. Teramoto, M. Inaba, R. Kitamuki, H. Fuju, Y. Senda, Y. Tago and Y. Umetani, "A problem-solving environment (PSE)", Int. J. High Performance Computing and Networking, Vol. 1, No. 4, 2004, pp223-230

15) H. Kanazawa, M. Yamada, Y. Miyahara, Y. Hayase, S. Kawata, and H. Usami, "Problem Solving Environment based on Grid Services: NAREGI-PSE", Proceedings of the First International Conference on e-Science and Grid Computing, 2005

16) Y. Kadooka, H. Kobashi, J. W. Choi, Y. H. Lee and Y. Tago, "PIV Virtual Laboratory using Grid Technology", Proceedings of PSFVIP, 2003, F4027