A framework for adaptive visualization

Brodlie, K, Brooke, J, Chen, M, Chisnall, D, Hughes, C, John, N, Jones, M, Riding, M, Roard, N, Turner, M and Wood, J

A Framework for Adaptive Visualization

K. W. Brodlie[1], J. Brooke[2], M. Chen[3], D. Chisnall[3], C. J. Hughes[4i], N. W. John[4],
M. W. Jones[3], M. Riding[2], N. Roard[3], M. J. Turner[2] and J. Wood[1]

University of Leeds[1], University of Manchester[2], University of Wales,Swansea[3] and University of Wales, Bangor[4]

1. Introduction

Although desktop graphical capabilities continually improve, visualization at interactive frame rates remains a problem for very large datasets or complex rendering algorithms. This is particularly evident in scientific visualization, (e.g., medical data or simulation of fluid dynamics), where high-performance computing facilities organised in a distributed infrastructure need to be used to achieve reasonable rendering times. Such distributed visualization systems are required to be increasingly flexible; they need to be able to integrate heterogeneous hardware (both for rendering and display), span different networks, easily reuse existing software, and present user interfaces appropriate to the task (both single user and collaborative use). Current complex distributed software systems tend to be hard to administrate and debug, and tend to respond poorly to faults (hardware or software).

In recognition of the increasing complexity of general computing systems (not specifically visualization), IBM have suggested the Autonomic Computing approach [1] to enable self-management through the means of **self-configuration, self-optimisation, self-healing** and **self-protection.**

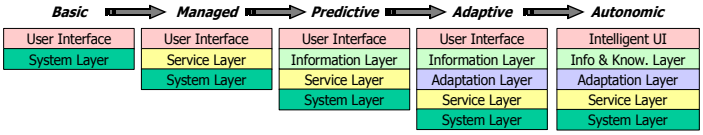

Figure 1: The deployment model for developing a visual supercomputing infrastructure

We can think of a typical user scenario, beginning with the import of a new data set into the system. The system will attempt to visualize the data based upon previous usage. A selection of images will be presented. The user will select the most appropriate, and then will describe their requirements for the system (e.g. real-time, critical, or lower quality and cheaper). The system determines the resources available to fulfil the requirement, and then takes care of the visualization. The user may migrate their visualization images from their desktop to their PDA, or may invite others to join in collaborative visualization. All changes are managed by such an autonomic visualization system. Our previous work [2] made an extensive study of the enabling technologies and the challenges that will be faced in implementing such a system. We proposed a model for the deployment of such a system (figure 1) wherein we suggested that more intelligence about the system environment, user requirements and visualization can lead to a system that is able to analyse, predict and modify its own behaviour, and result in the above autonomic behaviour.

Basic systems provide a user interface to the visual task. Managed systems introduce a service layer (or middleware) (e.g., Grid) to manage the security, distribution, output destination and resources available to a task. Predictive systems add an information layer that can provide data about the performance of the system and quality of visualization. Adaptive systems will begin to use such data to alter their own state in order to achieve self-management, which leads on to a fully autonomic system, where a knowledge base is added to reason about the intelligence (both task side intelligence (e.g., this visualization method is the best for this kind of data), and user side intelligence (e.g., this user prefers this camera control widget)).

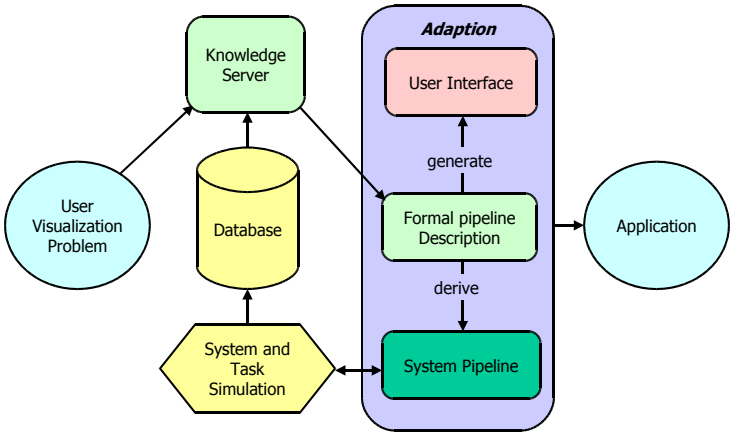

Figure 2: Functional description of e-Viz using the Adaptive Layer (colours illustrate the connections to the adaptive layer in Figure 1)

This poster will present the model of semantic data flow within the visualization system (called e-Viz) that allows the system to self-configure, self-model and self-adapt its configuration subject to general goals from the user. The motivation is to simplify the user's experience of interacting with such complex visual systems, and simplify the integration cost for developers of visual software.

2. Adaptive visualization

In general middleware allows the connection of two or more software components by translating the interface between them. In this context middleware can be regarded as software that allows the programmer to abstract away from system level details. We can further relate this to visual problems by stating that a programmer can write their visualization code as if it is generating images on a single machine, and the middleware will take care of any distribution over a heterogeneous cluster, display on different devices, and/or interaction through different user interfaces. Adaptive (or reflective) systems [3] take advantage of the ability to inspect their own state and adapt their configuration to suit a particular environment or the demands of the user. This act in itself gives the basis for such a system to implement self-management.

Figure 2 shows the functional description of e-Viz:

- The system and task simulation service layer performs a simulation using descriptions of the available hardware and proposed (or used) system pipeline. The aim of this service layer (called Simu-Vis) is to

i e-mail: chughes@informatics.bangor.ac.uk

provide optimised system pipelines for various types of visualization problems.

- All knowledge and information is stored in a database that can be queried using particular data types, in order to provide a valid pipeline.
- This pipeline is encoded using a formal pipeline description language, from which the user interface and system pipeline can be generated.
- During execution, the interface, formal description and system pipeline can be adapted to meet the user requirements.

3. Demonstrators

We have evaluated and employed three demonstrators to test the e-Viz framework. The first is an environmental pollution disaster simulation. In this scenario an accidental release of a chemical has taken place and it is necessary to compute in faster than real time the concentrations of the pollutant in the environment to inform decisions with respect to evacuation of population centres. The pollutant is moved under the action of the wind which may change as time progresses and alter the levels of concentration in different locations. This scenario is modelled using a PDE based numerical simulation generating data that is visualized and presented to the user in real time. The user is able to change the direction of the wind while the simulation is running and see the effects of these changes as they are computed.

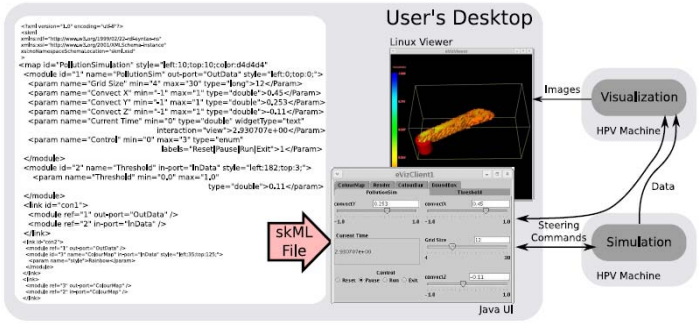

Figure 3: Environmental Pollution Demonstrator

In the e-Viz framework the system is responsible, using high level directives from the user, for generating an appropriate visualization pipeline. This is represented using skML, an XML based language used to represent the formal pipeline description. [4] The various components that make up this pipeline can then be launched and connected. This skML description is passed to the e-Viz client which parses it to dynamically create an appropriate user interface – see Figure 3. For each parameter it selects an appropriate widget; the selection is initially based on its type but can be refined on the basis of special widgets specified within the skML, or by local user preferences or by the nature of the client device. skML snippets are also used by the simulation component to influence the state of its user interface to change the state of its initialisation parameters from active to disabled while running and back to active upon reset.

The second demonstrator implements an Augmented Reality interface for Transcranial Magnetic Stimulation (TMS) that is the process in which electrical activity in the brain is influenced by a pulsed magnetic field. Common practice is to align an electromagnetic coil with points of interest identified on the surface of the brain, which can be stimulated helping researchers identify further information about the function of the brain – see Figure 4. The cranium is remotely rendered using high performance visualization resources and the compression used on the image stream to the local client is adapted to maintain real-time performance regardless of network latency.

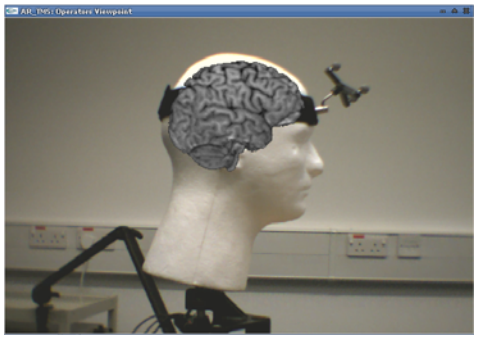

Figure 4: Augmented Reality Demonstrator

The third demonstrator is an anatomical atlas volume rendering demonstrator that offers a degree of self-healing by monitoring active visualization components through agents. If one component is lost due to a software failure or network problems, the monitoring agents forward the information to the associated pipeline, which can then switch to other similar running components (if they exist) or request new ones. Different strategies can be chosen to deal with potential failures: the pipeline can for example use twice as many components as needed to offer true redundancy, or can use a pooling strategy. Performances may be degraded in the case of failure, but the visualization will not terminate, as images will continue to be produced. If no redundancy strategy is in place for a pipeline, a failure will be translated into a service interruption on the client side, but a replacement component will be requested and the visualization will thus restart automatically. Each strategy is associated with a certain cost, so the user can choose to trade-off reliability and frame rate with cost, and e-Viz automatically can allocate and manage the pool of rendering agents. Self-optimization is achieved by using SimuVis [5] - a task simulation engine provided within the e-Viz environment.

4. Conclusion

By exploring the Autonomic Computing approach, we have demonstrated how it can impact on the way that visualization services are implemented and presented to users and developers. We have taken our proposal of a deployment model for a visual supercomputing infrastructure, and have carried out a thorough analysis of how it can be achieved with a functional implementation.

References

[1] IBM. Autonomic computing website (Last visited 29th June 2006), http://www.ibm.com/autonomic/about.shtml.

[2] K. Brodlie, J. Brooke, M. Chen, D. Chisnall, A. Fewings, C. Hughes, N.W. John, M.W. Jones, M. Riding, and N. Roard. Visual supercomputing – technologies, applications and challenges. Computer Graphics Forum, 24(2):217–245, 2005.

[3] F. Kon, F. Costa, G. Blair, and R. H. Campbell. The case for reflective middleware. Communications of the ACM, 45(6):33–38, June 2002.

[4] K. Brodlie, D. Duce, J. Gallop, M. Sagar, J. Walton, and J. Wood. Visualization in grid computing environments. In Proceedings of IEEE Visualization 2004, pages 155–162, 2004.

[5] D. Chisnall and M. Chen, The Making of SimEAC, 3rd IEEE International Conference on Autonomic Computing, Dublin, Ireland, 2006.