Future Generations of
Problem–Solving Environments

José C. Cunha
Computer Science Department
Faculty of Science and Technology
New University of Lisbon
Monte da Caparica, PORTUGAL
jcc@di.fct.unl.pt

This paper discusses issues towards future generations of Problem–Solving Environments (PSE). The paper discusses several dimensions involved in the design and implementation of such environments. It illustrates the above concepts in two distinct ways. Several efforts on the development of PSEs are discussed according to the above dimensions. A case study is then discussed which relates to an ongoing project in the author’s institution. This discussion focus on research on coordination issues involving computational and steering components in a dynamic PSE.

1 Problem–Solving Environments

The main goal of a PSE is to provide a suitable level of transparency to an end-user concerning the specification and solution of a problem in terms of concepts specific to the problem domain. A PSE should allow the development of rapid and efficient prototypes to ease the experimentation with specific solutions and allow the user to learn from experience. Although the idea of PSE is not new and may be seen as an ultimate goal of any computer system, several recent technologies are enabling us to develop more fully integrated environments. Such technologies range from parallel and distributed computing, component based systems, advanced interactive visualization, intelligent knowledge processing and discovery, to large-scale distributed computing. Without the support of adequate tools and environments an end-user will not be able to exploit the full potential of those
technologies. The awareness to these issues has been emerging in multiple projects all over the world[5, 6].

This section defines the concept of a PSE as an integrated environment, supporting an entire life cycle of development and execution steps to solve problems in a specific application domain.

The development steps help the user in producing a specification of the problem to be solved and to support its rapid prototyping so that it may be submitted to execution. This phase involves tools which are dependent on the specific application domain. They may range from visual specification languages to intelligent components providing expert assistance to help the user generating and tailoring the PSE to the specific application requirements and user needs.

The execution steps allow the user to interact with an ongoing experiment, i.e. by controlling and monitoring scientific experiments. They also provide support for the visualization, processing and interpretation of the input or generated data, according to the user’s interests at each point during the experiment. This may simultaneously involve both simulation processes and online devices or physical instruments. So this phase requires the ability to perform activities on a diversity of heterogeneous components. Examples of such activities include the selection, evaluation and testing of individual components to assure they meet the intended specifications. It also includes their activation, interconnection and configuration, the management of working sessions, and the monitoring and control of their dynamic evolution.

Examples of components include the problem solvers, their associated expert assistance tools, tools for data processing, interpretation and visualization, tools for monitoring and computational steering, online access to large databases, and so on.

Some of those components are specific to the application domain, while others are generic tools that can be adapted according to each application and experiment. The diversity of the above mentioned components requires adequate infrastructures to support heterogeneous computing models. Such heterogeneity concerns not only the hardware and operating system levels, but also the distinct computational models that are used within each component. In the past decade, many issues for handling heterogeneity including component interconnection have been addressed by multiple models and associated middleware platforms. This has enabled the research and development of higher levels of functionalities to support the above mentioned activities in a Problem-Solving Environment.
This paper consists of the following sections. Section 2 identifies the requirements for modern generations of PSE. Section 3 presents the main dimensions that should be considered when developing those environments and briefly survey approaches to develop flexible infrastructures for PSEs. Section 4 describes ongoing work at the author’s institution concerning an experience on building a parallel and distributed PSE. Finally the paper presents an outline of future research directions.

2 Requirements for Future Generations of PSE

This section identifies several user and application requirements and it discusses several aspects that should be supported by future PSEs.

Due to the complexity of the simulation models, the large volume of input or generated data, and the difficulties of their interpretation and classification, PSE must satisfy several main requirements which are briefly summarized in the following.

The first series of requirements concerns the end-user and application needs. Three main aspects were selected in order to focus the discussion on more specific issues.

Higher Degrees of User Interaction. In the recent past there has been an increased awareness to the importance of providing user interfaces at distinct abstraction levels. There is also a need of an increased flexibility in user and component interaction. On one hand this puts a requirement for more advanced tools for computational steering and advanced visualization. On the other hand, there is a need to provide distinct modes of operation in the same PSE, e.g. allowing off-line of on-line processing or visualization, that should be selected depending on the user interest at each point during an experiment. Another example corresponds to the ability of including additional observation or control components into an existing environment in order to provide some specific functionalities.

Intelligence in PSEs. Advising, explaining, and expert system tools will become more and more important to assist the user during the development and execution steps. The search for a balance between automated intelligent tools and an adequate level of user interaction will be a major issue in future PSEs.
Multidisciplinary Nature of the Applications. There are two related aspects in this issue. One aspect deals with the need to support interactions between distinct submodels. Recently, multiple heterogeneous and hybrid components have been developed for a diversity of scientific applications, for example the coupling of numerical codes, or the interaction between evolutionary computing models. Other aspect of the multidisciplinary nature of PSEs concerns the need to rely upon distributed collaborative environments. This should enable the interactions and coordination of activities among multiple users which are experts from different subproblems in a given multidisciplinary application.

The following series of requirements concerns the PSE architecture issues that should be addressed in order to enable the above (and other still unforeseen) user requirements.

Infrastructures for PSEs. A PSE should work on top of low-level and middleware layers providing the services of an equivalent meta-level distributed operating system. Besides heterogeneity at the component level as mentioned above, other issues must be addressed, such as operation at a small or large scale, security, resource management and system configuration. Recently there has been intense research on the development of cluster computing and metacomputing platforms and this trend will surely continue in the near future.

Software Architectures. In order to build flexible PSEs that can meet the mentioned user needs, more flexible tools and software architectures should be devised. An important issue is the ability to adapt the tools and the entire environment according to the user interest. As such interests may change during both the development and execution steps, the focus is put on the reuse of components and their dynamic modification. So, the application of recent object-oriented and component-based technologies will continue having a great impact upon the design of PSEs. On the other hand, as applications become more complex, including a large diversity of components, one needs to rely upon models for the abstract specification of PSEs. So PSEs will also increasingly rely upon recent work on software architectures. This will naturally include the models and tools to reason about global system properties and to support the transformations between software levels.
Generation of PSEs. Historically, PSEs have been developed by "manually" assembling a usually small set of components that are interconnected in a specific way in order to satisfy some specific application. New methods for developing and generating PSEs are necessary in order to meet their intended flexibility and their increased complexity and size. This raises the issue of how to identify more generic architectures and services for PSEs, that can be tailored to specific classes of target problem domains. It also includes work on tools for supporting the more / less automatic generation of specific PSE, and will push the development of meta environments that will help generating specific working PSE environments and tools.

Dynamic Configuration and Coordination Issues. All of the above user requirements motivate the study of dynamic PSEs. These systems will support dynamic component integration and modification of interaction patterns among components. This will also benefit from existing theoretical and practical developments on the design of abstract patterns of interactions, and on the dynamic reconfiguration of software architectures. Issues of component and tool coordination will become increasingly important. This is due to the distributed and dynamic nature of the PSE components, and the need to dynamically adjust their interactions depending on the user needs, the evolution of the experiments, and the system behavior.

The above requirements pose new challenges to the design and implementation of future generations of PSEs, due the diversity of dimensions that must be considered and the increasing complexity of the resulting environments[10, 12].

3 Main Dimensions in PSE Development

The main dimensions involved in the development of a PSE are sketched in the figure 1 in a abstract way.

The figure aims at emphasizing the fact that on one hand, a PSE includes a set of components that are specific to each application domain (e.g. a simulator). On the other hand there is a set of generic tools that should be provided all along the above dimensions. For example, a monitoring tool or a component interconnection model. The task of building a PSE involves both types of tools and components, i.e. application specific and generic.
There is a need to provide tools to support the process of application building, by selecting, evaluating, testing, configuring, activating and interconnecting, monitoring and controlling the execution of multiple heterogeneous components.

These goals are best achieved by considering the development of a PSE at several distinct abstraction levels, as briefly summarized below.

**Coordination.** This aspect will become increasingly important to deal with highly dynamic systems[7]. It includes the consistent representation and management of patterns of interaction among components, and the definition of the corresponding cooperation and communication models. It requires adequate models and frameworks for the software architecture of the PSE.

**Software Architectures.** First, this concerns the specification of the struc-
ture of a system in terms of its components and interconnections[11]. Second, this provides the models and tools to reason about global properties of complex PSEs.

Monitoring and Control. This aspect relates to the facilities for observation and control of distributed experiments, including distributed monitoring, computational steering and advanced visualization[8, 9].

Resource Management and Interconnection Services. This aspect relates to the configuration of parallel and distributed heterogeneous virtual machines, to the activation of component instances, the models and infrastructures for the interconnection of heterogeneous components, and the management of local and large scale operations upon metacomputing infrastructures[4].

Infrastructures. This includes the hardware and operating system level platforms that support heterogeneous parallel and distributed processing as mentioned in a previous section.

Recently several projects on PSEs all over the world have been exploiting the above dimensions in distinct ways[6]. Among the diversity of ongoing efforts we briefly mention three approaches that we find address important dimensions for PSE development:

Globus. This project built a basic infrastructure for metacomputing, "the GRID", which gives access to distributed resources at a large scale[4]. The project addresses the infrastructures and the resource management dimensions in its basic layers. It allows the development of high-level services on top of the basic layers, in order to support the upper dimensions shown in figure 1.

Distributed Computational Laboratories. The main goal of this project is to support increased interactivity in high-performance computing for single users and for cooperative users[9]. In its basic layers it provides a basic infrastructure for distributed resource management. It then provides services for the management of experiments including computational steering, monitoring and dynamic system behavior.

A Generic Problem-Solving Environment. This project aims at building a generic infrastructure that can be used to implement PSEs for
distinct application domains[13]. It has a basic infrastructure for distributed computing. Then it offers an intermediate layer which provides a set of generic services and mechanisms for the specification of component-based systems and for abstract resource management. An application dependent layer is then used to build specific PSEs for each domain.

These three projects illustrate representative efforts on providing support for building PSEs. They show how a large diversity of issues are being addressed at all layers of a distributed computing system. These kinds of experiments are opening the way to the development of more advanced PSEs that can meet the above mentioned user and system requirements.

4 An Experience Towards Dynamic Problem-Solving Environments

This section discusses ongoing work at the author’s institution aiming at building more flexible and dynamic PSEs[3]. This project has two main goals:

- To develop a framework to support parallel and distributed PSEs consisting of heterogeneous components. The focus of this research concerns two main aspects:
  - The design of flexible and extensible tools supporting observation and control services.
  - The study of the requirements for dynamic PSEs, their impact upon their software architecture, and the required coordination models.

- To use the above framework to implement prototypes of PSEs for specific application domains. From the application side, early developed prototypes can be used in real applications and adapted according to the user needs. From the computing side, the user requirements are better understood and can be used to improve the functionalities offered by the PSE in terms of development and execution support tools.

In the following a description is given of the above aspects.
4.1 A Multidisciplinary Project

The main motivation for this project resulted from our work with research groups in other scientific domains, namely in the environmental sciences. Parallel and distributed processing solutions are necessary to exploit complex simulation and optimization processes. Such solutions can be tested on a distributed software laboratory that is built by interconnecting multiple components such as simulation, visualization and interactive control tools. By using such a laboratory, a scientist or an engineer will be able to perform experiments by observing and controlling the evolution of a simulation.

As an example, we have worked on the optimization of the design of Waste Water Treatment Plants (WWTP) in cooperation with colleagues from Environmental Sciences and Engineering Department[2, 3]. WWTP are pollution control infrastructures which involve investment and operation and maintenance costs. Such costs are both related to the process design calculations and to the layout. This work is part of a large project where the environmental engineers have the main responsibility of global modelling of WWTP. In their work, they need to integrate several submodels.

**Input models.** They provide the input in order to discretize the 3D information for a certain location, using GIS or ground surface modeling tools, plus input of georeferenced information on subsoil properties, climate, etc. They also include databases for parameters, heuristic rules, unit costs, optimization criteria, etc.

**Design and optimization models.** A WWTP has a certain number of units of treatment arranged sequentially and/or in parallel. All these are linked hydraulically by means of pipes and channels. All these first level units can be placed in any place in space, regarding some physical constraints. As there are a virtually infinite number of hypothesis, a mixture of heuristic rules and optimization techniques is used to find a best solution for a given layout or to choose the best design. This includes the basic design data to help generate the flows and pollutant loads for WWTP design. It also includes the unit operations design models, and the hydraulic model of the piping system which connects the unit operations. Economic modeling is also included to select the best alternative based on given criteria.

**Output models.** This includes the use of GIS and CAD systems, such as AutoCAD, for final visualization, and the creation of professional reports and drawings for the practitioner.
All these submodels need to be managed by a central model, which coordinates the data and control flows between them, resulting in a complete computer-aided design tool, very useful in engineering practice. As some of the submodels are very computing intensive, they may locally rely upon parallel processing.

In our work, we have focused our attention on a part of the above project, namely concerning optimization based on Parallel Genetic Algorithms (PGA). The following section gives a brief description of the main results of this work. Here the emphasis is more on the software architecture issues than on the application aspects.

4.2 A Parallel and Distributed PSE for PGA

An objective function describes the relevant aspects of the optimization of the WWTP, as mentioned above. This function is then submitted for optimization using genetic algorithm (GA) techniques. GA models are particularly interesting because they can solve a wide range of optimization problems in real-life situations. They have the following main characteristics, from the point of view of a PSE developer:

- GA are computation intensive so they require parallel processing. Furthermore they are easily parallelized and they provide the opportunity to exploit both shared memory and distributed memory implementations.

- GA are typically applied in problem domains requiring highly interactive computing environments. Due to the large number of parameters affecting the behavior of a GA and their mutual interactions, the following aspects should be supported:

  - Tools to support off-line processing and/or on-line visualization of the evolution of the objective function and the interpretation of the results. This is important due to the large volume of numeric data which can be generated in each run of the GA.

  - Tools to support on-line modifications of the GA parameters, during execution, depending on the observed behavior. This is important so that the user/expert can better understand the impact of certain parameters, and to adjust the prototype depending on the optimization criteria. In general, this is an unavoidable task
in GA applications, so that support for computational steering is very important.

- Besides the speedup obtained through parallel processing, GA can also benefit from distributed processing. This is obvious for example in the case of a PSE which integrates distinct components for visualization, and steering as mentioned above. However, distributed processing can bring increased potentialities beyond those. Due to the complexity and heterogeneity of modern applications, it is often necessary to subdivide them into multiple subproblems, each possible solved by a distinct GA or other technique. In some cases, separate subproblems are allowed to evolve autonomously but they must interact and cooperate in order to satisfy global application constraints or to improve the global application behavior.

In order to meet the above requirements we started the development of a heterogeneous component-based environment for PGA. The resulting environment will be characterized by its flexibility in the configuration and activation of its components, the programming of their interactions, and the monitoring and control of their global and individual behaviors.

Figure 2 gives a scheme of the main types of components in the environment.

The Genetic Algorithm component type supports the execution of genetic algorithms using a sequential or a parallel model. Multiple heterogeneous GA components can cooperate in a distributed implementation as explained below. The Visualization component supports the visualization of the evolution of the GA execution. The Control component supports the configuration and the interactive steering of the GA parameters.

From the point of view of an end-user, this PSE aims at supporting the following main steps of a life cycle of GA application development:

1. Specify the coding of the objective function and the problem representation using the GA model.
2. Configure an instance of the architecture of the PSE by selecting components for GA, visualization and control.
3. Activate component instances and map them onto the execution support platform.
4. Start a working session, with monitoring, visualization and steering.
5. Suspend or stop the execution and perform analysis of intermediate or final results

The goal is to offer as much automated support as possible to the above steps, although this is still under development. A friendly user-interface to support these steps is also lacking in the current prototypes.

Several prototypes of the PSE were developed for the execution, visualization and steering of parallel genetic algorithms. These initial versions only supported a static configuration consisting of multiple heterogeneous components. They mainly differ in the distinct implementations of the GA components: one is based on a shared memory model and runs on a dual-Pentium PC under WindowsNT, and the other two are based on a distributed memory model and run on PVM and MPI-1 environments. The distributed memory model also allows the cooperation between distinct distributed GA components, each component being possibly based on a different model, PVM or MPI-1. This is an extension to the island model for GA where migration of individuals is allowed between islands. It is based on a
group based model for component interconnection which allows communication between independent PVM and MPI-1 applications[3].

The visualization component is the same in all the prototypes and supports on-line display of the evolution of an optimization function.

There are several versions of the configuration and steering for the on-line modification of the parameters of the GA components. One is based on a PVM console which uses also PVM to communicate with the GA components and only runs on the PVM based prototype. Another implementation is based on a distributed monitoring tool (DAMS). DAMS is based on a flexible software architecture which only provides the minimal functionalities for observation and control of a parallel and distributed application. The main idea behind DAMS is to allow incremental extension of new services, depending on the requirements of individual tools in an environment. DAMS is neutral concerning the supported services and the target application model. Instead of a fixed application programming interface (API), DAMS allows each service module to provide a specific API, and allows the configuration of the corresponding low level drivers which act upon the target application. DAMS was used to implement a resource management service for the configuration and launching of the components of the above mentioned PSE. DAMS was also used to implement a steering console for the same PSE.

The main conclusions of the work done until now are as follows:

- The practical work allowed us to developed early prototypes, that could be tested by end-users, although in the end the prototypes were not evaluated in real applications.

- A flexible monitoring and control architecture supporting heterogeneous tools and their interoperability was designed and used to support control and resource management services. From our experience we believe the DAMS architecture will have an important role in the support of the above described life cycle.

- A group based model for interconnecting heterogeneous components was developed. This model allows efficient communication between PVM and MPI-1 applications. However, it has the disadvantage of not being a standard interface. Work is under way towards using standard component based models such as a CORBA. Evaluation of the Globus infrastructure is also part of the current work. This aims at
evaluating more generic implementations of the PSE, possibly allowing large-scale metacomputing.

4.3 Towards Dynamic Problem Solving Environments

Concerning component integration into a PSE, in the most simple case, the components are statically specified and the configuration of the PSE remains unchanged during an entire experiment. In order to provide increased flexibility and allow the user to have a more interactive role regarding the execution of an experiment, it is necessary to consider the dynamic insertion and removal of components from an existing configuration. This applies to the case where a single user is changing the system configuration as a specific experiment progresses, in order to evaluate distinct aspects of the problem. This also applies to the case where multiple users concurrently join ongoing experiments with distinct roles (observers, controllers).

Computational steering plays an increasingly important role in many complex applications where a user must be able to control and observe an ongoing simulation. This is important to help the user learning how the simulation behaves depending on a diversity of application and system parameters. On the other hand, this is also important to allow the user to focus on specific parts of the problem models or to specify the most desirable levels of detail in very complex systems.

Besides user driven steering, agent driven steering can be useful to allow the automatic control of the evolution of a computation. In some applications, previous knowledge about problem behavior can be integrated into so-called intelligent controllers that may act autonomously upon the computational components. For example, such knowledge can result from previous experimentation with the system. Furthermore, for heterogeneous applications it could be useful to simultaneously support user driven and automatic steering components.

Besides the definition of suitable interfaces for the monitoring and steering components, all of the above requires the system to provide adequate mechanisms for the consistent coordination of multiple components.

As part of our current work, we are studying how support for dynamic reconfiguration can increase the flexibility of a PSE, for an end-user. In order to evaluate this aspect, we are designing a collection of application level scenarios which involve multiple tools and components of a PSE. Then we analyze how their dynamic reconfiguration can improve the expressiveness of the life cycle for application development and execution. The final goal of
this work is to be able to model a diversity of interaction patterns between components and to support their dynamic modification. The model defines a collection of coordination operations that allow the control of components and their interconnections.

5 Conclusions

In this paper a survey was presented of the main dimensions involved in the design of future generations of Problem-Solving Environments. A hierarchy of conceptual layers was presented in order to identify the most important issues on PSE development. Namely, coordination issues and specification of software architectures are considered important to handle the increased complexity of dynamic PSEs as well as to increase the flexibility of these environments. The paper has presented an outline of ongoing work at the author’s institution concerning experimentation towards developing flexible tools and their integration in a PSE. The paper also presented the main goals of ongoing research towards the support of dynamic PSEs.

Acknowledgments. The project described in this paper is the result of a joint collaborative effort involving all the members of the Parallel and Distributed Processing Group of the CITI Centre. The work on WWTP modeling is due to David Pereira and Luís Almeida of the Environmental Sciences and Engineering Department. Thanks are due to Bruno Horta, Luís Duarte, Júlio Duarte, Nuno Neves, and Gregzor Fert for their work in the development of several GA prototypes, and to João Vieira and Bruno Moscão for their work on the DAMS. The work reported in this paper was partially supported by the Portuguese CIENCIA, the CITI Centre, and the PRAXIS XXI Programme (SETNA Project).

References


ing: Problem-Solving Environments: Infrastructure and Prototypes (http://www.cs.cf.ac.uk/euresco99/)


