The Development of a Grid Based Engineering Design Problem Solving Environment

A. D. Scurr[1] and A. J. Keane
School of Engineering Sciences
University of Southampton.
[1]{ads294@soton.ac.uk}

Abstract: This paper gives an overview of the grid based Engineering Design Problem Solving Environment (PSE) being developed at Southampton University. Our current PSE is based on our Options optimiser and the Cardiff VCCE and XML component model. Essentially, VCCE provides a GUI to enable a user to setup and execute a computation by creating a task graph from available components via drag and drop operations on a sketchpad display.

In order to provide an environment that more naturally meets the data-centric view of users, two major enhancements to the PSE are planned. The first concerns scheduling and task farming. The ultimate goal is to achieve within the PSE, an asynchronous computational workflow pattern where analysis tasks can seek to exploit whatever computational resources are available in various workstation clusters. The second enhancement concerns computational resource control and job control and the setting up of an Engineering Design Grid Portal.

1. Problem Solving Environments (PSEs) and Grid Portals

To enhance engineering insight, reduce development costs and improve product quality, engineering design studies are increasingly using sophisticated analysis packages together with optimisation tools. The chief characteristics of these design studies are the need to optimise a design where the analysis is very time consuming, where there are multiple methods or domains of analysis and where we need to deploy distributed, cluster based computing systems. The use of PSE's facilitates working on such problems. The accepted definition of a PSE is: "A PSE is a computer system that provides all the computational facilities needed to solve a target class of problems." (J. Rice - Purdue University). Closely aligned with PSE development is the concept of grid computing and the idea of a grid portal. A Grid Portal is an access point (usually a web browser page) designed to facilitate the use of a PSE in a particular discipline by providing seamless access to the PSE's range of computational tools, information and resource.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 881−889, 2002.

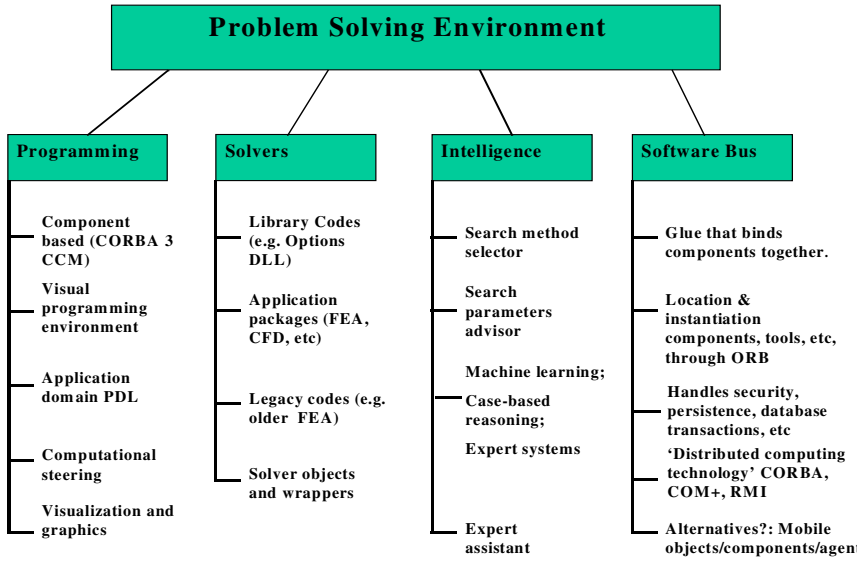

Figure 1: The main components of a generic PSE

The main sub-systems in a generic PSE (Figure 1.) include visual-programming environments for graphically composing, steering and monitoring applications, various component wrapped analysers and solvers, an integrating software-bus (here CORBA) and optionally one or more AI systems to assist the user formulate a computational strategy.

2. Prototype Engineering Design PSE

We have developed a prototype PSE built around our Options [1] suite of search/optimisation programs and various CFD codes. Our current Engineering Design PSE incorporates the Cardiff University Visual Component Composition Environment (VCCE) as its front-end [2] and uses CORBA 2.3 as its distributed object middleware (software bus above). In Figure 2, components are dragged from the Component Repository on the left column of a sketchpad display and dropped onto the canvas of the Component Composition Tool on the right of the display. Components can then be joined together if their interfaces are compatible to form a task graph. This defines the execution order and dependencies between the various components making up a job. For a component to be available for use in VCCE, an XML definition of the component must be available in the Component repository. The component's XML definition is based on the Component Model defined in VCCE.

The Engineering Design PSE (Figure 3.) uses a component architecture based on CORBA objects (here, the terms *CORBA Object* and *component* are synonymous) whose structure and functionality are defined in CORBA IDL interfaces. Optimisation

is a procedure that searches through the design domain for optimum designs. During the search, the optimiser continuously calls the analysis code and, based on the results to date, decides on the next design point to evaluate. In the Engineering Design PSE, the optimiser is contained in the Options CORBA Object and the analysis codes are contained in the OPTFUN/OPTCON CORBA Object.

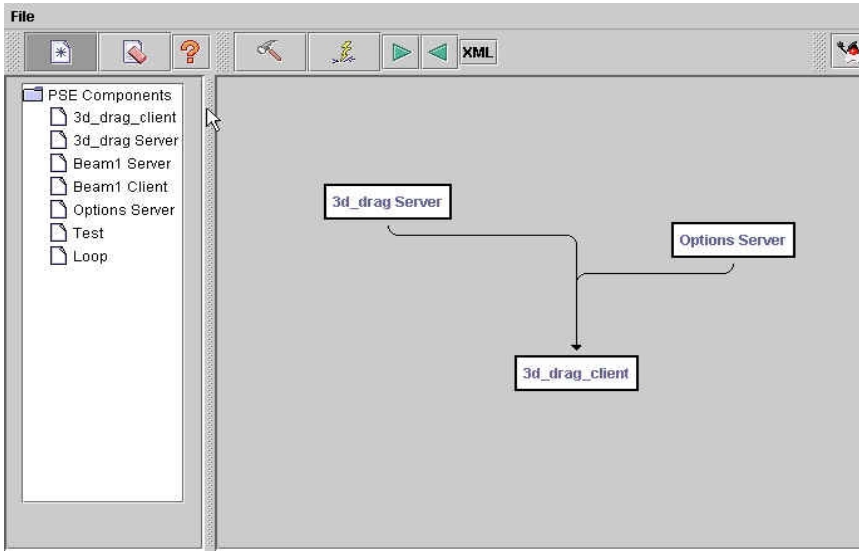

Figure 2: VCCE drag and drop interface to Engineering Design PSE

The current PSE has the following components:

- An Options component which consists of the Options search package wrapped in a C++ wrapper which uses the Options direct subroutine access facilities to set up and run a problem defined in the Client Application code. Options stores all problem data in its own database which can be output as a problem definition file for later re-input.
- A user supplied problem specific component that provides the analysis calculations on the optimisation design variable values found in the Options component solution method.
- A Client Application component, which defines the problem to be solved and specifies the Options search/optimisation method to be used, input parameter values, etc.
- A callback object created by the Client Application, which helps de-couple the Options and OPTFUN/OPTCON components.

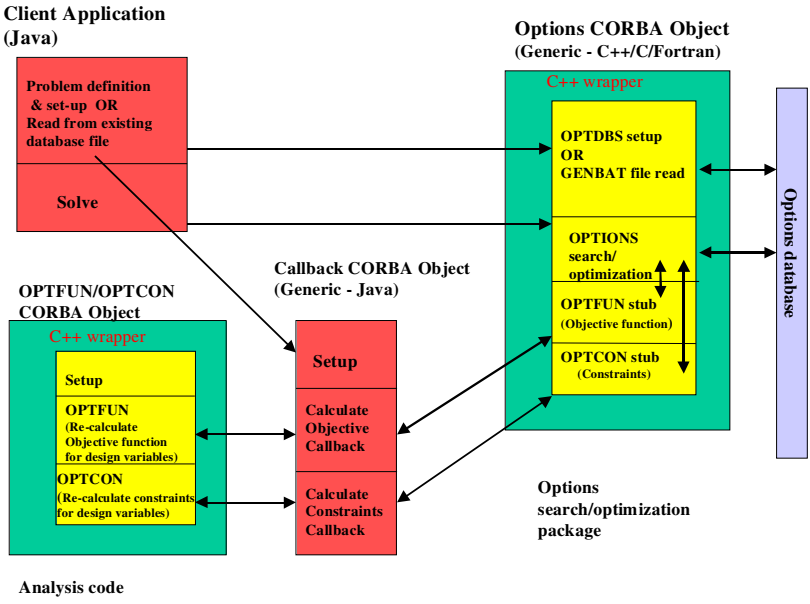

Figure 3: The Engineering Design PSE component architecture

3. Design Studies Process

Designers generally use approximation techniques to solve optimisation problems in order to reduce the otherwise enormous computational effort that would be involved if only full fidelity models were used. Depending on the size of the problem domain space, local, mid-range or global approximations are employed. For single design point optimisation, the above PSE architecture is appropriate, but for global approximation techniques such as Response Surface Modeling (RSM) [3], rather different computational workflow strategies are required.

In the DoE/RSM process (left part of Figure 4.) some technique (e.g., Latin hypercubes) is used to select a set of initial design points (typically say 100 points) that will span the design space. The analysis codes are then run for each design point, ideally in parallel on some large compute cluster, as the design point evaluations are independent of each other. From the resulting database, a response surface is built using the chosen approximation method (e.g., kriging [4]). If satisfactory, the surface is searched for an optimal design point (right part of Figure 4.) and if found, full analysis codes are run at that point. If the analysis is satisfactory and all design constraints are met, the process terminates; otherwise the new design point is added to the design set, the surface regenerated and the optimisation repeated.

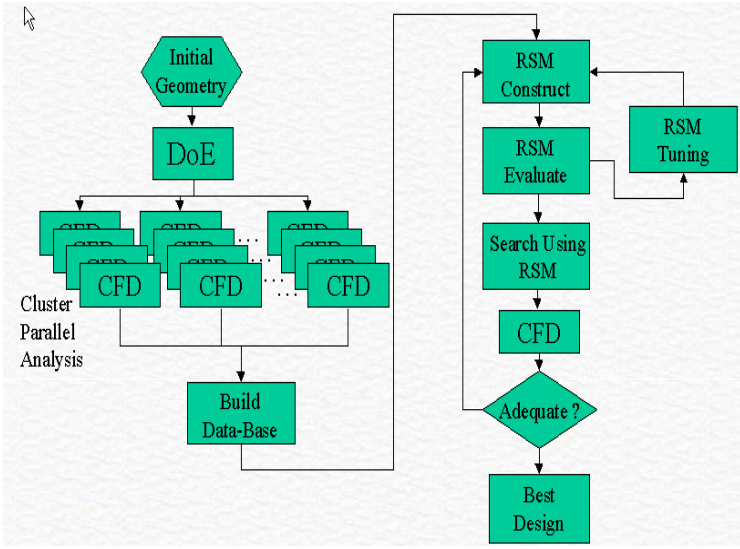

Figure 4: Overall Design of Experiments (DoE)/RSM Process

From a user perspective, the above RSM process suggests a data-centric approach to the computational workflow, where the user is primarily looking at problem design point and results data to-date (and possibly other test data such as flight trial or wind tunnel results in aerospace design problems) in order to decide where new problem analyses need to be run (see figure 5 below). From a computational perspective, the aim is to achieve the maximum amount of parallelism in the analysis computations by exploiting the various cluster resources available. This can be accomplished by de-coupling the search engine computation from the analysis computations and using some task farming approach to schedule parallel running of analysis tasks over the available resources.

4. Engineering Design Grid Portal/PSE Architecture

Rapid advances in commodity computing and the emerging nation-wide computational grid are significantly influencing the development of the next generation of PSE's. The term 'commodity computing' encompasses the development of distributed computing technologies (Web services, Java, JINI, CORBA, DCOM, etc.), plus the use of standard PC hardware in networked clusters. 'Grid computing' refers to the high-performance computing community's creation of Grids; advanced infrastructures designed to enable the co-ordinated use of distributed high performance computers for scientific problem solving. Closely aligned with the concept of grid computing is the idea of portal computing [5]. A user or application portal is a web-based collection of information presented together on a browser page. In many cases a portal can replace the need for users to log on to a number of different computers to gain access to the various resources that exist. In portal computing well-defined services are delivered to the portals instead of giving users direct access to the

operating system prompt. As a result, resource management and security can be exercised at a higher level.

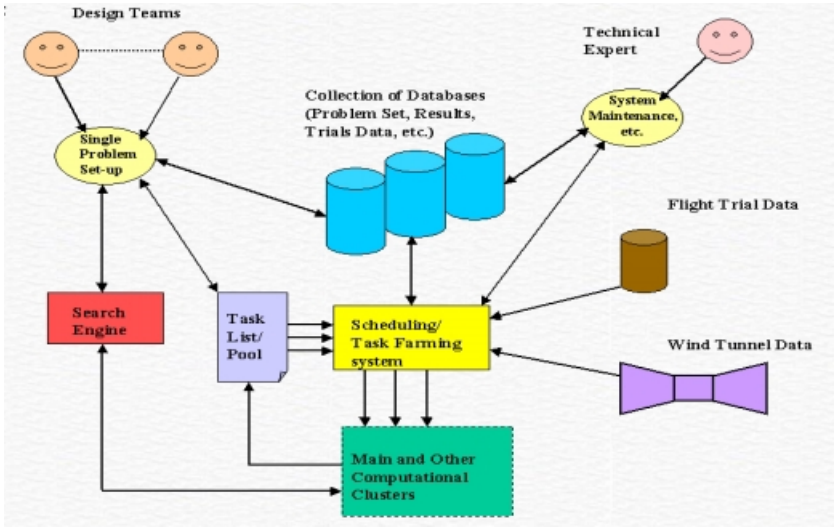

Figure 5: User view of engineering design system

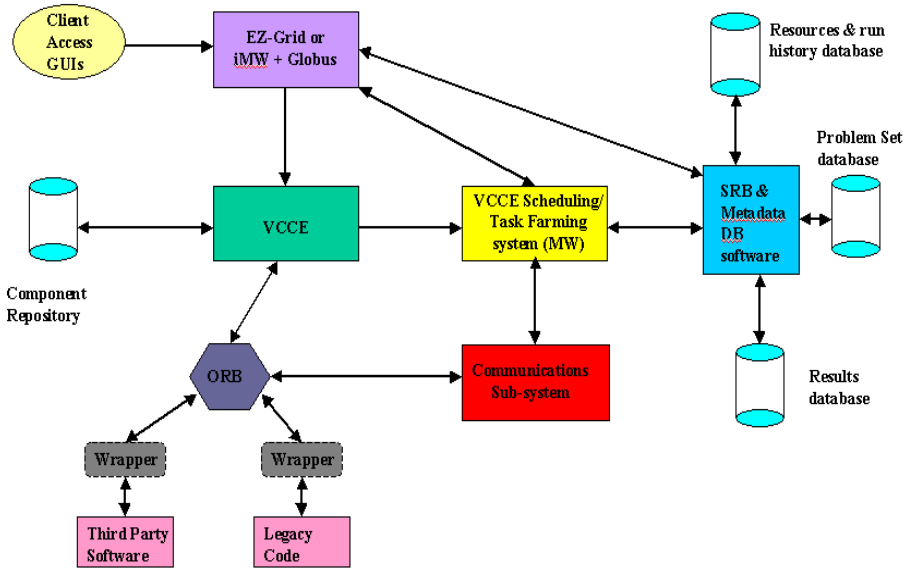

Figure 6: Grid Portal/PSE Architecture

The proposed PSE design involves integrating a number of existing PSE and Globus based software packages in a layered structure, in order to provide a fully functional Engineering Design Grid Portal (Figure 6.). The Globus project [6] aims to provide PSE developers with Grid services for resource management, security and resource discovery. The University of Houston EZ-Grid project [7] uses Globus services to make the use of the Grid easier and transparent for the user. This is achieved by developing easy-to-use interfaces coupled with brokerage systems to assist the resource selection and job execution process.

We propose using the PSE prototype described above to provide the basis for a Design Optimisation Grid portal. This involves extending the PSE functionality with the EZ-Grid/Globus software at a level above the existing VCCE system to provide for Grid user login, job submission, etc. In addition, the development of a task farming approach on the analysis side should de-couple optimisation from analysis task execution in order to exploit Grid technologies to gain more parallelism.

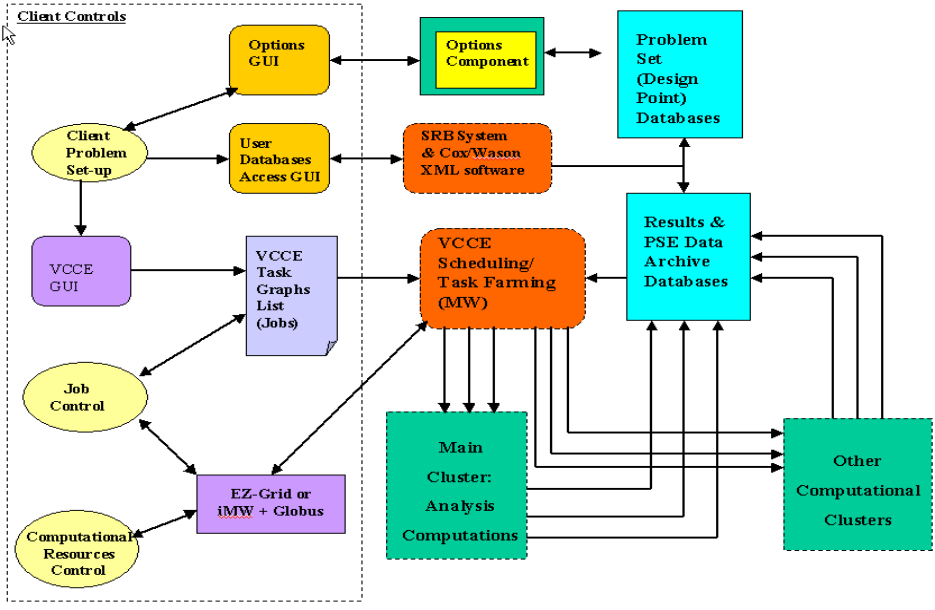

Figure 7: Grid Portal/PSE Computational Workflow

The aim of our Grid Portal/PSE system is to provide the computational workflow scheme shown in figure 7 above where analysis and search/optimisation tasks are de-coupled by the two central database systems (the Problem Set and Results databases). Thus the search and analysis tasks can compute asynchronously, with the Search Engine adding new design point data sets into the Problem Set database and large scale "task farming" on the analysis side. Here, a number of independent analysis tasks can be run in parallel over one or more linked computational clusters under some form of computational resources control. The analysis results are added to the Results

database, where the Search Engine can be guided via the user or advisor (machine learning based) systems in selecting the most promising new design to analyse.

At present, the problem is set-up using the pre-defined components available to the VCCE task-graph interactive front-end. Once the problem task graph is built, the only options are to run or reset the components in the graph. The Job Control function provides facilities to control computations and monitor their progress. Essentially the Job Control will be able to dynamical start/stop/restart computations and change parameters of running processes i.e., provide interactive computational control as opposed to the process scheduling/task farming available in the Computational Resources Control. Also, for some applications, it might be useful to interactively display feedback on the progress of computations (e.g., provide on-demand graphical displays of analysis or search results to-date).

5. Conclusions

The proposed Engineering Design PSE can probably be best seen as an example of a distributed collaborative system, where the Client, Computational Resources, Job Control and multiple computational components (different kinds of agents) engage in a shared activity. The defining feature is that agents in the system are working together towards a common goal and have a critical need to interact closely with each other: sharing information, exchanging requests with each other and informing each other of their status. Concurrency is important, with any agent interacting with other agents running in parallel in a distributed heterogeneous computing environment.

We have developed a prototype Engineering Design PSE that uses the Cardiff VCCE software to provide a drag and drop user interface to create a task graph based on a comprehensive component model and that uses CORBA as the software bus or middleware. The PSE currently supports a restricted cluster communications structure involving three component types (Options, OPTFUN/OPTCON and the Callback CORBA objects) and relies on specific features in the Orbacus ORB to handle asynchronous callbacks. To overcome these communications restrictions and to provide the framework for a generic PSE system, a more general communications sub-system is being developed.

As developed, the current PSE will be able to compute single or multi-point design evaluations over network clusters. However, in order to fully exploit parallel computation over intranet or even internet clusters and to provide an environment which more naturally meets the data-centric view of users, two additional major enhancements to the PSE are planned.

- The first concerns scheduling and task farming. The ultimate goal is to achieve within the PSE, an asynchronous computational workflow pattern where analysis tasks can seek to exploit whatever computational resources are available in various workstation clusters (i.e., task farming), running independently of search/optimisation tasks. Communication between the search and analysis tasks is through the Problem Set and Results databases. Whilst the Options package has some of this functionality available through script files, this functionality should

be provided in a generic way through the development of a scheduling/task farming PSE component.

- The second enhancement concerns computational resource control and job control and the setting up of an Engineering Design Grid Portal. EZ-Grid provides a basic set of components for creating a grid portal, including a GUI for Grid authentication and login, a Job Manager for job submission and monitoring and a Broker Kernel for resource control. However, these components are likely to require considerable enhancements to work with the scheduling and task farming components above.

References:

[1] A.J. Keane, The Options Design Exploration System Reference Manual and User Guide – Version B3.0 February 2000.

[2] O. F. Rana, M. Li, M. S. Shields, and D. W. Walker, A Wrapper Generator for Wrapping High Performance Legacy Codes as Java/CORBA Components in Proceedings of the IEEE/ACM SC2000 Conference, held in Dallas, TX, Nov 10-12, 2000.

[3] Myers, R. H. and Montgomery, D. C. (1995), Response Surface Methodology: Process and product optimization using designed experiments, John Wiley and Sons inc.

[4] Jones, D. R., Schonlau, M. and Welch, W. J. (1998), Efficient global optimization of expensive black-box functions, Journal of Global Optimization, 13, 455-492.

[5] G. von Laszewski, Ian Foster, Jarek Gawor, Peter Lane, Mike Russell. Designing Grid-based Problem Solving Environments and Portals. HICSS 2001.

[6] The Globus Project web-site: http://www.globus.org/

[7] The EZ-Grid Project web-site: http://www.cs.uh.edu/~ezgrid/EZ-Grid.htm