1996

Scalable Scientific Software Libraries and Problem Solving Environments

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
96-001

SCALABLE SCIENTIFIC SOFTWARE LIBRARIES AND PROBLEM SOLVING ENVIRONMENTS

John R. Rice

Purdue University
Department of Computer Sciences
West Lafayette, IN 47907

CSD TR-96-001
January 3, 1996

Scalable Scientific Software Libraries and Problem Solving Environments

Report on a Workshop* held at Purdue University
September 25–27, 1995

John R. Rice
Department of Computer Sciences
Purdue University

January 3, 1996

Abstract

Software libraries provide encapsulated problem solving power and *problem solving environments* (PSEs) give ordinary users painless access to problem solving power. Thus the structure of libraries and the design of PSEs are inextricably linked. This workshop explored the state-of-the-art in these two areas and their interdependence. A few application areas (e.g., linear algebra) have a rather simple structure for a software library and a very widely known language (mathematics) to use as the basis for a PSE (e.g., MATLAB). This simplicity and standard language are missing for most of the important scientific application areas where both the libraries and the PSEs are embryonic. These application areas also have almost unlimited complexity so that high performance computing power is essential. Thus both the PSE design and library structure must be scalable in the complexity of the applications. The workshop focused on partial differential equations (PDE) based applications and closely related areas for its examples.

A. Introduction

There are two different ways to view the scalability of a problem solver. First is that the work to solve a particular problem decreases proportional to the power of the computing resources used. Second is that the work to solve similar problems grows proportionally to the problem size as the size increases. The first view is that of speed up in parallel computing and the second is that of computational complexity using a fixed solver. Both views are important in practical applications.

*This workshop was supported by ARPA under ARO grant DAAH04-94-G-0010 and the NSF under grant CCR95-23243.

A library contains many solvers and is said to be scalable if it contains a set of solvers for a particular problem or problem family which achieve scalability. Thus one may change the solvers as the computer resource power changes or as the problem size changes. Of course, there are many ways to change both computing power and problem size so that libraries (or solvers) may be scalable in some ways and not others. In changing computing resources one usually assumes the changes are balanced in some reasonable way. In changing problem size there is more variability and it is unlikely that a library will be scalable in all possible ways. Thus changing the dimension of physical space in a model, the accuracy required, or the number of physical phenomena in a model can have very different effects. Nevertheless, the goal for a scalable scientific library is that the work required decreases in a direct, maximal way as the computing power increases and that it increases in a known, minimal way as the problem size increases.

A problem solving environment (PSE) is a computer system that provides all the computational facilities necessary to solve efficiently a target class of problems. Moreover, PSEs use the language of the target class of problems, so users can solve them without specialized knowledge of the underlying computer hardware, software or algorithms. The facilities include advanced solutions methods, automatic or semi-automatic selection of solutions methods, and ways to easily incorporate novel solutions methods. They also include facilities to automatically or semi-automatically, select computing machines, to view or assess the correctness of solutions, to check the formulation of the problem posed, and to manage the overall computational process. Overall, PSEs are to be a framework that is all things to all people; they solve simple or complex problems, support rapid prototyping or detailed analysis, and can be used in introductory education or at the frontiers of science.

The primary mode for the (PSEs and their user interfaces) workshop were eight panel discussions which resulted in four reports: (B) PSEs and their user interfaces, (C) Enabling technologies and virtual parallel environments, (D) Scalable libraries, their architecture and PSEs for PDEs, (E) Future research directions. Each of these reports is organized roughly as 1. Definitions, 2. Recent Developments, 3. Open Problems and Barriers to Progress, and 4. Recommendations. This report consists of these four reports plus a final section describing the program, presentations and participants.

These reports were developed almost completely independently and thus there is overlap in many places. Perhaps the frequency of mentioning certain topics (e.g., need for a forum for developers, impact of global network, lack of accepted interface standards and methodologies, need to study architecture) indicates their importance for the future. It is planned that a much shorter workshop report will be written to remove the redundancies and to concentrate on the most significant points.

B. PSEs and Their User Interfaces

Panel:	User Interfaces for PSEs
Members :	Elias Houstis (moderator), Chandrajit Bajaj, Robert Nelson, Granville Sewell
Panel :	Application Specific PSEs
Members :	Lennart Johnsson (moderator), Randall Bramley, Dan Marinescu, Stratis Gallopoulos

Report Authors: Elias Houstis, John Rice

B.1. Definition of Area

The concept of a mathematical software library was introduced in the 1960s to support the reuse of high quality software. In addition special journals, conferences, public domain software repositories (e.g., ACM, Netlib), and commercial libraries (i.e., IMSL, NAG) have been established to support this concept. Similar efforts can be found in engineering software particularly in the areas of structural and fluid mechanics. The increasing number, size and complexity of mathematical software libraries made necessary the development of a classification and indexing of existing and future software modules. Library software is currently organized in terms of the mathematical models it supports. A significant effort in this direction is the GAMS on-line advisor system which has become a standard advisory framework for indexing and classifying mathematical software. Information about engineering software can be found in several handbooks which usually describe the applicability and functionality of existing packages. There is definitely a need to extend GAMS scope to cover available engineering software.

The advances in desktop software/hardware, workstation clustering and distributed computing technologies, and the ease of access to supercomputing facilities have made computational prototyping a cost effective alternative to design new products and to study science and engineering phenomena. Although the software library provides some form of abstraction and a facility for reusing software parts, it still requires a level of expertise beyond the background and skills of the average scientist and engineer who usually is involved in the design of manufactured artifacts. This recognition has lead to the new concept of software reuse, Problem Solving Environment (PSE), defined in the Introduction. The current PSEs consist of small set of modules, usually taken from existing libraries, integrated (packaged) to solve a predefined class of engineering or mathematical problems. Early PSE examples are Macsyma, Mathematica, Maple, ELLPACK, MATLAB and several engineering software systems. Similar evolution has been observed in the pre-processing (CAD, mesh generation) and post-processing (data visualization) tools. These libraries, interfaces and pre- and post-processing tools have increased the abstraction of computational prototyping and allow users with a minimum computational background to prototype complex artifacts. PSEs are distinguished from monolithic systems by the wide domain of problems or applications they can handle; they have built-in flexibility, extensibility, and prototyping facilities. The software architecture of PSEs is characterized by the integration methodology used to connect the software parts involved and the underlying execution model assumed. The common shortcoming of current PSEs is the lack of adequate knowledge based support systems for the applicability, compatibility, and performance (i.e. complexity) of library modules, the selection of user defined parameters, and navigation. Similarly, error estimation is not encoded in the PSE in some reusable form but is assumed to be part of the responsibility of the user. An ideal PSE is one that can make many decisions for the user by consulting its associated knowledge base. The above discussion leads us to a formal definition of a PSE in terms of its autonomous components as follows:

$$\text{PSE} = \text{User interface} + \text{libraries} + \text{knowledge base} + \text{software bus}.$$

B.2. Recent Developments

In the area of scientific PSEs the most important developments are systems like Macsyma, Mathematica, Maple, //ELLPACK, and MATLAB. Similar important developments have occurred in engineering PSEs and pre-processing and post-processing tools. The concept of parallel libraries is maturing, especially in the area of linear algebra (i.e. ScaLaPack and LAPACK). In the area of user interfaces for PSEs, the dominant paradigm is still the GUI which has many shortcomings. Some of their disadvantages include their restriction to highly educated and unimpaired users sitting in front of a desktop computer, the long series of precise manipulations required to complete a task, the single-user view, the need for the user to organize, search, and filter information, the lack of scalability, and high development cost. It was concluded that the GUI paradigm is too restrictive to be the basis for PSE development. Network software and integration standards are recent enabling technologies for PSEs which are very compatible with the flexible software architecture needed.

B.3. Principal Open Problems and Barriers to Progress

The panel discussion raised a number of questions; we list the most important ones. It is clear that the majority of them could be considered as open problems.

- Is there a market for application PSEs? What are the economics of commercial PSE development?
- Are there software engineering methodologies, kernels, CASE tools, and enabling technologies for building PSEs out of reusable parts?
- What is the role of academia, industry, and funding agencies in application PSE development?
- Can PSEs be used for production instead of rough or preliminary prototyping?
- PSEs need the integration of geometric, numeric, symbolic, graphical, high performance (HPC), and AI technologies and infrastructure. Is there a suitable integration software engineering paradigm for scientific PSEs?
- In addition to algorithmic changes, PSEs require costly maintenance and updating due to software/hardware infrastructure changes. Is the concept of PSE network server feasible and desirable? How much does it reduce these costs? What are the added costs of this approach?
- How should decision making be divided between PSEs and users?
- What are the target PSE user communities?
- What should be the relationship between PSE performance and ease-of-use?
- The world is multidisciplinary. How multidisciplinary can the scope of PSEs be?

- Is the GUI interface paradigm too old? Too restrictive? Programmability and customization of problem solving environment is necessary in many applications. Domain specific languages and compilers offer a natural way of communication with the user. What are the new user interface paradigms for PSEs?

- For most important application areas, the underlying solution methods and even problem solving paradigms change rapidly. This means the systems must be "open" in the sense of allowing rapid changes of modules "under the hood". What are the software architectures for PSEs that allow such changes?

- PSEs are supposed to handle wide spectrum of user backgrounds and expectations. What is the appropriate help/tutoring/educational system for PSEs?

- The concept of a library/PSE network server will be the next advance for the reuse of scientific software. What enabling technologies are needed to support network based problem solving?

B.4. Future Software and Computing Paradigms

There are many signs that PC industry sets the vision and standards in user interfaces, software-development tools, operating systems, and software engineering methodologies. These technologies will definitely influence the scientific PSE technology. Thus, it is important to pay attention to the forecasts in areas related to PSE technologies. Ted Nelson (the inventor of hypertext) predicts that

- The future interface won't be the Mac/Windows type or the voice interface seen in Star Trek. Both ideas are inane. The alternative is the design of abstract worlds and spaces with new rules and topologies that people can easily understand, use, and enjoy. Everybody should have the interface they want, so it will be increasingly important for applications to be delivered detached from a specific user interface.

- There will be no more monolithic applications, because future problem solving will be done by software parts tied together, piped together, and scripted together for the problem at hand.

Microsoft is investing heavily to move from separate complex application programs to a concept that Microsoft calls the project-oriented workspace. Users will work in a basic blank document, called a container or binder, that has a series of tools that present themselves when needed.

These predictions fit well with the multidisciplinary problem solving environment (MPSE) concept that allows users to bind PSEs and specify solution agents that solve their problem. Integration concepts like the software bus and execution environments for network computing (meta-computing) will definitely be important for MPSE projects. It is predicted that the next generation of PCs will have multiprocessing capability. Thus, research on heterogeneous models of parallelism and parallel reuse methodologies will be needed. There is no technological justification for the off-line integration of physical and computational prototyping. Research and development in virtual environments is justified to unify these two every day forms of prototyping.

B.5. Recommendations

- Promote cooperation between academic PSE developers and commercial PSE developers/marketers.
- Promote the involvement of application end-users in the PSE design phase. Create forums where users, researchers, and developers can interact during iterations of PSE prototyping and use. Identify alpha and beta testers early in the development process.
- Provide seed money for building example second-generations scientific PSEs.
- Provide funding for experimentation with and development of software integration tools and architectures for scientific PSEs.
- Support the development and refinement of pre- and post-processing tools (symbolic analysis, visualization, etc.).
- Develop large scale knowledge bases to support a few application areas. Develop and test methodologies to use knowledge bases for intelligent PSE support systems.

C. Enabling Technologies and Virtual Parallel Environments

Panel:	Enabling Technologies
Members:	Ronald Boisvert (moderator), Faisal Saied, Paul Wang, Sanjiva Weerawarana
Panel:	Virtual Parallel Environments and Languages
Members:	Geoffrey Fox (moderator), Andrew Sherman, Calvin Ribbens, Manish Parashar
Report Authors:	Ronald Boisvert, Geoffrey Fox, Paul Wang, Sanjiva Weerawerana

C.1. Definition of Area

Problem Solving Environments were identified as open and extensible, multi-disciplinary, hierarchical or multilevel, systems of systems. Although the standard architecture of a PSE was unclear, it was agreed that a PSE can be viewed as a set of component systems (or sub-PSE's) such as compute servers, a control kernel, algorithm servers and libraries, knowledge bases or intelligence, a user interface, utilities and tools, help and documentation support, etc. Each subsystem has a well defined interface specification, and a suite of middle-ware ("glue") links the individual components. The two combined panels addressed the overall environment for constructing PSE's in this context, as well as the set of tools with which to do this.

Virtual parallel environments (VPEs) and languages (VPLs) are methodologies for hiding the complexities of the underlying parallel architecture/paradigm and for making parallel computing "easier" and more natural. These ideas are achieved by using machine-independent programming models that are simpler to use than the model provided by the hardware. The result of using virtual parallel environments and libraries is the development of software that is portable to a wide range of hardware platforms. VPE's and VPL's include both high level languages of general and domain specific type as well as compiled and scripted middle-ware. VPE's and VPL's are built of (software) enabling technologies such as HPF, MPI and Java.

More generally, enabling technologies (ET) for PSEs are key technologies that make constructing, prototyping, realizing, testing, maintaining, and evolving PSEs possible, easier, simpler, or faster. Due to the all-encompassing nature of PSEs, the set of technologies that enable PSEs is wide and varied. Note that we need both general technologies which can be applied to all PSE's as well as those that are specific to one area. As the overall computer science issues pertaining to PSE's are not yet well established, it is not easy to give a clean classification of technologies, and indeed interpreting PSEs broadly implies that essentially all technologies are relevant to PSE's for some application! Some of the enabling technologies highlighted by the panel are listed below:

- Collaborative computing technology
- Configuration control and human-in-the-loop (Computational steering)
- Computational geometry and grid generation
- Scalable algorithms
- Scalable solver libraries
- Parallel/Distributed computing — metacomputing
- Fault tolerance and security
- Federated multi-media databases
- File system and I/O technologies
- Visualization including virtual reality, televirtuality etc.
- Interactive interface development (GUI) technologies
- Symbolic manipulation and automatic code generation
- Artificial intelligence and expert systems
- Performance monitoring and modeling
- "Low-level" virtual machine such as MPI, PVM etc.
- "Fine grain high-level" languages (C++, HPF etc.)
- Software engineering and coarse grain software (software bus) integration
- "Web-ware" and scripting middle-ware(Perl, Java, VRML, Python, etc.)
- Agent search and communication systems
- Wrapper technology for legacy systems and interoperability

- Interface specification support and information exchange protocols (such as CORBA, Opendoc, metadata and web standards)
- Object oriented software technology - object transport and management
- PSE templates and frameworks

C.2. Important Recent Developments

Several factors have contributed to the blossoming of interest in problem solving environments. For instance, within the manufacturing sector there is overwhelming pressure both to decrease time to market and increase the affordability of products. However it has become clear that one cannot achieve this by, say, just porting CFD codes to parallel machines, as manufacturing improvements require linkage of conceptual, preliminary and detailed design, manufacturing process and life-cycle maintenance. One must integrate multi-fidelity simulations from many disciplines with CAD and other databases, and clearly a PSE is the only approach to such a complex system of systems. In a recent study, it was estimated that some 10,000 separate programs will be run during the design of next generation aircraft and this will involve engineers around the world collaborating on this project. This emphasizes the need for a multi-purpose PSE with support for software and human integration with databases, simulation as well as the need to support multi-language and legacy codes. We also believe that much could be gained if several of the current Grand Challenge and other major application efforts could be structured as projects developing component modules of a few scientific and engineering PSE's. With the current focussed independent structure, it is hard to link the interesting technologies (e.g., libraries and productivity tools) produced by the different groups.

Outside the scientific computing community, modest scale problem solving environments have become quite common; Microsoft Office and TurboTax/Quicken are examples. A variety of systems, such as MATLAB, Mathematica, and Maple, are leading a paradigm shift from low-level linkage to high-level interfaces in the scientific computing domain as well. The availability of such highly interactive computing environments has increased our expectations of the scope, power and ease-of-use of computing systems, and the emergence of powerful scripting languages and GUI development tools such as Tcl/Tk have made it reasonable to develop such environments for even the smallest application.

Extending such environments to complex simulation modeling will continue to require use of the most powerful parallel computers available. If such devices are to provide an effective platform for PSEs, however, new levels of usability and stability must be attained. Recent community efforts to develop and propagate standards such as High Performance Fortran, PVM, and MPI are first steps in the direction of providing effective virtual parallel environments. Prototypes of scalable libraries for a few narrow problem sets, which will require firm system software foundations such as these, are beginning to emerge. Prototype PSEs, such as the //ELLPACK project at Purdue, have demonstrated the feasibility of creating PSEs for complex problem domains such as the solution of partial differential equations on a foundation of parallel computation.

At the same time, high-speed computer networks are interconnecting computer systems on a worldwide scale, where concepts such as the World Wide Web are expanding perceptions of our

effective computing environment. Web browsers, such as Mosaic and Netscape, are demonstrating the great power and utility of universal network clients in a rich environment of information servers. Substantial applications such as federated databases have demonstrated the utility of client-server computing, and have renewed the interest in coarse-grain distributed parallel computing in science and engineering. Clustered (workstations or PC's) computing is viewed by many as a more attractive (or perhaps practical is a better term) implementation of high performance computing than tightly coupled MPP's.

Developments such as these are contributing to a new vision for scientific computation. By combining and integrating multiple computing paradigms (e.g. symbolic manipulation, numeric computing, graphics visualization, database management, document preparation, parallel/distributed processing, Web-based retrieval) a PSE can become much larger than the sum of its parts and will bring the power, functionality, and convenience of scientific computing systems to a new level.

C.3. Principal Open Problems, Barriers to Progress and Promising Research Directions

Probably the most central problem in PSE's and the VPL/VPE/ET's for them is the lack of a general understanding of their needs and their architecture. This handicaps both the development of PSE's themselves and their needed technologies. Hopefully this workshop will partially address this point. But more generally, further research on the overall architecture and the corresponding appropriate integration technologies would be very fruitful.

There are some interesting lessons we can learn from the current HPCC program. It has developed many important ET's in both the algorithm and software areas. However let us note the slow development of HPF and HPC++ compilers; they are still research prototypes instead of robust commercial "products". Here the basic research was very successful and we "know" how to build very good high level compilers with excellent support tools. However the commercial vendors are putting modest effort into products as they do not see a compelling business case for a larger investment. This is both a challenge and an opportunity for PSE's. The challenge is to avoid this circumstance and to make to certain that PSE's build on supportable technologies and are themselves supportable. On the other hand note that, for instance, a PSE for integrated design and manufacturing would have a better business case than a HPF code for CFD. Thus focusing on PSE's rather than their component ET's is a useful contribution to the development of the necessary support base for high performance computing and communication enabling technologies.

In the ET's for the programming environment, we naturally see more maturity in message passing than in high level compiler areas as the latter is much more complex. However, the needs of PSEs emphasize the importance of integration of the several different programming paradigms and their implementation on heterogeneous distributed metacomputers. Here we see an interesting contrast with the closely coupled MPP hardware trend to shared memory support, and the need of PSE's for more sophisticated support of the distributed memory model with, in fact, complication from the use of heterogeneous nodes.

More generally, we see a need to integrate issues and expertise between the distributed and parallel computing communities. Some of the work on metacomputing and distributed shared memory illustrates this but we need to address it more broadly. Parallel computing has taught us

much about synchronization and decomposition which needs now to be implemented in a distributed heterogeneous fashion. Typical problems include — providing a shared memory model of the world wide metacomputer; developing new algorithms tolerant of the latency in geographically distributed systems; taking the promising research in scalable I/O for scientific problems and implementing it in a database-dominated distributed environment; efficient implementation of MPI with interoperable ATM networks using switches supporting ATM adaptation (AAL) layers developed outside the parallel computing community.

Both in the PC and Web arenas we see an increasing realization of the importance of middleware. This represents the sum total of all infrastructure between the operating system and the application PSE. Building PSEs on layers of accepted middleware will result in interchangeable components as well as more reliable software, as the middleware layer provides an environment which reduces significantly the amount of software needed to be developed for a PSE. Since a natural part of middleware is scripted or interpreted languages such as Perl or Visual Basic, there are some interesting research issues regarding the linking of compiled and interpreted environments and, more ambitiously, on the optimization of interpreted scripts. This interplay between compiled and interpreted languages is very important in linking coarse grain software integration (e.g., "software buses", adding task parallel constructs to Fortran/C++) to the traditional fine grain languages. Up to now, there has been much emphasis on a single solution which unifies task and data parallelism, but a set of interoperable paradigms is probably more realistic. Further research in this area would be very important. Another component of this research would be the integration of "little languages" focused on a single domain (naturally interpreted) with more general compiled and/or interpreted systems. There are important trade-offs between efficiency and functionality here.

Middleware is often associated with the "sweet spots" in a layered design. Here we use an hour glass model, attempting to define crucial common interfaces at (the several) necks where a thin layer of middleware can efficiently link lots of products (here PSE's in multiple domains) with lots of enabling technologies. Promising research directions are in the identification of such necks of the hour glass and quantification of the associated public domain interfaces.

Another important problem is to look into techniques such as object orientation and program interface specifications for developing reusable, evolutionary software. Software evolution is an important problem as every piece of software will need to evolve over time as new algorithms/techniques are developed. The overall architecture of PSEs and PSE middleware must support evolution well. Returning to our multi-disciplinary manufacturing, we note that PSE's must cope with a myriad (10,000 were mentioned above) of existing codes and so it is essential to develop good wrapper technology to allow the integration of existing codes into new PSE's.

In spite of many efforts in the operating systems and programming languages communities, connecting heterogeneous software components remains a problem. Many systems have been developed for interconnecting software components (OpenDoc, OLE, CORBA, ILU, Glish, PolyLith, ...), but most ignore the issue of interconnection to existing (legacy) systems. While it is convenient to assume that all components of some system can be built with a specific model of interconnection, that is not practical for building PSE prototypes. Further research and development on techniques for interconnecting heterogeneous software components is much needed. Note that such a modular

approach is essential in linking commercial and academic software components. A potential barrier, but also an opportunity, is seen in this area with Web technology. Here there is so much exciting commercial work on both products and (ad-hoc) standards that it is not so easy to see how to mesh in university activities with a longer term focus. These must use many of today's software modules for their viability, however. They must also respect today's realities, with standards being developed with strong commercial and near term focus.

Connections allow heterogeneous PSE components to talk to each other and exchange data. But knowledge of the format and meaning of the exchanged data must be programmed into each component. The lack of community standards for data exchange makes communications ad hoc and components hard to reuse. Work has just started in the MP and OpenMath projects that address standards for mathematical data exchange. Other languages have been developed for general knowledge and physical object exchange (e.g., KQML and VRML). A common mechanism for exchanging mathematical and scientific data, that can be adopted by new and existing systems alike, would go a long way towards removing hurdles for building distributed PSEs. More work is needed in this direction, and in providing interoperability among different data exchange "standards". A good example here is the possible synergy between VRML developed by the Web community and the STEP/PDES product specifications coming from the manufacturing industry. If these were combined, one can imagine the same definition used by the CAD database in the design of a product and by potential customers using a PC based VRML viewer in a virtual "test-drive" before purchasing a new product.

Symbolic code generation has been proven effective and advantageous in various aspects of a PSE (e.g., preparing custom numeric codes, intelligent plotting and visualization, high-level code parallelization). Availability of flexible, automatic, parallel code generation tools will enable PSE developers to use symbolic code derivation and generation in problem solving.

Dealing with scale has always been a problem with high level software environments. Enabling technologies for PSEs must be designed to deal with multifidelity analysis ranging from quick, short analyses to detailed, time-consuming (parallel) analyses as well as widely differing classes of uses and users. Much work is needed to develop an understanding of these issues and to develop software frameworks for solving them.

Intelligence is now widely regarded as an essential feature in complex PSEs. However, the current state-of-the-art of knowledge-based system frameworks for mathematical computing is very low-level and far from appropriate for building PSEs. More work is needed in developing appropriate knowledge formats, ontologies, exchange protocols and databases of meta-data.

The unavailability of various important PSE components such as reusable numeric, symbolic, or geometry libraries/servers/systems is hindering progress towards building prototype PSEs.

The most important barrier to progressing with many of the problems listed above is cost. Experience has shown that it takes many person-years to build even a small scale PSE with the current state of infrastructure. On the other hand, the lack of experience in building PSEs is a barrier to realizing better and more appropriate PSE infrastructure so that the cost of building PSEs may be reduced. This cycle must be broken if PSEs and scalable libraries are to deliver on the promise of providing interesting answers to important problems in a reasonable time. An essential feature of any solution to this difficulty must lie in a component-oriented approach where

individual groups do not build a full PSE, but rather sub-PSE's or ET's for a PSE that is then incorporated into larger systems which have enough infrastructure and capability to be interesting for users. This goal illustrates the critical importance of common standards and consensus on the overall architecture of PSE's.

C.4. Recommendations

The following summarize the major overall recommendations of the combined panel.

- PSEs represent a promising new methodology for computational science and engineering, and their development should be encouraged and supported.
- Of critical importance to the further development of PSEs is continued research in PSE frameworks, architectures, and enabling technologies.
- Commercialization of PSEs is critical to their sustainability. However, since component technologies are moving so rapidly, and industry commitments are increasingly short term, it is unrealistic to expect that such systems will be developed in the commercial sector. As a result, there is great need for a long term government-sponsored research agenda to advance development of these systems.
- Efforts to build prototype PSEs must be encouraged, both to learn what middleware is needed as well as to demonstrate the utility of newly developed middleware. Such efforts should be multidisciplinary, drawing on experts in human-computer interfaces, artificial intelligence, database management, parallel processing and networking, as well as experts in the particular application domain of the PSE.
- A paradigm shift in scientific software development is needed, changing its emphasis from bricks to glue. New methodologies must be developed to allow independently developed components to be easily combined to cooperate in the solution of common problems. Of crucial importance here are standards for interfaces and for data exchange.
- Open architectures that promote the development of reusable modules are needed. These modules should be designed to provide a quantifiable fidelity level (through the use of performance models, for example) so that control and feedback by and from users or their agents is possible.
- The use of problem and object abstraction must be further exploited in scientific software design to aid in the usability, portability, and maintainability of components.
- Particular attention is needed in critical component technologies, such as computational geometry, grid generation, and sparse matrix methods; in each case new, more effective algorithms are needed, as well as community standards for core functions, data representation and exchange.

- The PSE development community is not coherent; indeed, many researchers developing application-specific PSEs are unaware of basic PSE concepts. Thus, there is a great need for forums in which PSE developers in all domains can exchange case histories, design concepts, infrastructure and components.

D. Scalable Libraries and Use for PDE Solvers

Panel: Architecture of Scalable Libraries
Members: Michael Heath (moderator), Roger Grimes, Richard Sincovec, Anthony Skjellum

Panel: Characteristics and Components of PDE Libraries
Members: John Rice (moderator), James Demmel, Lutz Grosz, William Mitchell

Panel: How to Achieve Scalability in PSEs for PDEs
Members: Ahmed Sameh (moderator), Wayne Joubert, John Rice, Barry Smith
Report Authors: John Rice, Ahmed Sameh

D.1. Definition of Area

This group considered three issues that are highly overlapping and common to a large part of scientific computing. The issues are:

D.1.1. How does one structure libraries of PDE solvers?

There are many traditional and well defined components of the PDE solving process, e.g., matrix/vector operations, solution of linear systems, discretization of differential operators, discretization of geometric domains. For some of these components there are standards and widely distributed software (e.g., the BLAS, LINPACK, LAPACK). For other components there are no standards (e.g., discretization of operators), multiple and incompatible standards (e.g., discrete representations of geometry), or multiple and compatible standards (e.g., sparse matrices). By compatible we mean there is a well defined, accurate process to transform the representation of an object from one standard to another. By incompatible we mean that the representations might be only approximately equivalent and an accurate transformation is either impossible or prohibitively expensive.

It is intuitive that there is some natural structure to the PDE solving process which must be expressed in the library structure. However, this structure has not yet been defined in a complete way. Further, the existence of very similar but not equivalent components introduces a certain ambiguity or fuzziness into the library structure.

D.1.2. What is a scalable library?

The description is given in the Introduction.

D.1.3. How does one achieve scalability for PDE solvers?

There is a myriad of techniques to achieve scalability of various types in solving PDE problems. The issue here is not to perfect or analyze these techniques but rather how to assure that the important techniques are usable within the solvers built from the components of a PDE solver library.

D.2. Important Recent Developments

The solutions of PDEs is an extremely active research area, even the subarea of practical computational methods generates journals full of papers every month. There are three recent developments at a higher level that impact this area in a general way. Each of these is a new technology advance available for application to this area.

D.2.1. High performance computing

The growth of computing power has had a steady exponential increase for decades. This increase has finally reached the point where enough power is available to ordinary scientists so that they can, given appropriate software, solve PDE problems quickly that model accurately a wide range of important, complex physical situations. This increase will continue for some years to come, probably for one or two decades.

D.2.2. Object-oriented software engineering

This technology evolves naturally from the early ideas of modularity and encapsulation of functionality. The advances that have occurred recently are (i) a general awareness of the value of the approach, (ii) a general awareness that objects need not be tied to specific languages or systems (i.e., modularity occurs at a higher level of abstraction). (iii) general availability of software tools to support modularity, and (iv) enough computing power is available that the inherent extra overhead costs of interfacing abstract modules are affordable in most cases. If extreme efficiency is required then, for a particular problem and computing environment, one can optimize the code (much as a compiler does) to remove the code that only provides generality.

D.2.3. Problem solving environments

This technology evolves naturally from the view that end-user software systems must adapt to the user's needs and expertise; that the complexity of machines, algorithms, and software should not be visible to the user. The practicality of this technology is also due to the growth in high performance computing power. The problem solving environment (PSE) concept is most evident in PC applications but there is no doubt that it is feasible and needed for scientific and engineering applications.

D.3. Principal Open Problems

The workshop identified several open subproblems encountered in defining and achieving this library structure. These are listed below. They range from basic scientific problems to organizational ones.

These technology advances make it practical to build the much larger and more powerful software systems needed to analyze and control complex physical phenomena. These systems require a well organized library structure for PDE computations. This structure must encompass the very basic software components (e.g., a BLAS-1 routine) as well as the very complex PDE solvers (e.g., a 3-D Navier-Stokes solver with general geometry). Thus the overall open problem is how to define and implement this structure.

D.3.1. How is the complexity managed?

The high complexity of the software modules and library is directly derived from the complexity of the physical world. We do not even know yet about many complexities that must one day be handled by PDE solving systems.

D.3.2. How is complex geometry managed?

A large part of real world complexity is due to geometric complexity. This problem is further complicated by the fact that there exist several incompatible approximate ways to represent geometry (e.g., CSG=Computational Solid Geometry, triangulations, boundary representations, spline-like methods, pixel-like methods). While all computation involves approximations, those in geometry are much less effective than in numerical, symbolic or logical computing. An obviously simple geometric shape may require many megabytes of data for some of these representations. To change from one representation to another might require a major computation (more than that of solving a PDE on the domain) by an amazingly complex program.

D.3.3. How is robustness and reliability measured? achieved? assured? tested?

It is well known that PDE solvers cannot be proved correct in any mathematical sense (at least with our current understanding) but one must be able to provide high levels of confidence for reasonable PDE problems. One must have a methodology that increases the level of confidence systematically by expending more computing effort.

D.3.4. How is parallel and distributed computing handled?

The current tendency is to have completely new algorithms and libraries for parallel machines, even different libraries for each machine architecture. This is expensive to manage and it is not clear that a completely different library structure is needed for each architecture. The hope is that one structure can accommodate all parallel/distributed architectures.

D.3.5. How are polyalgorithms, expert systems, smart software, etc., involved?

One might assume that each library component is independently responsible for "being smart" without affecting the library structure. But then it might be that a component does not have access to the information it needs to make decisions. It might be more effective to create library components whose function is to collect, measure, and provide information during the computation.

D.3.6. How is agreement reached on a common infrastructure and interface standards?

It is clear that the future development of PDE software depends on adopting various standards (or pre-standards, community standards, etc.). Only a very, very rich organization could take on building an effective PDE library from scratch.

D.3.7. How is the "plug-and-play" paradigm achieved? controlled?

Plug-and-play means that one can take a new algorithm, a new representation, a new implementation, etc., and exchange it for an existing library component to see how it performs. A common infrastructure does not automatically guarantee that plug-and-play is easy even though it is essential. The evolution of PDE solving technology seems to demand this capability, otherwise it becomes too expensive to test new ideas and approaches – and too expensive to put them into operational PDE solving systems. The control issue arises because standards can be (and often have been) used as a mechanism to exclude certain groups or methodologies.

D.3.8. What is the relationship between PDE library standards and sparse matrix standards? and the sparse BLAS?

The solution of PDEs is probably the source of the bulk of the sparse matrices appearing in computing. Yet the matrices in PDE problems have very special properties (e.g., a direct connection to geometry) that may be irrelevant in other applications. It is important to know whether the existing approaches to sparse matrix standards naturally accommodate the special needs of PDE solving systems.

D.3.9. How are "Industrial strength" libraries achieved?

Libraries are more than just a collection. Useful scientific software libraries require long term, expert, and expensive support to assure that: errors are fixed, use is documented, contents are kept up-to-date, the library is reasonably complete, general access is provided, and the software is robust/reliable. New ideas and codes tend to come from research groups whose interests do not include most of these activities. Library structure and interface standards from academia might be designed for elegance instead of for practical use.

D.3.10. Who should be involved in creating library standards?

It is clear that the standards should involve people who are expert in PDE software. It probably should also involve some "consumer testing". But it is less clear just who the experts are and who the consumers are. The experts in PDE solving are spread over many disciplines (e.g., numerical analysis, structural engineering, fluid dynamics, nuclear power engineering, electromagnetics) that are somewhat isolated from one another. The consumer might be an engineer designing a bridge, or a builder of a PSE for bridge design, or a group writing a new stress analysis code, or a programmer implementing the linear equation solver for a stress analysis code. The point is that almost everyone

works primarily at a certain level in the natural heirarchial structure of solving PDEs. Thus almost everyone is both a consumer and a producer.

D.4. Recommendations

A wide range of potential actions were discussed in the workshop. Many of them were narrowly focused or were about specific PDE solving methods. There were also recommendations discussed which have no immediate associated action. Examples of these are: (1) matrix free methods must be accommodated, (2) we should take both top-down and bottom-up views, (3) try not to insult dusty desk owners, (4) the design methodology should exploit the nature of the PDE area, (5) use good software engineering practices, (6) support multiple languages.

There were nine recommendations focused on the specific workshop issues given above. These start with "collect people and data" and end with "define the desired library structure".

D.4.1. Identify a broad set of people interested in PDE library standards.

It is important both to have a broad range of expertise and to have a large group of people who might eventually support the standards.

D.4.2. Catalog the contents and structure of existing PDE solving libraries and systems.

These data provide the basis for both a bottom-up and a top-down view of the PDE solving process.

D.4.3. Catalog the methods and software approaches used in parallel computing which are applicable to PDE solving systems.

These data provide the basis for deciding how to accommodate parallel/distributed computing into a PDE library.

D.4.4. Define the relevant characteristics of PDE problems, library components, systems and software tools.

These definitions provide a common vocabulary for discussing the problems. Eventually, many of them will be parts of the terminology used in defining the library structure.

D.4.5. Define the library structure in an abstract way.

This introduces more vocabulary and a framework for the next action.

D.4.6. Define the library structure and interface standards.

This definition is to be rather complete and precise. Yet it should be at as high a level of abstraction as is consistent with clarity. Since the structure will have multiple levels, this definition might be

bound to particular languages at the lower levels and not at the higher levels. It is envisaged that this action will be broken into a number of parts addressed by various groups.

D.4.7. Define a set of tools and environments for building and managing PDE libraries.

One hopes that the tools and environments will be implemented by someone. However, just the process of making these definitions should help clarify the issues for PDE library design and construction.

D.4.8. Define measures of performance for the library components.

It is unrealistic to hope that high performance computers will allow us to ignore performance issues. There will always be needs to know (estimate) the amount of computer time, memory, accuracy, bandwidth, etc., associated with library components.

D.4.9. Define a testbed for the design and evaluation of PDE library components.

Eventually PSEs should provide such testbeds as a by-product of their "plug-and-play" design. But perhaps it is possible to create something much less complicated (and implemented sooner) for the specific purpose of evaluating PDE library components.

E. Future Directions and Research Thrusts

Panel: Future Directions and Research Thrusts
Members: Ronald Boisvert, Geoffrey Fox, Elias Houstis, Lennart Johnsson, Robert Lucas, Ahmed Sameh
Report Authors: Ronald Boisvert, John Rice

E.1. TRENDS

Three trends are transforming the landscape for scientific computation.

E.1.1. Increased need and capability for realistic mathematical modeling and simulation

The use of computational modeling provides distinct advantages in modern manufacturing design, which is increasingly driven by the need for high flexibility and rapid product development. In this environment rapid computational prototyping presents a grand challenge of critical importance. Advances in computing hardware, along with continual improvements in algorithms and software, have enabled the much more widespread use of such techniques in science and engineering. In mature areas quite realistic models are beginning to be developed within the research community, and many of these efforts are multidisciplinary in nature. Clustered computing systems based upon

commodity RISC processors are supplanting MPPs as the platform of choice for such work; this has lead to a re-emergence of interest in coarse-grained parallelism.

E.1.2. The emergence of flexible and usable computing environments

A wide variety of useful computational tools have emerged in the last decade, including extensive numerical libraries, symbolic computing engines, tools for parallel computation and visualization, database management and expert systems. Products like MATLAB and Mathematica, which have combined several such tools into an integrated environment enhanced with sophisticated graphical user interfaces (GUIs), have begun to increase the expectations of those practicing scientific computing. As a result, many application-oriented research teams have begun to use tools such as these to construct focused environments for problem-solving within their own application domains. This shows promise for new levels of usability for scientific problem-solving systems.

E.1.3. The deployment of global communications networks

Global networks are changing our view of the boundaries of our computing environment. The World Wide Web has been a key enabling concept, built on a simple software architecture — TCP/IP based communications — and sustained with tools such as browsers, servers, and protocols.

These trends, which we expect to continue, will converge to a new paradigm for scientific computation: high fidelity, multidisciplinary, distributed modeling and simulation.

Future mathematical modeling will be done by bringing a wide variety of components to bear on a given problem. High fidelity refers to new levels of accuracy and reliability in these components and in the computational models themselves. Complex, but more realistic, models will be built up by combining coarse-grain components spanning a wide variety of disciplines. These heterogeneous components will be dynamically linked together over the global network, where they will be exercised remotely. In this way the network itself plays the role of a huge metacomputer. In such an environment new and improved model components are developed by research laboratories and made available for use on an experimental basis. Stable and trusted components are marketed by value-added service providers. New and highly sophisticated problem-solving environments (PSEs) will enable modelers to marshal this diverse array of tools to develop, exercise, analyze and manage their modeling and simulation efforts. PSEs will exist for rapid prototyping, as well as for detailed systematic analyses. Virtual environments to support hybrid (computational/experimental) analysis and prototyping will also emerge.

E.2. Barriers

Many difficulties need to be overcome before such a vision can be realized. The first is the current inability to sustain high-performance software development, even for the most basic mathematical components, on a volatile hardware base. This has made it difficult to translate hardware advances into better mathematical models. Many fundamental issues still remain in the software engineering of scalable libraries. As a result, few credible examples have emerged, and those that have are very narrowly focused. We do not yet have a viable infrastructure on which to build new high performance modeling and simulation systems.

The second barrier is the high cost of development for flexible and highly capable problem solving environments. This is not due just to the complexity of usable, portable GUI development. It still remains quite difficult to add even rudimentary "intelligence" to these systems. Early efforts in the integration of coarse-grain components have been ad-hoc in nature, as have been techniques for the distribution of such components over the network. Finally, transforming the World Wide Web from a loose federation of networks, protocols and volunteerism into a global problem-solving environment will require levels of cooperation, scalability, and interoperability far beyond today's levels.

E.3. A Research and Development Agenda

Overcoming these barriers will require the building of a new community of mathematicians, computer scientists, and application engineers who share the goal of shaping the next generation of complex scientific problem solving environments. This community must foster these developments from the initial raw, partial, academic prototypes to test ideas to the production of "industrial strength" systems for commercial markets. Thus the group must be diverse in its areas of expertise, in the scale of software systems represented, and in the level of involvement with finished software products. Among the research directions of this community are the following.

E.3.1. Developing an infrastructure for sustainable high performance software development

New techniques, standards and tools for constructing scientific software based on the use of a hierarchy of levels of abstraction will be necessary. This is one of the tenets of object-oriented approaches to software development, an approach which is only just beginning to affect scientific software. Of prime importance are techniques which enhance reusability, reliability, robustness, as well as performance portability. Also crucial are mechanisms for validating the correctness of numerical software. Without these features successful commercialization of high performance scientific computing tools cannot occur. The research community must lead the way in developing a broad-based framework in which to build scalable and maintainable component libraries for a wide range of core problems.

E.3.2. Developing an infrastructure for cost-effective development of complex heterogeneous scientific problem solving environments

New emphasis must be placed on research in technologies which enable development of the next generation of scientific PSEs. In particular, the identification of "middleware" which can provide a common substrate for PSE development should be a high priority. Examples are open distributed object management systems which promote coarse-grain software component integration. On a more mundane level, standards for the exchange of scientific data to promote the interoperability of components must be pursued in a wide variety of application domains. Examples are interface standards for pre- and post-processing in PDE solving environments. Systems must be built in such a way that they can be easily composed to obtain new systems of systems.

User interface development will be simplified with continued development of universal Web-based GUI clients with features such as embedded scripting languages and support for executable content. Modern database systems and object technology should be adapted and used for managing the enormous volumes of data that scientific problem solving systems will generate.

Knowledge-based systems with particular domain expertise must be devised to aid the use of complex systems such as these. For example, such systems could help select among solvers, data structures, and network resources to maximize performance. Detailed performance models must be developed for software as well as for the underlying communications and computing fabric to provide the basis for such reasoning systems.

Existing static network-based information repositories need to evolve into highly interoperable problem-solving service providers. Repositories for high level scalable computational models for a variety of disciplines need to emerge. Standardized meta-data for collections and services to aid in search and discovery among repositories will become increasingly important.

E.3.3. Developing working prototypes

Infrastructure development cannot occur in a vacuum. Extensive experimentation will be necessary to determine what is feasible and to demonstrate the merit of new approaches. Infrastructural components must be shown to be practical and cost-effective so that commercial support is viable. Without this such new technologies will not be sustainable.

Library software efforts centered about new development frameworks need to occur. These efforts must extend beyond dense linear algebra to large sparse matrix computations, including both direct and parallel preconditioned iterative approaches, where standardization efforts are needed. Another critical area involves efficient and effective methods for computing with and about geometric objects. Current multiple incompatible, and overly complex approaches must be replaced by new techniques which lead to both natural manipulation and high precision. Development of compatible mesh generation techniques and tools for complex three-dimensional objects is also critical.

Experiments with heterogeneous coarse-grain client/server-based simulation and modeling must begin. Fully functional application-oriented PSEs must be constructed. With these as a basis, prototype distributed PSEs for multidisciplinary design optimization can be realized in which the enabling tools are application-specific PSEs.

F. Workshop Organization

Organizers: Elias N. Houstis and John R. Rice

Sponsors: Advanced Research Projects Agency
National Science Foundation

Panels

1. *Application specific PSEs*

Moderator: Lennart Johnsson
Panelists: Randall Bramley, Dan Marinescu, Stratis Gallopoulos

2. *User Interfaces for PSEs*
   Moderator: Elias Houstis
   Panelists: Granville Sewell, Robert Nelson, Chandrajit Bajaj

3. *Enabling Technologies for PSEs*
   Moderator: Ronald Boisvert
   Panelists: Faisal Saied, Paul Wang, Sanjiva Weerawarana

4. *Virtual Parallel Environments and Languages*
   Moderator: Geoffrey Fox
   Panelists: Andrew Sherman, Cal Ribbens, Manish Parashar

5. *Architecture of Scalable Libraries*
   Moderator: Mike Heath
   Panelists: Roger Grimes, Richard Sincovec, Tony Skjellum

6. *Characteristics and Components of PDE Libraries*
   Moderator: John Rice
   Panelists: Jim Demmel, Lutz Grosz, William Mitchell

7. *How to Achieve Scalability in PSEs for PDEs*
   Moderator: Ahmed Sameh
   Panelists: Wayne Joubert, Barry Smith, John Rice

8. *Future Directions and Research Thrusts*
   Moderator: Robert Lucas
   Panelists: Lennart Johnsson, Elias Houstis, Ronald Boisvert, Geoffrey Fox, Ahmed Sameh

Presentations

1. *Overview of Problem Solving Environments*
   John R. Rice, Purdue University
2. *PDEase*
   Robert Nelson, MACSYMA Corp.
3. *Scalable Libraries*
   James Demmel, University of California–Berkeley

Participants

Bajaj, Chandrajit	Purdue University
Bangalore, Purushotham	Mississippi State University
Boisvert, Ronald	NIST
Bordner, James	University of Illinois
Bramley, Randall	Indiana University
Char, Bruce	Drexel University
Demmel, James	University of California, Berkeley
Fox, Geoffrey	Syracuse University
Gallopoulos, Stratis	University of Illinois
Gray, Simon	Kent State University
Grimes, Roger	Boeing
Grosz, Lutz	University of Karlsruhe
Harrand, Vincent	CFD Research Corp.
Heath, Michael	University of Illinois
Houstis, Elias	Purdue University
Jamieson, Leah	Purdue University
Jin, Li	Mississippi State University
Johnsson, Lennart	University of Houston and Harvard
Joshi, Anupam	Purdue University
Joubert, Wayne	Los Alamos National Laboratory
Lu, Ziyang	Mississippi State University
Lucas, Robert	ARPA
Luke, Edward	Mississippi State University
Marinescu, Dan	Purdue University
Masso, Joan	University of Illinois
McInnes, Lois	Argonne National Laboratory
Mitchell, William	NIST
Nelson, Robert	SPDE, Inc and Macsyma
Parashar, Manish	University of Texas at Austin
Ribbens, Calvin	Virginia Tech
Rice, John	Purdue University
Saied, Faisal	University of Illinois
Sameh, Ahmed	University of Minnesota
Sewell, Granville	University of Texas at El Paso
Sherman, Andrew	Scientific Computing Assoc., Inc.
Sincovec, Richard	Oak Ridge National Laboratory
Singhal, Ashok	CFD Research Corp.
Skjellum, Anthony	Mississippi State University
Smith, Barry	Argonne National Laboratory
Thompson, Joseph	Mississippi State University

Wang, Paul — Kent State University
Weerawarana, Sanjiva — Purdue University
Weerawarana, Shahani — Purdue University