PSE Research at Virginia Tech: An Overview

**Marc Abrams, Donald Allison, Dennis Kafura, Calvin Ribbens,
Mary Beth Rosson, Clifford Shaffer, Layne Watson**

**Department of Computer Science
Virginia Tech
Blacksburg, VA 24061**

The purpose of this report is to give an overview of the activities of the research group in **problem solving environments** (PSEs) at Virginia Tech. Most of the report is devoted to an introduction to the area itself, with particular emphasis on our perspective and goals. After a brief attempt at defining the term PSE, we consider individually the three key words:

problem:	what problem domains are we addressing?
solving:	what characterizes the solution process?
environment:	what should the computational environment include?

The final section summarize our current specific interests. For more information about PSE research at Virginia Tech, please visit our home page at www.cs.vt.edu/~pse

What is a PSE?

The term problem solving environment (PSE) is a slippery one. Like many terms in Computer Science, different people use it in different ways. Furthermore, as a relatively new and diverse area of research, it resists simple universally accepted definitions. In fact, one of the main points of this research is to understand what a PSE is or should be: what its capabilities should be, what the needs of users are, who the users are, what the fundamental architecture of a PSE should be, what components can be re-used from one PSE to the next, etc.

Keeping in mind that the term PSE is hard to pin down, here are a few attempts at a definition:

- From [Gallopoulos, Houstis, and Rice, 1994]:

  > ‘‘A PSE is a computer system that provides all the computational facilities necessary to solve a target class of problems. These features include advanced solution methods, automatic or semiautomatic selection of solution methods, and ways to easily incorporate novel solution methods. Moreover, PSEs use the language of the target class of problems, so users can run them without specialized knowledge of the underlying computer hardware or software. By exploiting modern technologies such as interactive color graphics, powerful processors, and networks of specialized services, PSEs can track extended problem-solving tasks and allow users to review them easily. Overall, they create 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.’’

- A similar definition from one of our recent proposals:

  ‘‘A problem solving environment is a computational system that provides a complete and convenient set of high level tools for solving problems from a specific domain. The PSE allows users to define and modify problems, choose solution strategies, interact with and manage appropriate hardware and software resources, visualize and analyze results, and record and coordinate extended problem solving tasks. A user communicates with a PSE in the language of the problem, not in the language of a particular operating system, programming language, or network protocol.’’

- Here are a few additional remarks that help to describe how we view PSEs and PSE development:

  - A PSE applies state of the art research and technology in many subdisciplines of Computer Science to create powerful and convenient computing environments for specific application areas, primarily in science, engineering, and manufacturing. The relevant subdisciplines include artificial intelligence, collaborative computing, graphics and visualization, human-computer interaction, networks and the World Wide Web, numerical analysis, object oriented computing, parallel and distributed computing, and software engineering.

  - A motivational example: A PSE is the software infrastructure needed to support hundreds of engineers, physically distributed, collaborating on the design of a new airplane, and using a heterogeneous collection of models and codes and machines. More specifically, the design approach known as *multidisciplinary design optimization* (MDO) used in aerospace engineering is a good example of a problem domain that could benefit from a PSE. In MDO, the disciplines of fluid dynamics, controls, and structures are all combined in aircraft design. Large scale engineering design problems such as this are heterogeneous in the extreme: many models, many codes, many people, many machines, many locations. Without a sophisticated, usable software environment, MDO is virtually impossible.

  - Another motivational example: A PSE is the system that the engineer/scientist of the year 2010 will use to solve problems on the global meta-computer.

  - The *need* for a PSE increases with the complexity and heterogeneity of the application and the problem solving process, i.e., more models, more realistic models, more people, more codes, more machines, more cases. A computational scientist of a generation ago used a relatively simple set of tools: FORTRAN, mainframe computers, output consisting of tables of numbers and simple plots. Today’s scientists use a variety of programming languages---from Fortran90 and HPF to C++ and Java to problem-specific and visual programming environments; they use a complex and rapidly evolving set of parallel computer architectures; they visualize output in three dimensions, with motion, and in immersive environments; and they collaborate with a research team scattered over the world. Clearly, a usable software infrastructure to help scientists manage and coordinate all these problem solving tools is needed.

  - The *possibility* of building a good PSE increases with the power and sophistication of the key technologies; i.e., networking, visualization, parallel processing, interface tools, etc.

Problems of Interest

The initial focus of our work is on problems and problem-solving arising in manufacturing, engineering research and development, and the natural sciences. Our motivating examples have most of the following characteristics:

- They are based on sophisticated mathematical models.
- They require sophisticated numerical solution methods.
- They require large scale high performance computing resources.
- They are often multi-disciplinary.
- They are often solved by groups rather than individuals.

The Problem *Solving* Process

What characterizes the problem-solving that PSEs assist? An important aspect of this work is to understand what tasks the scientific problem-solver wants to do in a PSE, and how the PSE can help. So it is important to have an understanding of the problem solving process itself. One can imagine many different settings or scenarios. Here are a few:

- **Research.** A scientist wants to compare models or algorithms or implementations. A small number of test cases are used. Accuracy or computational performance are of primary interest. Comparison with experimental data may be required. The goal is to model some physical phenomenon more accurately, or to develop a better solution algorithm.

- **Design and development.** An engineer is trying to design a better system of some kind. The mathematical models and numerical methods used are relatively fixed, although a few key parameters (e.g., the geometry) may be varied. The goal is to find a better design, according to some criteria. This optimization process may be automatic, but is most likely to involve human-in-the-loop steps. Previous ‘‘nearby’’ solutions are of considerable help.

- **Manufacturing.** A plant manager or technical officer must monitor and adjust some process to respond to current conditions, e.g., properties of input materials, the current operating environment. Possible adjustments are evaluated for their predicted effect on the manufacturing process. The goal is to produce a product in the most timely, cost-effective manner, while maintaining quality. Speed may be especially critical in this setting, where shutting down the manufacturing line is virtually unacceptable. Previous solutions and expert help are extremely valuable.

- **Education.** A student is studying a well-understood problem domain, experimenting with a variety of models, test cases, methods, etc. The goal is primarily to understand the current state of the art, rather than to generate new insight or new designs. A repository of sample problems and solution strategies would be very helpful, as would expert assistance, and explanation regarding problem solving steps and choices.

Characteristics of the *Environment*

The adjectives below represent a shopping list of characteristics that are desirable, to one degree or another, in a scientific PSE:

- **Problem-oriented.** The PSE should allow the specialist to concentrate on their discipline, without having to become a self-taught expert in networks or parallel computing or the World Wide Web.

- **Integrated.** Many problems and solution strategies are extremely heterogeneous: in models, codes, applications, machines, even terminology. One of the most challenging aspects of good PSE design will be to manage all this heterogeneity in an integrated way, so that the user has at least the illusion of working in a predictable and consistent environment.

- **Collaborative.** Virtually all science and engineering is done in a collaborative mode, increasingly with physically distributed participants. Collaboration may be synchronous or asynchronous.

- **Distributed.** A PSE may need to be distributed to facilitate remote collaboration (as just mentioned), or simply to make remote computational resources available to a single scientist.

- **Persistent.** Problem solving sessions are not isolated events. The process is distributed in time as well as space. A PSE should support this, perhaps with a ‘‘notebook’’ paradigm, perhaps with some concept of continuing sessions, perhaps with a problem and solution repository, etc. Each problem solving session has a previous context it may need to inherit, and is likely to produce results and insight that will be used in subsequent sessions.

- **Powerful.** No matter how good the PSE looks and feels, if it does not make available sufficiently powerful hardware and software resources, it will not be useful in solving the problems of greatest interest. Note that in educational settings, good use could be made of slightly less powerful PSEs, however.

- **Open, flexible, adaptive.** In many settings it is important that PSE-builders and sophisticated users be able to tailor or add to the functionality of a PSE.

- **Graphical, visual.** Most large scale applications require visualization of results; many rely on graphical input as well.

- **Intelligent.** In certain settings it is likely that a PSE could supply some expert ‘‘advice,’’ in choosing among several numerical methods, for example, or in advising a machine operator on the factory floor about process control.

Our interests in PSEs

The primary goal of this research is *not* to build a series of PSEs that will solve all problems for a set of application areas. Instead, our focus is on software infrastructure to make it easier to build and adapt powerful and usable PSEs. Of course, in order to understand the problem and investigate solutions, one needs to work with real users and build some real PSEs ...

Specifically we are most interested in issues such as these:

- Object oriented computing. How can OO technology be leveraged to increase the flexibility and interoperability of PSEs and PSE components? How can OO principals and technology be used in the context of legacy codes that are definitely not object oriented?

- Human-computer interaction. What are the best ways (paradigms, models, ...) for users to interact with PSEs and with other users? How can PSEs be made more flexible and adaptable to different types of users? How usable are PSEs? How do users of scientific PSEs like to interact with the system and with fellow users? How can a PSE be useful for both disciplinary experts and general or casual users?

- Parallel algorithms and parallel computing. The computational heart of most scientific PSEs is a substantial numerical simulation, requiring state of the art high performance computing. What level of expertise regarding parallel computing should an engineer or scientist be forced to have? What can be hidden, what should be visible regarding parallelism?

- Numerical analysis. What accuracy and robustness issues arise when PSEs begin to combine problem solving modules in unanticipated ways? How can non-expert users be supplied with guidance in choosing mathematical or numerical models? How can a system be designed to degrade gracefully when an inner or server computational kernel fails?

- Network-based computing. How can the Web and its descendents be leveraged to make better PSEs available more cheaply? When enough affordable bandwidth exists to make machine cycles a utility (like electricity) or to make problem solving servers a good option, how can all of this be used in scientific problem solving? Or are there other ways to think about it?

- Computer-supported collaborative work. How can collaboration across space and time, among many users, be supported? What technical and other issues are there and how can they be addressed in this context? How can collaboration be exploited to allow new efficiencies in problem solving?

- Visualization. How can scientific visualization tools be more easily integrated with the other components of modern scientific problem solving? What special challenges and opportunities arise with the rise of immersive visualization environments?

References

E. Gallopoulos, E. Houstis, J.R. Rice, ‘‘Problem-solving environments for computational science,’’ *IEEE Computational Science & Engineering*, 1, 1994, 11--23.

J.R. Rice & R.F. Boisvert, ‘‘From scientific software libraries to problem-solving environments,’’ *IEEE Computational Science & Engineering*, Fall, 1996, 44--53.