14th World Congress in Computational Mechanics (WCCM)
ECCOMAS Congress 2020
11 – 15 January 2021, Paris, France
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

REVIEW OF PSE (PROBLEM-SOLVING ENVIRONMENT) IN COMPUTING SCIENCE

SHIGEO KAWATA[1]

[1] Graduate School of Eng., Utsunomiya University
Yohtoh 7-1-2, Utsunomiya 321-8585, Japan
kwt@cc.utsunomiya-u.ac.jp and http://www.ee.utsunomiya-u.ac.jp/~kawatalab/

Key words: Problem Solving Environment, Computer assisted computations, Scientific simulation, e-Science, PSE, Uncertainty in scientific computing.

Abstract. In this paper a review is presented on the PSE (Problem Solving Environment) concept in computing science. PSE is an emerging scientific and technological active area in computing science. In the PSE concept, human concentrates on target problems and works on solutions, and a part of application of solutions, which can be solved mechanically, is performed by computers or machines or software. PSE provides integrated human friendly innovative computational services and facilities for easy incorporation of novel solution methods to solve a target class of problems. PSE is an innovative concept to enrich our e-Science, e- Life, e-Engineering, e-Production, e-Commerce, e-Home, etc. The PSE-relating studies were started in 1970's to provide a higher-level programming language than Fortran, etc. in scientific computations [Trans. Jpn. Soc. Comput. Eng. and Science, 20171001, (2017)]. The trend at that time was natural to deliver more human- friendly programming environment, and was resulting in PSE, CAE (Computer Assisted Engineering), libraries, etc. At present PSE covers a rather wide area, for example, program generation support PSEs ["*Enabling Technologies for Computational Science*", Kluwer Academic Pub., 291, (2000)], education support PSEs, CAE software learning support PSEs, Grid/Cloud computing support PSEs, job execution support PSEs, e-Learning support PSEs, etc. This review paper includes the PSE definition, a brief history of PSE, example PSE study activities, uncertainty management PSE and future research directions in PSE.

1 INTRODUCTION

Problem Solving Environment (PSE) concept provides integrated human-friendly innovative computational services and facilities for easy incorporation of novel solution methods to solve a target class of problems. For example, a PSE generates a computer program automatically to solve differential equations [1-12]. Now the PSE concept covers rather wide areas in our society. PSE is an innovative concept to enrich science, human life, engineering, production and our society toward a programming-free environment in computing science. In the PSE concept, human concentrates on his/her target problems and works on solutions, and a part of application of solution, which can be solved mechanically, is performed by computers or machines or software. At present many kinds of computer-assisted PSEs are found everywhere in our life and in our society.

On the other hand, even today human power still contributes greatly to develop and write new software. For example, in scientific researches scientific discoveries are supported by theory, experiments and computer simulations. New researches tend to require new computer programs to simulate phenomena concerned. In another example, in developing new products engineers would also need new computer programs to develop the products cost-effectively. They may have to develop the new programs or learn how to use the programs for the product development. New services may also need new software systems. Therefore, the researchers and engineers may write or develop the new computer programs or learn how to use the programs. They do not like to develop nor learn the computer programs to solve their problems, but they like to devote their efforts to solve their target problems themselves.

In addition, computer simulations became the third important method after theoretical and experimental methods in science and engineering. Computer assisted problem solving is one of key methods to promote innovations in science and engineering, and contributes to enrich our society and our life from scientific and engineering sides. The PSEs have provided the new directions to support users, engineers and scientists for developing new services and new software, and also for solving their target problems based on computer systems.

The PSE-relating studies were started in 1970's to provide a higher-level programming language than Fortran, COBOL, ALGOL, PL/I and others in scientific computations. The trend at the time was reasonable to deliver more human-friendly programming environment beyond the higher-level languages shown above. Then the PSE research activity was started as well as activities of Computer Assisted Engineering (CAE) and software libraries. After the intensive developments in computer hardware and also in computer algorithms, researchers and engineers had expected an innovation in program writing and developing power. However, the enhancement in the programming power was relatively slow and weak compared with the enormous evolutions in the present hardware and algorithm power enhancements.

At present PSE covers rather wide areas, for example, program generation support PSE, education support PSE, CAE software learning support PSE, grid/cloud computing support PSE, job execution support PSE, learning support PSE, uncertainty management in scientific computing, PSE for PSE generation support (PSE for PSE), etc.

The paper includes a brief history of PSE, example PSE study activities and a future of PSE, including uncertainty management in scientific computing.

2 BRIEF HISTORY OF COMPUTER-ASSISTED PSE

PSE is defined as follows: "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" [6]. PSE provides integrated human-friendly innovative computational services and facilities to enrich science, life, engineering, production, commerce and our society. Based on the PSEs, human concentrates on target problems themselves, and a part of solution is performed mechanically by computers or machines or software.

In computing sciences, we need the computer power, the excellent algorithms and the programming power in order to solve scientific problems leading to scientific discoveries and development of innovative new products and services. So far, the computer power and the

computing algorithms have been developed incredibly, and have provided enormous contributions to sciences, productions and services. On the other hand, the programming power has not been developed well. The concept of PSE was proposed to support the programming power in science and engineering, and has been explored for decades.

In 1985, IFIP (International Federation for Information Processing) WG2.5 (Numerical Software) [16] organized a working conference on PSE and published the proceedings [17]. In 1991, a working conference on Programming Environments for High-Level Scientific Problem Solving was held [18]. In addition, a book on PSE was also published [19]. In 2007, a working conference on Grid-Based Problem Solving Environments was held [20]. A working conference on Uncertainty Quantification in Scientific Computing was held in 2012 [21]. The PSE activity including scientific computing environments is one of research projects in IFIP WG2.5 [16]. In these decades, international conferences tend to include the topic of PSE as one of standard topics in scientific computing. It has been recognized that PSE is an important research area in scientific computation and high-performance computing. In parallel, the PSE activities have started in several societies, scientific groups and countries. For example, in Japan, the PSE research group started in 1998 based on the previous individual PSE study activities, and the Japan Society for Computational Engineering and Science (JSCES) started in 1995, including the Study Group on PSE [22, 23].

The PSE studies have been extensively explored over the past few decades. The explorations have been supported by the reinforced computer power and algorithm power. PSE has boosted the programming power, and have enriched problem solving methods in science and engineering to bring us innovations.

One of PSE studies [1-12] has been extensively explored in order to support engineers and scientists to compute or solve their own problems based on for partial differential equations (PDEs), for example. One of the major objectives in PSE researches is to help users compute or solve their problems without heavy tasks, for example, without complete knowledge for computations [24] and/or the programs used. In this sense, the PSE provides an infrastructure for software for computational engineering and sciences.

One of typical PSEs for PDEs-based problems is ELLPACK [8, 24]. ELLPACK is a high level system for solving elliptic boundary value problems. One can solve routine problems by simply writing them down and naming the methods to be used. The ELLPACK high-level language can reduce the programming effort for a "routine" elliptic problem.

Another typical PSE for PDEs-based problems is DEQSOL [7, 10, 11]. DEQSOL was designed to develop an easy-to-use system for problem solving of PDE-based problems by finite difference method and finite element method. DEQSOL creates optimal Fortran codes oriented to the Hitachi vector processors.

Another PSE system of NCAS [1, 2, 4, 9] inputs a problem information including PDEs, initial and boundary conditions, and discretization and computation schemes, and outputs a program flow graph, a C-language source code for the problem and also a document for the program and for the problem (see Fig. 1). On a host computer a user inputs his/her problem, and NCAS guides the user to solve the problem. The distributed PSE for PDEs consists of several modules: problem description, discretization, equation manipulation, program design, program generation, documentation support, module liaison and job execution service. Each module is distributed on distributed computers. Each distributed module communicates with the host module, so that outputs from each module are visualized. Independent modules, which

are developed by other engineers or users for one of the functions specified above can also be used after adjustments to the distributed PSE interface [25], if necessary. The module liaison module generates an adapter module for the distributed PSE modules. The adapter module generated by the module liaison system inputs output data from preceding modules and/or external modules, and connects the data to the input data for the next module. The PSE contains all the information of the problem, PDEs, discretization scheme, mesh information, equation manipulation results, program design structure, variables and constant definitions and program itself. Therefore, the documentation support module also generates a document for the program generated together with the problem itself in the PSE [26].

A PSE module in NCAS also helps users generate MPI-based parallel simulation programs based on PDEs [4]. The NCAS capability explores possibilities to visualize and steer the parallel program design process. At present NCAS supports a domain decomposition in a design of a parallel numerical simulation program, and the domain decomposition is designed or steered by users through a visualization window. After designing the domain decomposition, the parallel program itself is also designed and generated in NCAS, and the designed parallel program is visualized and steered by a Problem Analysis Diagram (PAD). In NCAS, MPI functions [27] are employed for message passing, and a single program multiple data (SPMD) model is supported. The visualization and steering capabilities provide users a flexible design possibility of parallel programming.

Some PSEs provide a job execution support service on cloud or grid [28-31]. It is difficult for users to submit jobs to distributed computers and to retrieve calculation data from them in scientific computing. A robust job execution service system was also developed in a closed distributed computer system [28]. The job execution service system consists of dynamic system management servers, execution servers and data servers. The dynamic system management server is duplicated in order to keep the system robust, and has an assistant management server. The dynamic system management server has a function of the job execution system management, including software deployment, program compilation, job execution, job status retrieval and computing data retrieval. Users access the WWW page on the dynamic system management server, and the clients submit jobs. After

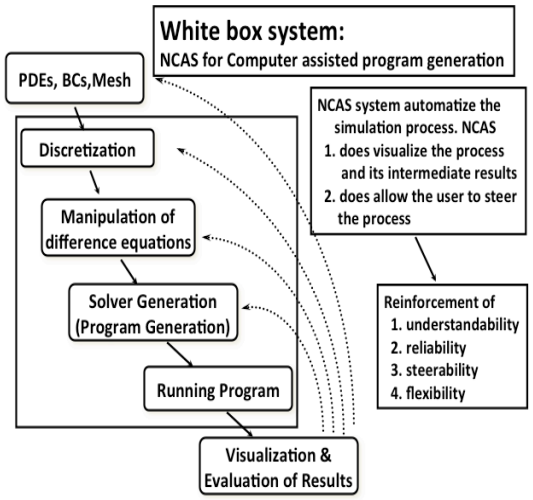


Fig. 1 An example PSE for computer assisted scientific program generation support: NCAS. NCAS inputs partial differential equations (PDEs), initial and boundary conditions, discretization method and algorithm, and outputs a C language program. The PDEs are automatically discretized and the program is generated mechanically. NCAS is a white box system, in which users can see and steer all the processes of program generation. NCAS system contains all the information for program generation, including basic equations, discretization schemes, discretized equations, boundary and initial conditions, mesh structure, program structure, and definitions of variables and constants. Therefore, a document for the corresponding program is also generated together with the program itself.

the submitted job finishes, the dynamic system management server collects the information from other distributed computers. The dynamic management server and its assistant server move dynamically to new servers, if the present servers become busy. The dynamic system management server also demands the execution server to transfer the result data to the optimal data server. The dynamic system management server copies the computing data and sends the compressed computing data to another optimal data server in order for a robust data storage system. The clients can deploy their programs, execute jobs and retrieve the result data by accessing only the WWW page in the job execution service system. This job execution management server also has a function of automatic system construction, so that the users can manage the setup of the job execution management system easily on their closed distributed computers.

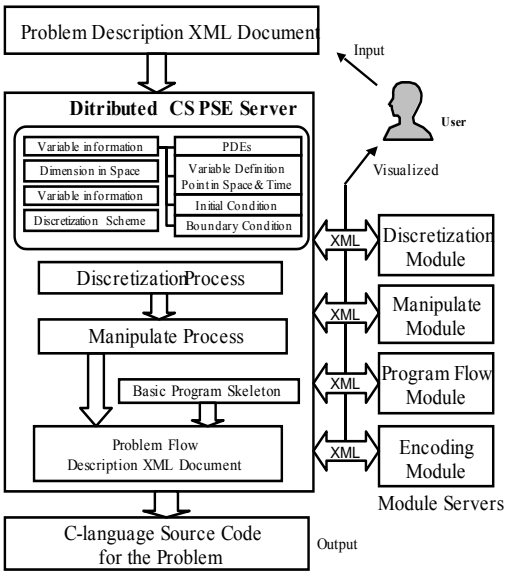


Fig. 2 Distributed-PSE: NCAS workflow.

Another remarkable example of PSE is an education support PSE [32]. Network-based learning has been taking an important role in education as helpful education tools. However, it is difficult for teachers to retrieve education data from students or to obtain data from the student activities. Therefore, a problem solving environment (PSE) for the education and learning support: TSUNA-TASTE [32] was developed. The TSUNA-TASTE system collects the system-usage statistics, the information for the windows used and the operation situation of the mouse and key board of all students. The data, which the system TSUNA-TASTE collects, are stored in a database on the TSUNA-TASTE system server. Based on the data collected, teachers can have the learning status data for each student, and can guide the students in a better way.

Another research issue in PSE is validation, verification and uncertainty control in scientific simulations. When a software gives incorrect results for users, it may cause some difficulties, errors and accidents, depending on target problems [21, 33-37]. The validation and verification mechanism is essentially important in scientific computing. This point was also pointed out by E. Houstis, J. Rice and his colleagues [38]. Standardization and benchmark problems in each field may help to perform the validation and verification. In addition, uncertainty management must be addressed intensively in order to avoid serious accidents and disasters in our society. PSE is one of candidates to manage the uncertainty in a relatively easy way [39]. The topic on the uncertainty is also discussed in this paper. There are many PSE examples studied so far. In the references of [17-20] and [25] one can also find the example PSEs. In the next section typical example PSEs are introduced.

3 PSE EXAMPLES

3.1 NCAS: A program generation support PSE for PDEs-based problem

PSE studies [2-14] for partial differential equation (PDE) based problems have been extensively explored in order to support engineers and scientists to compute or simulate their problems and products on computers in e-Sciences and e-Productions. One of the major

objectives in PSE researches is to help users compute or simulate their problems without heavy tasks, for example, without complete knowledge [11, 12] for computations or without a full programming [2-13].

In this subsection a program generation support PSE, called NCAS is presented. NCAS inputs partial differential equations (PDEs), the initial and boundary conditions, the discretization method and the algorithm, and outputs a C language program for the problem. The PDEs are automatically discretized and the program is generated mechanically. NCAS is a white box system, in which users can see and steer all the processes of the program generation. NCAS contains all the information for program generation, including the basic equations, the discretization schemes, the discretized equations, the boundaries and the initial conditions, the mesh structure, the program structure, and the definitions of all the variables and the constants. Therefore, a document for the corresponding program is also generated together with the program itself. In PSE for PDEs problems, one of problems, which should be addressed, is to develop huge PSE systems, including reusability of legacy PSE software. In order to solve this problem, a module-based PSE is proposed [9, 40]; each PSE module solves a part of PSE tasks, for example, a problem description interface, a discretization module, a scheme suggestion module, a program flow designer, a program generator, a data analyzer, a visualizer, and so on. If each module can be developed independently and works cooperatively and smoothly to solve one PSE job, the heavy work of PSE development may be drastically relaxed. In this subsection a distributed PSE, called NCAS, is introduced, which supports users to generate computer programs [1-4, 9, 26, 40].

The PSE system of NCAS inputs a problem information including discretization and computation schemes, and outputs a program flow graph, a C language source code for the problem and also a document for the program and for the problem. On a host computer a user inputs his/her problem, and the host guides the user to solve the problem. The distributed PSE for PDEs consists of several modules: a problem description, a discretization, an equation manipulation, a program design, a program generation, documentation support, a module liaison and a job execution service. Each module is distributed on distributed computers, and all the information is described by the Extensible Markup Language (XML) including the Mathematical Markup Language (MathML). Each distributed module communicates with the host module by using XML documents, so that outputs from each module are visualized. Independent modules, which are developed by other engineers or users for one of the functions specified above can be also used after adjustments to the distributed PSE interface, if necessary. Therefore, the concept of the distributed PSE extends the potential of conventional PSE systems. The PSE contains all the information of the problem, PDEs, discretization scheme, mesh information, equation manipulation results, program design structure, variables and constant definitions and program itself. Therefore, the documentation support module also generates

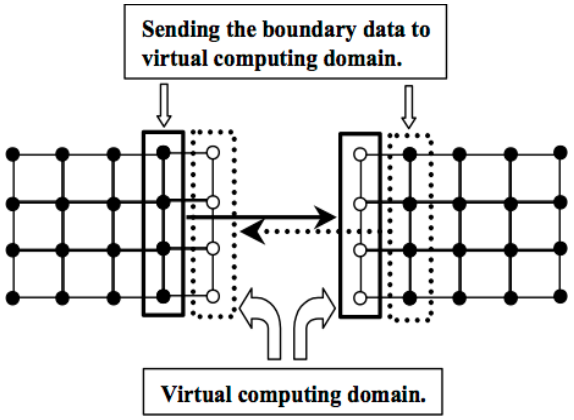


Fig. 3 Boundary data are communicated by the MPI functions in NCAS among domains decomposed. The MPI functions are automatically inserted into the program designed and generated in NCAS. The Finite Difference method (FDM) is employed in this example.

a document for the program generated and the problem itself in the PSE. The module liaison module generates an adapter module for the distributed PSE modules. The adapter module generated by the module liaison system inputs output data from preceding modules and/or external modules, and connects the data to the input data for the next module. The program generation PSE module provides a workflow shown in Fig. 2, and the user follows the workflow navigation for a problem generation.

The NCAS modules also help users generate MPI based parallel simulation programs based on partial differential equations (PDEs). The NCAS capability explores possibilities to visualize and steer the parallel program design process. At present NCAS supports a domain decomposition in a design of a parallel numerical simulation program, and the domain decomposition is designed or steered by users through a visualization window. After designing the domain decomposition, the parallel program itself is also designed and generated in NCAS, and the designed parallel program is visualized and steered by a PAD diagram. In NCAS, MPI functions are employed for message passing, and a SPMD (single program multiple data) model is supported. The visualization and steering capabilities provide users a flexible design possibility of parallel programming.

In the parallel program generation support in NCAS, for the data communication among the processors, the MPI functions are used. At least the boundary data for each domain decomposed are required to complete the computation in the adjacent processor (see Fig. 3). In NCAS the MPI functions are also automatically inserted to complete the parallel data communication programming. After specifying the domain decomposition information in NCAS, the parallel program is generated and provided to the users.

Figure 4 presents an example description of an input problem information, and Fig. 5 shows an example domain decomposition information. Through the NCAS visualization windows, for examples, shown in Figs. 4 and 5, one can check all the information and can also edit the information. In NCAS, after setting all the information for the problem description, the

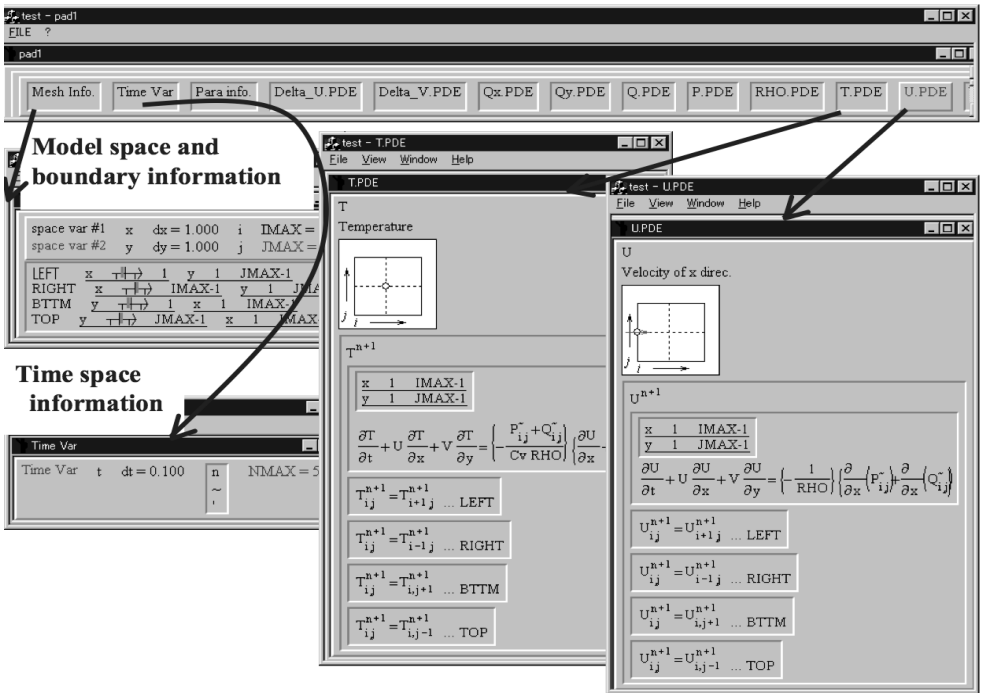


Fig. 4 An example PDE-problem description in NCAS. On each window users can edit the input description.

dicretization information and the parallelization information through the NCAS windows, all the information is visualized to the users and the users can edit all the information through the windows. The discretization of each PDE is also performed automatically; depending on the discretization information which users input through the NCAS windows, the PDEs are discretized and manipulated appropriately according to the PDEs solving scheme. Then NCAS designs the parallel program for the problem, and outputs the parallel program and the corresponding document. Figure 6 shows an example MPI program automatically generated in NCAS.

In order to check the dynamic load balance function automatically generated by NCAS, during the computation an additional load was applied as shown in Fig. 7 (the left graph): by the additional load the computation time increases much in this specific case, if the static load balancing is used. When the dynamic load balancing method is selected in this example case, NCAS generates the functions, which measure the load balance of each machine dynamically, and according to the measured result each domain size is changed and adjusted dynamically to minimize the computation time. The right graph in Fig. 7 demonstrates the viability of the dynamic load balancing functions generated in NCAS, and the computation time reduction is significant in this case.

In the distributed PSE all the modules are distributed on network-linked computers. The information for the distributed modules and the computers are registered in a host computer. Newly developed modules by some users or scientists or so can be also registered in the host PSE server. The distributed PSE host server has the registered information for the modules oriented to one specific purpose, and users can obtain the information for each module and can select one of the modules to perform one task in all the PSE process.

The communication is accomplished through an interface using WWW server and Applet. The PSE server sends information described by XML to a module, and the module performs the task. The module sends the result based on the input XML information back to the PSE

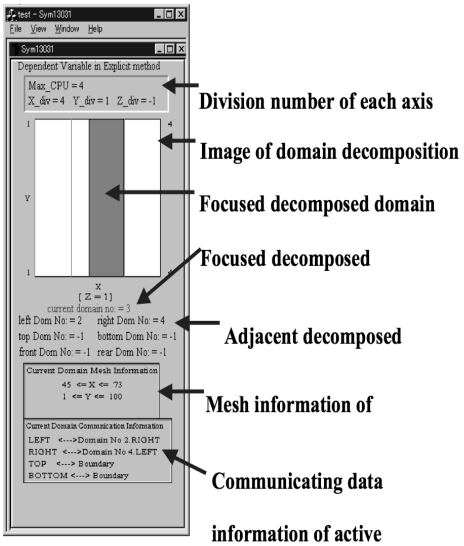


Fig. 5 Input and visualization of domain decomposition information.

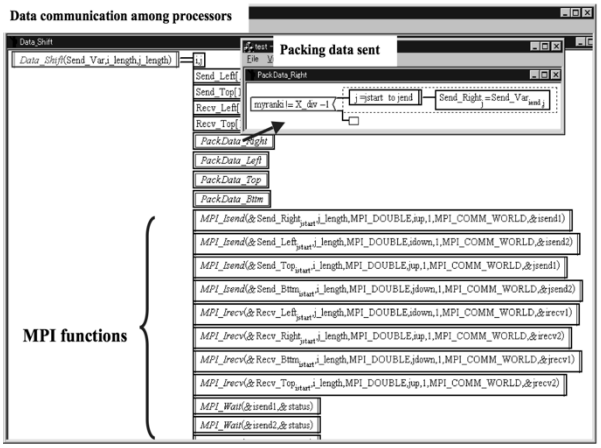


Fig. 6 Visualization of MPI functions designed for a domain decomposition information in NCAS.

server. The result is visualized so that the user can check if the result is appropriate. After the successive processes, finally the NCAS generates a designed program flow and then a C program.

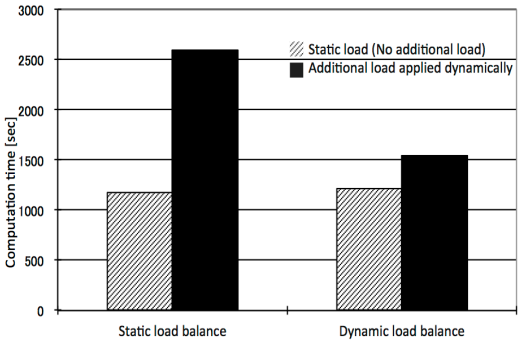


Fig. 7 A performance test result for a dynamic load balance. An additional load was applied during the computation, so that the computation time increases as shown in the left graph. When the dynamic load balancing function generated automatically by NCAS is used to this specific case, the domain size changes automatically depending on the machine load during the computation, and the result of the dynamic load balancing in the right graph shows the viability of the dynamic load balance functions generated by NCAS.

3.2 An education support PSE

Network-based e-Learning is one of important education ways. In addition, the network-connected personal computers have become popular to schools and homes widely. In the network-based e-Learning, each learner can access education contents through the network anytime and anywhere. In an actual education, a network-based e-Learning system becomes popular in a long-distant learning and at the same time in a class-room education. Even inside class rooms, each computer is connected and can be used as a detecting device for the learners' progress and status. An e-Learning server may have facilities such as a user identification method or a file sharing tool in the network-based environment.

It would be difficult for teachers to know the learning state of students through each personal computer connected by a network. Without the detailed information of the students' achievement, it is difficult for the teachers to perform an appropriate guidance and education depending on the students' learning level. Therefore, the state of the students is important and required for the suitable guidance. The education-support PSE system, which provides teachers the student-achievement information, helps them in their teaching planning or the appropriate guidance.

The network education support system (TSUNA - TASTE) consists of four parts. The first part is an agent of student. It is a software, which always works on each personal computer of the student. The agent obtains the operation information of each student. The data are obtained from the operation information of the student through the OS with a resident software working on the personal computer of each student. The second part is the education support server, that collects the data, which each student agent obtains via the network. The third part is the database system. The database system stores the student profile data, the student personal data, the curriculum data and the teacher's personal data. The fourth part is the WWW server displaying the information stored in the database. The WWW server (Servlet system) provides an interface to the TSUNA-TASTE handling.

The agent for each student resides on the memory of the personal computer of the student and performs the following three operations. First, the agent exchanges the messages with the education support server. The education support server transmits the messages to each agent. The student agent analyzes the messages, and obtains the process priority and the start time of the process described in the messages. The agent stores the message data in its task table.

Secondly the agent manages the module program execution based on the task table. The module programs are small-size programs, which retrieve and output the student personal data from the student computers. The student personal data includes the achievement data, the operation data, the active window names and the process names. The third operation is a job to send back the data, which each module program collects, to the education support server. The module programs obtain the information of the student personal data from the OS of the student PCs. The module programs are implemented in the C++ language.

The module collects information about the students, however it does not store the raw data for security. It converts the raw data into statistics data. The transmission data are encoded and transferred. Furthermore, the personal information is not included in the data transmitted to the server. Thus, this system is robust for the electronic eavesdropping of data.

The education support server receives the operation data of the students through the student agent, and transmits the teacher's instructions to the students through the student agent. The education support server marks the students' absence, and identifies the students and their PCs. The education support server sends the messages of the data process demand to the agent of the student, and transmits the student personal data to the database server. The education support server also retrieves the student personal data requested by the teachers through the WWW server, and sends them back to the WWW server in the TSUNA-TASTE. The education support server is built using the Java language.

The data, which the education support server receives, are stored in the database. The database includes the private information of each student and teacher. The data contains the private information such as college student registration numbers, mail addresses and so on. These unchanged data are stored in the database together with the temporal data like the site of the student PCs.

The WWW server system provides the user interface of the TSUNA-TASTE system. The teachers can obtain the state data of the students from the WWW system. The WWW server system presents the student achievement data, the learning progress and error occurrences situation during the programming exercises. The WWW system also provides an input interface to control the action of the TSUNA-TASTE: Through the WWW system, teachers can send a data gathering command, monitor the students' present usage of applications, and kill the unnecessary application processes on the students' PCs. The TSUNA-TASTE may open a new helpful e-Learning world.

3.3 Job execution support PSE on GRID/CLOUD

It is difficult for users to submit jobs to distributed computers on Cloud /Grid and to retrieve calculation data from them in scientific computing. In this subsection, a robust job execution service system is introduced in a closed distributed computer system [28, 29, 41]. The job execution service system consists of dynamic system management servers, execution

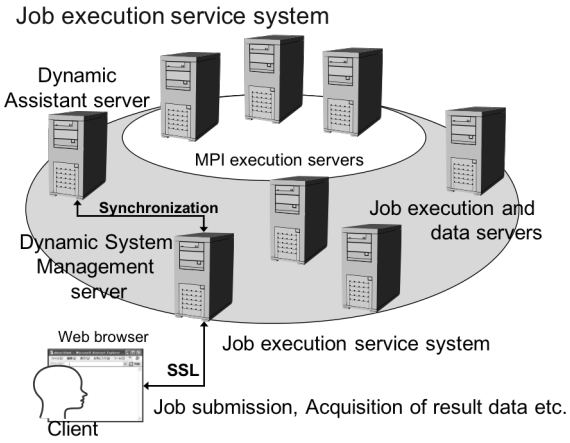


Fig. 8 Job execution service on distributed computers n Cloud / Grid.

servers and data servers as shown in Fig. 8. The dynamic system management server is duplicated in order to keep the system robust, and has an assistant management server. The dynamic system management server has a function of the job execution system management, including software deployment, program compilation, job execution, job status retrieval and computing data retrieval. This system does not require special middleware for Cloud /Grid. Users access the WWW page on the dynamic system management server, and the clients submit jobs. After the submitted job finishes, the dynamic system management server collects the information from other distributed computers. When the present servers become busy, the dynamic management server and its assistant server move dynamically to new servers. The dynamic system management server also demands the execution server to transfer the result data to the optimal data server. The dynamic system management server copies the computing data and sends the compressed computing data to another optimal data server in order for a robust data storage system. The clients can deploy their programs, execute jobs and retrieve the result data by accessing only the WWW page in the job execution service system. This job execution management server also has a function of automatic system construction, so that the users can manage the setup of the job execution management system easily on their closed distributed computers.

Users access the WWW page on the dynamic system management server, and the clients submit jobs. After the submitted job finishes, the dynamic system management server collects the information from other distributed computers on Cloud / Grid. The clients can deploy their programs, execute jobs and retrieve the result data by accessing only the WWW page in the dynamic system management server.

The job execution service system acquires necessary resource information for servers for job execution and for data retrieval and storage. The resource information contains CPU architecture name, CPU operation frequency, total memory, memory in use, unused memory, load average and unused capacity of hard disk. The system sorts the resources in order for effective job execution. The users can find the resource information on the job execution service system WWW page.

Clients access the dynamic system management server through the WWW page of the system, and perform computing. Through the WWW page, the clients can up-load source files or executables to the dynamic system management server, select computing servers from among resources recommended by the system, and set execution environment. When two or more files are required for one job, the client should compress those files. When MPI jobs are executed, computing servers, on which MPI is installed, are recommended to the clients.

The compilation command, the execution method, the comment and the server name for job execution are specified by clients. The clients can also specify the storage location of the result data of the job. When the clients do not especially specify the storage data server, the system forwards the result data to the best data server. When input information is not sufficient, the job execution service system displays an error message, and advises to input the required input data. The clients can select the execution methods, or make scripts for the execution on the WWW page in a compressed file format.

When the job setting ends, the job execution service system forwards the job to a pertinent server or a server set based on the setting information. The setting file is described in XML, and contains the computing server information, the compilation command, the execution information and the data storage server information. When a compressed file, which contains

source files/binaries and a make file, is sent to the computing servers, the compressed file is decompressed and the decompressed information is sent to the dynamic system management server, so that the clients can check if the file decompression is succeeded. When the compilation or the execution errors appear, the computing server notifies them to the dynamic management server. When the execution server specified is occupied by another job, the job is scheduled by the dynamic system management server.

When a job ends, the job execution server forwards the result data to a pertinent server based on the setting information. In addition, its compressed result data is stored in another data server. The result data duplication makes the data server robust and fault tolerant. When no data server is specified in the setting information, the computing server asks the dynamic system management server about the data storage servers. Based on the unused hard disk capacity, the better two data storage servers are selected from among the servers, on which no jobs run. One is for the result data uncompressed and the other is for the compressed backup data. When the result data is stored on the data servers specified, the data server locations are notified to the dynamic system management server. The client can refer to and can download the data from the WWW page on the dynamic system management server.

3.4 Toward uncertainty management PSE

Computer simulations and high-performance computing have also contributed to scientific discoveries, innovations and new findings. In physics, chemistry and other disciplines, mathematical equations including PDEs (partial differential equations) may be employed to model phenomena concerned. The mathematical equations are discretized so that the equations can be treated and solved on computers. In computer simulation, then numerical data are obtained and analyzed on physical quantity of interest (QOI). Not always but frequently QOI

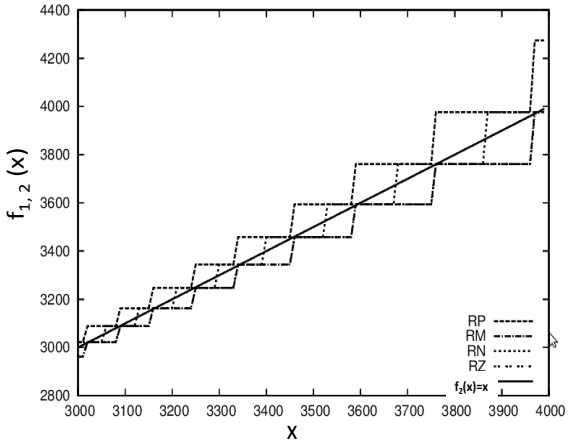
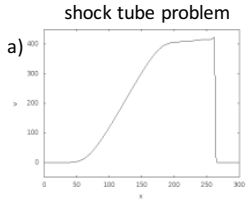


Fig. 9 Numerical results for $f_1(x) = \left(-ln\left(exp(-x^{-4})\right)\right)^{-1/4}$ and $f_2(x) = x$ based on the IEEE rounding methods [35]: Round to nearest (Even) (RZ), Round Upward (RP), Round Downward (RM) and Round toward 0 (RZ). For the sensitive case of $f_1(x)$, large differences appear among the numerical results based on the four IEEE rounding methods.

is visualized.

In the process of computer simulation or scientific computing, the origin of uncertainty is found[21, 33, 35-37, 42-45]: physical model uncertainty, mathematical model errors, unknown input data, unknown boundary condition errors, numerical model errors, insufficient numerical precision, round-off error, floating point precision, programming errors, data processing errors or uncertainty, visualization errors, human errors, etc. So far the uncertainty or the errors in scientific computing have caused serious accidents and disasters.

In 2009 an airplane of Air France447 met a blocking of all the Pitot tubes, by which airplanes measure their flying speed in the air [43]. The Pitot tube consists of a tube pointing directly into the air flow, and has multiple holes to detect the airplane speed by the difference between the static and dynamic pressures. All the holes were blocked by the ice in the condensed super cooled air moisture. That means that the speed of the airplane becomes very low, and the computers started to speed the airplane up. But the three pilot crews could not find the reason for the acceleration. Finally at the steep attack angle the airplane stalled and was crashed into the ocean. All 228 people were killed by the accident. In this disaster, the input data from the Pitot tubes was wrong to the computers. Another accident happened at the Gulf war in 1991, and a missile killed unpredictably 28 people [44] by a failure to track and intercept an incoming Iraqi scud missile by an inaccurate calculation performed. This accident happened due to the software error.



b) Single precision

ROUNDING MODES	RMS
RN	3.64×10^{-1}
RP	1.87×10^{-1}
RM	1.70×10^{-1}
RZ	3.09×10^{-2}

c) Double precision

ROUNDING MODES	RMS
RN	2.50×10^{-5}
RP	8.90×10^{-5}
RM	3.20×10^{-5}
RZ	3.20×10^{-5}

d) Quad precision

ROUNDING MODES	RMS
RN	4.13×10^{-13}
RP	2.58×10^{-13}
RM	1.47×10^{-13}
RZ	1.47×10^{-13}

Fig. 10 A shock tube problem in fluid dynamics is solved by multiple programs with the different rounding methods and precisions (see Fig. 10a)). The single precision shows a significant difference among the computational results depending on the rounding method as shown in Fig. 10b). The double precision program provides better results as shown in Fig. 10c). However, the quadruple precision shows sufficiently accurate results (see Fig. 10d).

From real physical phenomena, physical model is constructed to find out which physics is concerned. In this process, some physics involved could be missed, and it may lead to uncertainty to describe the real phenomena. From physical model, a mathematical model is derived. The mathematical model does not always present the real world. Sometimes exact equations are not known, or some perturbations are ignored, which may be essentially important in the phenomena. The mathematical model may often include PDEs, which should be

discretized to solve on computers. In the discretization, well-known numerical instabilities may appear. In the computing program the numerical stability condition must be always fulfilled during a whole computation. If the stability condition is violated during the computation, the numerical results do not meet the validity. The floating-point error is another issue in computer simulation, because recently long computing time on super computers becomes common to obtain meaningful results. A finite digit number is used in the computers to describe real number and to perform arithmetic operations. This induces the floating-point errors, including rounding error and truncation error. For the computations, input data and boundary data should be prepared. Sometimes the input data is measured, and in this case the measurement itself may have some errors. Sometimes it is difficult to find the exact input data, which may induce another source of uncertainty. We may approximate the input data. After or during computations, output data come up and the data processing is needed to find characteristics of QOI or so. We also often perform scientific visualization. In the visualization process, we could find some uncertainty depending on the visualization method and precision [34]. Sometimes hidden important structures could not be found in the simple visualization, or a surface position may not be exact. Human errors also share the contributions to uncertainty with other issues discussed above.

Uncertainty, verification and validity in scientific computing have been also studied intensively [21, 33, 35-37, 42-45]. On the other hand, each uncertainty has its own origin, and has different characteristics with one another, as we have discussed above. Just one solution might be insufficient to manage all the uncertainty.

This consideration suggests us multiple solutions in the various uncertainty characteristics. One promising possible way is to develop a PSE for the uncertainty management [21, 39, 42]. W. H. Enright has proposed an interesting PSE as a tool for the verification of approximate solutions to differential equations [45]. The uncertainty knowledge sharing has been also proposed to reduce the uncertainty in a PSE framework [13]. PSE is good at working for sharing the uncertainty knowledge. This idea could contribute also to reduce human errors. Previous example cases would provide good instructive examples for our future activities.

In this section, one realistic method is introduced to reduce the uncertainty in computing science. One of uncertainty sources is the rounding error. In IEEE 754 [46], the four rounding methods are specified: Round to nearest (Even) (RZ), Round Upward (RP), Round Downward (RM) and Round toward 0 (RZ). Not always but in many cases, computation results by vulnerable or sensitive computations are strongly influenced by the rounding methods. Here we call a software containing uncertainty a sensitive software. So we would presume that computational results from the sensitive software may be influenced by the rounding methods.

Based on this consideration, one would use a PSE for automatic program generation to detect a sensitivity of the software to the rounding method. This could be relatively easy in PSE for computer assisted program generation. When a program is generated by the PSEs, for example, the four programs for one problem are generated corresponding to the four rounding methods. The four programs provide independent numerical results for one specific problem. If the PSE also has a comparison function among the results, one can easily find the difference among the results. If the numerical results depend on the rounding methods, one can suspect that the software may have some uncertainty. That means that the software is sensitive against the rounding errors. In this case the software should be run with a higher precision, for example, with the double or quadruple precision further to reduce the uncertainty.

This method is rather generic, and is easily implemented in the PSE framework. Figure 9 shows an example result for $f_1(x) = \left(-ln\left(exp(-x^{-4})\right)\right)^{-1/4}$ and $f_2(x) = x$ [35]. The example in Fig. 9 shows a possibility of detecting the uncertainty by a comparison among the results by the different rounding methods. The result suggests us to employ a higher precision to reduce the numerical error especially for solving $f_1(x) = \left(-ln\left(exp(-x^{-4})\right)\right)^{-1/4}$.

We also applied this suggested method to estimate the numerical errors in a shock wave problem in a fluid. Figure 10 a) shows the result for the shock wave propagation. At the single precision, the numerical results present significant differences depending on the rounding methods (see Fig. 10b). Figure 10c) shows the results at the double precision. The results at the double precision provide the better results. The quadruple precision presents sufficiently accurate results as shown in Fig. 10d).

The PSE assisted simulation program generation would be one of good methods to manage the uncertainty. When we write programs with PSEs, the PSEs would generate the multiple programs with different precisions and different rounding methods; The PSEs would also provide a capability to compare the numerical results among the data obtained by all the programs for a specific problem. This approach is realistic and simple to detect the uncertainty relating to the numerical operations.

4 CONCLUSIONS

The PSE has been extensively explored over the past few decades. The explorations have been supported by the reinforced computer power and algorithm power. PSE will boost up the programming power, and will enrich our e-Life and e-Science. In the near future of the PSE development we should consider how to create a PSE. To build up a PSE is a very hard task and needs huge human efforts. Therefore, PSE researchers have still been meeting this difficulty. In this research issue meta PSE or PSE for PSE may play an important role to build up service-oriented PSEs. One example of the meta PSE is a PSE Park [39], in which many modules, developed by PSE researchers / developers, are distributed. Each module has one function or may be a one PSE, and is developed by many independent researchers and developers. By connecting the modules, PSE researchers or users can construct a single-purpose PSE or so. The interface should be opened, so that each PSE connector can be easily developed by each user or researcher or developer. The module mediator / connector may come into play there. The module base PSE may open a new PSE World.

Another research issue in PSE is validation, validity and uncertainty. When a PSE gives a wrong result for users, it may cause some difficulties, errors and accidents, depending on target problems. The validation and verification mechanisms are essentially important as usual software.

ACKNOWLEDGEMENTS

This work was supported partly by JSPS, MEXT and Japan Society of Computational Engineering and Science. The authors would like to extend their appreciations to Prof. J. Rice, Prof. E. Houstis, friends in IFIP WG2.5, and friends in the PSE research group in Japan

including Prof. S. Hioki, Dr. Y. Miyahara, Prof. D. S. Han, Dr. S. W. Hwang, Prof. T. Teramoto, Dr. H. Usami, Dr. T. Maeda, Dr. Y. Manabe, Dr. H. Kobashi, Dr. Y. Hayase and our former students and friends contributed to the works relating to the paper.

-

REFERENCES

[1] Boonmee, C. and Kawata, S., Computer-Assisted Simulation Environment for Partial-Differential-Equation Problem: 2. Visualization and Steering of Problem Solving Process, *Trans. of the Japan Society for Computational Engineering and Science*, Paper No. 19980002, (1998).

[2] Boonmee, C., Kawata, S., Fujii, S., Manabe, Y. and Tago, Y., Visual Steering of the Simulation Process in a Scientific Numerical Simulation Environment - NCAS -, in print, *Enabling Technologies for Computational Science*, edited by E.Houstis and J.Rice, Kluwer Academic Pub., (2000).

[3] Fujio, H., & Doi, S. (1998). Finite Element Description System as a Mid-Layer of PSE. Proceedings of Conference on Computation Engineering and Science, 3(2), (pp. 441-444).

[4] Fujita, A., Teramoto, T., Nakamura, T., Boonmee, C. and Kawata, S. (2000). Computer-Assisted Parallel Program Generation System P-NCAS from Mathematical Model-Visualization and Steering of Parallel Program Generation Process-. Transaction of the Japan Society for Computational Engineering and Science, Paper No. 20000037.

[5] Gallopoulos, E., Houstis, E. and Rice, J.R., Future Research Directions in Problem Solving Environments for Computational Science, Technical Report CSRD Report No. 1259, *Report of a workshop on Research Direction in Integrating Numerical Analysis, Symbolic Computer, Computational Geometry, and Artificial Intelligence for Computational Science*, Washington DC, April 11-12, (1991).

[6] Gallopoulos, E., Houstis, E. and Rice, J. R. (1994). Computer as Thinker/Doer: Problem-Solving Environments for Computational Science. IEEE Computational Science and Engineering, 1(2), 1-23.

[7] Hirayama, Y., Ishida, J., Ota, T., Igai, M., Kubo, S. and Yamaga, S. (1988). Physical Simulation using Numerical Simulation Tool PSILAB, Proc. *1st Problem Solving Environment Workshop*, pp. 1-7.

[8] Houstis, E. N. and Rice, J. R., Parallel ELLPACK, a development environment and problem solving environment for high performance computing machienes, edited by In P. Gaffney and E. N. Houstis, *Programming Environments for Hihg-Level Scientific Problem Solving*, North-Holland, Amsterdam, (1992), pp. 229-241.

[9] Kawata, S., Boonmee, C., Fujita, A., Nakamura, T., Teramoto, T., Hayase, Y., Manabe, Y., Tago, Y. and Matsumoto, M., *Visual Steering of the Simulation Process in a Scientific Numerical Simulation Environment –NCAS-*, Enabling Technologies for Computational Science, E. N. Houstis and J. Rice (Ed.), Kluwer, (2000), pp. 291-300.

[10] Okochi, T., Konno, C. and Igai, M., High Level Numerical Simulation Language DEQSOL for Parallel Computers. *Trans. of Information Processing Society of Japan*, 35(6), (1994), 977-985.

[11] Umetani, Y., DEQSOL A numerical Simulation Language for Vector/Parallel Processors,

Proc. *IFIP TC2/WG22*, 5, (1985), pp. 147-164.
[12] Rice, J. R. and Boisvert, R. F., Springer Series in Computational Mathematics 2, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, New York, (1984).
[13] Kawata,. S., Computer Assisted Problem Solving Environment (PSE), *Encyclopedia of Information Science and Technology, Third Edition*, ed. Mehdi Khosrow-Pour, Chapter 119, (2014), pp. 1251-1260, IGI Global, Hershey, PA, USA.
[14] Kawata, S., Computer Assisted Parallel Program Generation, *Encyclopedia of Information Science and Technology, Fourth Edition*, ed. Mehdi Khosrow-Pour , Chapter 398, (2017), IGI Global, Hershey, PA, USA.
[15] Kawata, S. Review of PSE (Problem Solving Environment) Study, *J. Convergence Information Tech.*, 5 2010, pp. 204-215.
[16] IFIP WG2.5 (International Federation for Information Processing, Working Group 2.5), IFIP WG2.5 homepage retrieved on June 12, (2017), http://www.ifip.org/wg-2.5
[17] Ford, B. and Chatelin, F. (Ed.), *Problem Solving Environments for Scientific Computing*, North-Holland, (1987).
[18] Gaffney, P. W., and Houstis, E. N. (Ed.), *Programming Environments for High-Level Scientific Problem Solving*, North-Holland, (1992).
[19] Houstis, E. N., Rice, J. R., Gallopoulos, E. and Bramley, R. (Ed.), *Enabling Technologies for Computational Science, Framework, Middleware and Environments*, Kluwer Academic Publishers, (2000).
[20] Gaffney, P. W. and Pool, J. C. T. (Ed.), *Grid-Based Problem Solving Environments*, Springer, (2007).
[21] Dienstfrey, A. and Boisvert, R. (Ed.), *Uncertainty Quantification in Scientific Computing*, Springer, (2012).
[22] JSCES (The Japan Society for Computational Engineering and Science), JSCES homepage retrieved on June 12, (2017), http://www.jsces.org/.
[23] PSE Research Group in Japan, homepage retrieved on June 12, (2017). http://www.jsces.org/activity/research/pse/.
[24] Rice, J. R. and Boisvert, R. F., *Solving Elliptic Problems Using ELLPACK*, Springer Series in Computational Mathematics 2, Sringer, (1984).
[25] Kawata., S., Tago, Y., Umetani, Y. and Minami, K. (Ed.), *PSE Book: Computer assisted Problem Solving Environment in computing science (Basic & Advanced) (in Japanese)*, Tokyo: Baifukan, (2005).
[26] Inaba, M., Fuju, H., Kitamuki, R., Kawata, S. and Kikuchi, T., Computer-Assisted Documentation in a Problem Solving Environment (PSE) for Partial Differential Equation Based Problems, *Trans. of the Japan Society for Computational Engineering and Science*, 20040025, 2004.
[27] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard, *Technical Report*, University of Tennessee, Knoxville, Tennessee, (1994).
[28] Fuju, H., Kawata, S., Sugiura, H., Saitoh, Y., Usami, H., Yamada, M., Miyahara, Y., Kikuchi, T., Kanazawa, H. and Hayase, Y., Scientific Simulation Execution Support on a Closed Distributed Computer Environment, *2nd IEEE International Conference on e-Science and Grid Computing*, (2006), 27340112.
[29] Kanazawa, H., Itou, Y., Yamada, M., Miyahara, Y., Hayase, Y., Kawata, S. and Usami, H.,

Design and Implementation of NAREGI Problem Solving Environment for Large-Scale Science Grid, *2nd IEEE International Conference on e-Science and Grid Computing*, (2006), 27340105.
[30] Globus, Globus Online. Retrieved June 12, (2017), http://www.globus.org/
[31] UNICORE, UNICORE. Retrieved June, (2017), http://www.unicore.eu/
[32] Teramoto, T., Okada, T., and Kawata, S., A Distributed Education-Support PSE System, *3rd IEEE International Conference on e-Science and Grid Computing*, (2007), pp. 516-520.
[33] CMFVVUQ (Committee on Mathematical Foundations of Verification, Validation, and Uncertainty Quantification), *Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification*, The National Academies Press, (2012).
[34] Johnson, C. R., Top Scientific Visualization Research Problems. *IEEE Computer Graphics and Applications: Visualization Viewpoints*, 24(4), (2004), pp. 13-17.
[35] Kahan, W., Desparately Needed remedies for the Undebuggability of Large Floating-point Computations in Science and Engineering, *paper presented at IFIP Working Conference on Uncertainty Quantification in Scientific Computing*, retrieved on June 12, (2017), http://www.eecs.berkeley.edu/~wkahan/Boulder.pdf
[36] Rump, S. M.,Verification methods: Rigorous results using floating point arithmetic. Acta Numerica, 19, (2010), 287-449.
[37] Weimer, W. R., *Exceptional Situation and Program Reliability*, PhD Thesis, University of California, Berkeley, (2005).
[38] Houstis, E. N., Gallopoulos, E., Bramley, E. and Rice, J. R., Problem-Solving Environment in Computational Science. *IEEE Computational Science & Engineering*, 4, (1997), pp. 18-21.
[39] Kawata, S., Kobashi, H., Ishihara, T., Manabe, Y., Matsumoto, M., Barada, D., Hayase, Y., Teramoto, T. and Usami, H., *Scientific Simulation Support Meta-System: PSE Park - with Uncertainty Feature Information* -, International Journal of Intelligent Information Processing, 3, (2012), 66-76.
[40] Teramoto T., Nakamura, T., Kawata, S., Matide, S., Hayasaka, K., Nonaka, H., Sasaki, E. and Sanada, Y., A Distributed Problem Solving Environment (PSE) for Partial Differential Equation Based Problems, *Trans. Jpn. Soc. Comp. Sci. Eng.*, Paper No. 20010018, (2001).
[41] Kawata, S., Usami, H., Hayase, Y., Miyahara, Y., Yamada, M., Fujisaki, M., Numata, Y., Nakamura, S., Ohi, N., Matsumoto, M., Teramoto, T., Inaba, M., Kitamuki, R., Fuju, H., Senda, Y., Tago, Y. and Umetani, Y., A Problem Solving Environment (PSE) for Distributed Computing, *Int. J. High Performance Computing and Network*, Vol. 1, No.4, (2004), pp. 223-230.
[42] Einarsson, B. (Ed.), Accuracy and Reliability in Scientific Computing, *SIAM(Society for Industrial and Applied Mathematics)* (2005).
[43] BEA (Bureau d'Enquêtes et d'Analyses pour la sécurité de l'aviation civile), *Accident to the Airbus A330-203 flight AF 447 on 1st June 2009*, (2011). http://www.bea.aero/fr/enquetes/vol.af.447/point.enquete.af447.27mai2011.de.pdf
[44] GAO (United States General Accounting Office), Patriot Missile Software Problem, (2012). www.fas.org/spp/starwars/gao/im92026.htm
[45] Enright, W. H., *Reducing the Uncertainty When Approximating the Solution of ODEs,* A. Dienstfrey & R. Boisvert (Ed.), Uncertainty Quantification in Scientific Computing, (2012), pp. 280-292, Springer.

[46] IEEE754, (2008). http://grouper.ieee.org/groups/754/index.html