FUTURE GENERATIONS OF PROBLEM–SOLVING ENVIRONMENTS*

José C. Cunha
New University of Lisbon
Monte da Caparica, Portugal

Abstract This paper discusses several dimensions involved in the design and implementation of future generations of Problem–Solving Environments (PSEs). The paper surveys the main requirements posed both by end users and by system developers. The main issues on the development of future generation of PSEs are identified. A case study is then discussed which relates to an ongoing project in the author's institution. This research concerns the study of component coordination in a dynamic PSE and how this issue may influence the design of the architecture of a generic PSE.

Keywords: Problem-solving environments, coordination.

1. PROBLEM–SOLVING ENVIRONMENTS

A Problem–Solving Environment (PSE) aims at helping an end-user in the specification and solution of a problem in terms of concepts specific to the problem domain. It should allow the development of rapid prototypes to ease the experimentation with specific solutions and allow the user to learn from experience. Several recent technologies are enabling to develop more fully integrated environments, ranging from parallel and distributed computing, component based systems, advanced interactive visualization, intelligent knowledge processing and discovery, to large-scale distributed computing. The awareness to these issues has been emerging in multiple projects all over the world[5, 6].

A PSE is 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

*Thanks to the Portuguese CIENCIA, FCT/MCT, the CITI, and the PRAXIS SETNA-ParComp (2/2.1/TIT/1557/95).

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35407-1_22

R. F. Boisvert et al. (eds.), *The Architecture of Scientific Software*

specification of the problem to be solved and to support its rapid prototyping so that it may be submitted to execution. This involves tools ranging from visual specification languages to intelligent components providing expert assistance to help the user generating and tailoring the PSE to the specific application and user needs.

The execution steps allow the user to interact with an ongoing experiment, by controlling and monitoring its evolution. They support the visualization, processing and interpretation of the input or generated data, according to the user's interests at each point during the experiment. This requires the ability to perform activities on a diversity of heterogeneous components, such as 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. Examples of such activities include the selection, evaluation and testing of individual components, their activation, interconnection and configuration, the management of working sessions, and the monitoring and control of their dynamic evolution.

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. 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 development of higher levels of functionalities to support the mentioned activities.

In the remaining of this paper, Section 2 identifies requirements for modern generations of PSEs. Section 3 presents the main dimensions that should be considered when developing those environments and briefly survey approaches to develop more flexible infrastructures for PSEs. Section 4 describes ongoing work towards building more flexible PSEs.

2. REQUIREMENTS FOR FUTURE GENERATIONS OF PSE

Due to the complexity of the simulation models, the large volume of data, and the difficulties of their interpretation, PSE must satisfy a series of requirements concerning the end-user and application needs:

- *Higher Degrees of User Interaction.* Increased flexibility in user and component interaction demands user interfaces at distinct abstraction levels. On one hand this requires more advanced tools

for computational steering and advanced visualization. On the other hand, it requires distinct modes of operation in the same PSE, e.g. allowing off-line of on-line processing or visualization, to be selected depending on the user interest at each point during an experiment. This also requires the ability of bringing new components into an existing environment in order to provide some specific functionalities.

- *Intelligence in PSEs.* Advisoring, explaining, and expert tools are 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.* This poses the need to support interactions between distinct sub-models, based on multiple heterogeneous and hybrid components, e.g. for the coupling of numerical codes, or the interaction between evolutionary computing models. On the other hand, PSEs should evolve towards distributed collaborative environments, to enable the interactions and coordination of activities among multiple users which are experts from different subproblems.

The PSE architecture must address the following main issues in order to enable the above user requirements.

- *Infrastructures for PSEs.* A PSE should be able to work on top of low-level and middleware layers which provide the services of a meta-level distributed operating system for cluster computing and global computing platforms. Besides heterogeneity issues, other aspects must be addressed such as operation at a small or large scale, security, resource management and system configuration.

- *Software Architectures.* Flexible PSEs require the ability to dynamically adapt the tools and the software architecture of the entire environment. As the user interests may change during both the development and execution steps, the focus is put on the reuse of components and their dynamic modification, relying upon object-oriented and component-based technologies. On the other hand, as applications become more complex, including a large diversity of components, one needs models and tools to support the abstract specification of PSEs, the reasoning about global system properties and the transformations between software levels.

- *Building PSEs.* Historically, PSEs have been developed by "manually" assembling a usually small set of components that are interconnected in a specific way for a 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 requires the identification of more generic architectures and services for PSEs, that can be tailored to specific classes of target problem domains. It also requires tools for supporting the more / less automatic generation of specific PSEs.

- *Dynamic Configuration and Coordination Issues.* Dynamic PSEs will support the modification of components and their interaction patterns. This requires both theoretical and practical developments on the design of abstract patterns of interactions, on the dynamic reconfiguration of software architectures, and on the coordination of distributed systems.

These requirements pose new challenges to future generations of PSEs [10, 12].

3. MAIN DIMENSIONS IN PSE DEVELOPMENT

A PSE involves tools which are specific to the application domain, e.g. a simulator, and other more generic ones, such as a monitoring tool. There is a need for tools supporting the application building, by selecting, evaluating, testing, configuring, activating and interconnecting, monitoring and controlling the execution of multiple heterogeneous components. These main dimensions are represented in the figure 1.

The development of a PSE addresses issues at several distinct abstraction levels. *Coordination* concerns the consistent representation and management of dynamic patterns of interaction among components [7], 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* concern the specification of the structure of a system in terms of its components and interconnections[11],and it provides the models and tools to reason about global system properties. *Monitoring and Control* includes support for the observation and control of distributed experiments, such as distributed monitoring, computational steering and advanced visualization[8, 9]. *Resource Management and Interconnection Services* handle configuration of parallel and distributed heterogeneous virtual machines, activation of component instances, infrastructures for the interconnec-

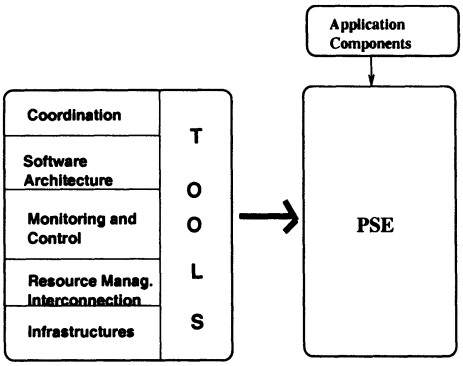

Figure 1 Dimensions in PSE development

tion of heterogeneous components, and management of local and large scale operations for clusters and metacomputing[4].

Among the diversity of ongoing projects[6] we mention some of the representative efforts which are opening the way to more advanced PSEs. *Globus*[4] provides an infrastructure for metacomputing, "the GRID", giving access to large scale distributed resources and allowing the development of high-level services. *Distributed Computational Laboratories*[9] supports increased interactivity in high-performance computing for single and collaborative users. It provides an infrastructure for distributed resource management, and services for the management of experiments with computational steering, monitoring and dynamic system behavior. *A Generic Problem-Solving Environment*[13] is building a generic infrastructure to implement PSEs for distinct application domains. It relies on an infrastructure for distributed computing and it offers an intermediate layer with a set of generic services for the specification of components and for abstract resource management. An application dependent layer is then used to build specific PSEs for each domain.

4. AN EXPERIENCE TOWARDS DYNAMIC PSE

A project at the author's institution aims at building more flexible and dynamic parallel and distributed PSEs[3]. One goal is to develop a framework for PSEs consisting of heterogeneous components. It should allow the design of flexible and extensible tools supporting observation and control services, as well as the study of dynamic PSEs, their software architecture, and the required coordination models. Another goal is to

use the above framework to implement prototypes for specific application domains which can be used in real applications and adapted according to the user needs. This contributes to improve the functionalities and tools offered by the PSE.

This project initially involved the cooperation with colleagues from the Environmental Sciences and Engineering Department[2, 3] for the design and implementation of a system architecture for Parallel Genetic Algorithms (PGA). The system also included tools for off-line / online processing and visualization of the evolution of the GA computations, as well as tools for on-line modifications of the parameters. Several prototypes were developed for the execution, visualization and steering of PGAs. These versions only supported a static configuration consisting of multiple heterogeneous components. They mainly differ in the distinct implementations of the GA component (based on shared–memory or distributed–memory models), and of the steering components. A flexible monitoring and control architecture (DAMS) supporting heterogeneous tools was designed and used to support control and resource management services. DAMS is based on a design which only provides the minimal functionalities for observation and control of a distributed application. The main idea is to allow incremental extension of new services, depending on the changing requirements. DAMS is neutral concerning the supported services and the target application model. Instead of a fixed API, DAMS allows each service module to provide a specific interface, 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 steering of the mentioned PGA prototype.

A distributed architecture brings increased potentialities to integrate distinct components. Due to the complexity and heterogeneity of modern applications, one often needs to subdivide them, each subproblem being solved by a distinct model which is allowed to evolve autonomously. Still they must be able to interact and cooperate due to global application constraints or to improve global application behavior. In order to meet these requirements, the development of a generic heterogeneous component-based environment is under way. It will provide increased 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.

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. In this way a single user is able to change the system configuration as a specific experiment progresses, in order to evaluate distinct aspects of the problem. Also, multiple users can concurrently join ongoing experiments with distinct roles (observers, controllers). Computational steering plays an increasingly important role in many complex applications to help the user learning how the simulation behaves depending on a diversity of application and system parameters. It 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 complex systems. Besides user driven steering, agent driven steering can be useful to allow the automatic control of the evolution of a computation. Previous knowledge about problem behavior can be integrated into intelligent controllers that may act autonomously upon the computational components. 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. We are studying how support for dynamic reconfiguration can increase the flexibility of a PSE. 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 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

A survey was presented of the main dimensions in the design of future generations of Problem-Solving Environments. A hierarchy of conceptual layers allows to identify the main issues. Namely, coordination issues and specification of software architectures are important to handle the increased complexity of dynamic PSEs and their flexibility. An outline of ongoing work was presented concerning experimentation towards developing flexible dynamic PSE and tools.

Acknowledgments

To the Parallel and Distributed Processing Group. To D. Pereira, L. Almeida, B. Horta, L. Duarte, J. Duarte, N. Neves, G. Fert, J. Vieira and B. Moscão.

References

[1] Casanova, H. and J. J. Dongarra (1997). NetSolve: A Network-Enabled Server for Solving Computational Science Problems, *Int. J. Supercomputing Appl.* **11**, **3**, 212-223.

[2] Cunha, J. C. (1999). Parallel and Distributed Processing in a PSE for Environmental Science, in *European Research Conference on Advanced Environments and Tools for High Performance Computing: Problem-Solving Environments, Infrastructure and Prototypes*, (http://www.cs.cf.ac.uk/euresco99/).

[3] Cunha, J. C. and P. Medeiros and V. Duarte, and J. Lourenço, and M. C. Gomes (1999). An Experience in Building a Parallel and Distributed Problem–Solving Environment, in *Proceedings of PDPTA'99 - International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, USA, CSREA Press, 1804–1809.

[4] Foster, I. and C. Kesselman (1997). GLOBUS: A Metacomputing Infrastructure Toolkit, *Int. J. Supercomputing Appl.* **11**, **2**, 115-128.

[5] Gallopoulos, E. and E. N. Houstis and J. R. Rice (1995). Workshop on Problem Solving Environments: Findings and Recommendations, *ACM Computing Surveys*, **27**, **2**, 277-279.

[6] Houstis, E.N. and J. R. Rice. and E. Gallopoulos and R. Bramley (2000). *Enabling Technologies for Computational Science: Frameworks, Middleware and Environments*, Kluwer Academic Publishers, Boston.

[7] Papadopoulos, G. A. and F. Arbab (1998). Coordination Models and Languages, *Advances in Computers*, **46**, The Engineering of Large Systems, Academic Press.

[8] Parker, S. G. and M. Miller and C. D. Hansen and C. R. Johnson (1998). An Integrated Problem Solving Environment: The SCIRun Computational Steering System, in *Proceedings of the 31st Hawaii International Conference on System Sciences (HICSS-31)*, 147-156.

[9] Plale, B. and K.Schwan and V.Elling and D.King and G.Eisenhauer and V.Martin (1998). Realizing Distributed Computational Laboratories, *International Journal of Parallel and Distributed Systems and Networks.*

[10] J. R. Rice (1997). Future Scientific Software Systems, *IEEE Computational Science and Engineering*, April-June issue, 44-48.

[11] Shaw, M. and D. Garlan (1996). *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, Englewood Cliffs, N.J.

[12] Walker, D. W. (1999). *European Research Conference on Advanced Environments and Tools for High Performance Computing: Problem-Solving Environments, Infrastructure and Prototypes*, San Feliu de Guixols, Spain (http://www.cs.cf.ac.uk/euresco99/).

[13] Walker, D. W. and M. Li and O.F. Rana and M.S. Shields and Y.Huang (1999). The Software Architecture of a Distributed Problem-Solving Environment, Report ORNL/TM-1999/32, Computer Science and Mathematics Division, Oak Ridge National Laboratory, Tennessee.