



**IP-FP6-015964**

**AEOLUS**

**Algorithmic Principles for Building Efficient Overlay Computers**

**Deliverable D6.2.13**

**JOpera software component to support computational overlays**

---

Responsible Partner: ETHZ (CH)  
Report Preparation Date: February 2010

Contract Start Date: 01/09/05 Duration: 54 months  
Project Co-ordinator: University of Patras (EL)

This is a software deliverable and consists of the prototype software JOpera that supports computational overlays [8]. Its design and architecture have been described in detail in Deliverable D3.4.2. Details for installing and running the software are provided in appendix.

Using JOpera, developers compose Grid services into workflows which are then automatically published as Grid services. The workflows are visually composed out of different heterogeneous tasks (mixing coarse grained Grid and Web service invocations with fine grained Java snippets) which are linked by a control flow and a data flow graph [9]. The control flow defines the partial order of invocation of the tasks while the data flow is a directed graph which defines the data to be copied between the tasks, i.e., what data is copied between output and input parameters of tasks. For each execution, a new instance of a workflow is created. The runtime state of a workflow instance consists of all data associated with the execution. This includes all the values of the input and output parameters of the tasks and the workflow, as well as process and task attributes written by the execution engine (e.g., execution status, debugging, profiling and lineage tracking information, and other execution related metadata). The state of the computation can be stored persistently in a database.

Additionally, the system incorporates autonomic computing principles [6] such as self-configuration, self-optimization and self-healing [3]. The system employs an autonomic controller in order to remove the need for manual configuration. This controller monitors the current workload and state of the system. It uses this information to determine whether the system is running in the optimal configuration or, alternatively, whether reconfiguration actions [7] have to be carried out. To define the behavior of the autonomic controller, we have developed basic policies [10]. These policies can be chosen according to different goals (e.g., minimize resource allocation or minimize response time). While such policies define how the autonomic controller reacts to changes in the workload by adapting its configuration, its exact behavior depends on configurations using thresholds. Using such threshold-based policies is only a partial solution to the self-configuration problem: if it is difficult to set the optimal configuration of a system, optimally configuring an autonomic controller with such thresholds is an even harder problem [1]. We have extended JOpera so that it can be replicated across a cluster to allow for scalable workflow execution but it still remains hard to find a suitable configuration of the distributed execution engine for a given workload. To this aim, we have developed requirements for an autonomic controller [4] which will adapt the engine's configuration to characteristics of the current workload.

For performance evaluation purposes, JOpera has been deployed on a cluster of up to 20 nodes. Each node is a 1.0GHz dual P-III, with 1 GB of RAM, running Linux (Kernel version 2.4.22) and Sun Java Development Kit version 1.4.2. One additional node was allocated to run the global tuple space server, running IBM T-Spaces v2.1.3 [5]. The evaluation in the context to the JOpera autonomic workflow engine has shown the feasibility of using zero-configuration policies for realistic workloads. Not only the proposed policies do not require any manual configuration, but they provide a significant performance gain over simpler policies based on thresholds, even when these are optimally tuned [2]. As our experiments indicate the autonomic controller of JOpera can adapt the system configuration optimally to unforeseeable, changing workload characteristics. The system furthermore takes failures into account and adapts the system's configuration accordingly. The source code of the prototype together with a slide presentation and instructions for running it

are available at <http://aeolus.ceid.upatras.gr/files/AEOLUS-DEMO-CD.rar>

## References

- [1] D. Breitgand, E. Henis, and O. Shehory. Automated and Adaptive Threshold Setting: Enabling Technology for Autonomy and Self-Management. In ICAC 05: Proceedings of the Second International Conference on Automatic Computing, 2005.
- [2] T. Heinis. Workflow-based services: infrastructure for scientific applications. PhD dissertation, ETH, 2009.
- [3] T. Heinis, C. Pautasso, and G. Alonso. Design and Evaluation of an Autonomic Workflow Engine. In ICAC 05: Proceedings of the Second International Conference on Automatic Computing, 2005.
- [4] T. Heinis, C. Pautasso. Automatic Configuration of an Autonomic Controller: An Experimental Study with Zero-Configuration Policies. In Proc. of the 5th IEEE International Conference on Autonomic Computing (ICAC 08), IEEE Computer Society, pp. 67-76, 2008.
- [5] IBM. TSpaces. <http://www.almaden.ibm.com/cs/Tspaces/>.
- [6] J. Kephart. Research Challenges of Autonomic Computing. In ICSE 05: Proceedings of the 27th International Conference on Software Engineering, 2005.
- [7] J. Parekh, G. Kaiser, P. Gross, and G. Valetto. Retrofitting Autonomic Capabilities onto Legacy Systems. *Cluster Computing*, 9(2):141-159, 2006.
- [8] C. Pautasso. JOpera: Process Support for more than Web services. <http://www.jopera.org>.
- [9] C. Pautasso and G. Alonso. The JOpera Visual Composition Language. *Journal of Visual Languages and Computing*, 16(1-2):119-152, 2004.
- [10] C. Pautasso, T. Heinis, and G. Alonso. Autonomic Execution of Web Service Compositions. In ICWS 05: Proceedings of the 3rd IEEE International Conference on Web Services, 2005.