

## *REP report*

Robert Lovas (MTA SZTAKI) visiting INRIA - Sophia Antipolis

May-June 2008 (2 weeks)

The main aim of this visit was to investigate and outline some methods and mechanisms that enable the use of advanced debugger and testing tools as a part of integrated development environments for Grid applications. During the visit, we exchanged our experiences between INRIA and MTA SZTAKI teams at several face-to-face meetings and live demonstrations on tools involving Arnaud Contes, Eric Madelaine, and Denis Caromel from the hosting institute. Additionally, one seminar was held on June 3, where more than 20 team members from OASIS attended (Robert Lovas: 'P-GRADE Grid Portal & gUSE environment for developers and end users of grid applications'). The parties agreed on further meetings at various events; such as DAPSYS '08 Conference in Hungary, or Grids@WORK in France.

In this 2-week REP visit, we studied jointly some feasible ways towards the integration of high-level debugger/testing tools in ProActive (from INRIA) and P-GRADE portal/gUSE (from MTA SZTAKI) based on the macrostep-based debugging method (for P-GRADE environment and HARNESS metacomputing framework), as well as on the behavioral modeling of distributed fractal components, and the IC2D monitoring/debugging tool for Proactive.

The first objective was to study and also outline new ideas that enable the interactive debugging of remote sequential entities in the integrated development environments; ProActive and P-GRADE Grid Portal/gUSE. Since both environments support various high-level programming styles (SPMD, Master/Worker, etc.), a plan of unified approach seems to be feasible but we have to take into consideration the differences as well. In term of dynamicity, the major difference between Proactive and gUSE is that the Proactive system provides more dynamic environment for communicating entities (Active Objects) than in case of gUSE's entities (jobs, services, or local applications). That is why, two different solutions are available for developers in the two system. In case of Proactive, the IC2D tool provides sophisticated run-time monitoring and control facilities. In case of gUSE, static workflow descriptions of orchestrated jobs/services are available with run-time job status indicators, and basic control facilities. However, in case of a debugging session, both tools can be used for navigation among entities (to be debugged). The new elements of graphical user interface can be easily added to both systems; as a new workspace of IC2D for Proactive tool, or as a new portlet of the GridSphere based WS-PGRADE portal in the gUSE environment. The underlying mechanism can be extended by remote interactive debugging facilities e.g. based on the HARNESS metadebugger concept;

1. As presumptions;
  - a. The Java and C/C++ source code (to be debugged) are recompiled with appropriate debug flags.
  - b. The hosting environments (JVM, service containers, etc.) launched with remote debugging flags.
2. Each major (re)configuration, service deployment, and 'job start'-like event is captured by the existing monitoring functionalities of gUSE and ProActive.

3. The appropriate sequential debugger are attached to the remote entity
  - a. GNU Debugger for C/C++ code with local debug agent
  - b. Eclipse Rich Client Platform (RCP) / Java Management Extensions (JMX) / Java Platform Debugger Architecture (JDPA) for Java remote objects/services
4. High level/switchable views for user interactions are developed or integrated in the framework; e.g. to browse the source code, to put breakpoints, etc.
5. In order to avoid firewall and security issues, communication with the remote entities is based on
  - a. the secure communication facilities of the given framework, or
  - b. the secure communication channel provided by the third-party sequential remote debugger tool, and/or a local debugger agent.

Interactive debugging is manageable until certain level of complexity. Large SPMD, Master/Worker, or Parameter study applications must be scaled down in this approach in order to observe efficiently the most representative entities of the application.

The above described approach can allow the remote interactive debugging of user-defined sequential code segments. As the second objective we step further; relying mainly on the macrostep-based debugging method and the component behavioral modeling, we studied some feasible ways of elaboration/integration of high-level debugger/testing tools in ProActive and P-GRADE Grid portal/gUSE.

In case of Proactive, now the main focus is on the metaobjects of Active Objects, and the non-functional part of components. Metaobjects are responsible for the transparent implementation of several features of ProActive system, such as migration, fault tolerance, security, handling (queuing) method invocations, SPMD programming paradigm, etc. Moreover, these metaobjects can be customized by the programmers, i.e. they can be a possible source of programming bugs in a non-deterministic execution environment, which cannot be easily replied. On the other hand, in GCM (Fractal and Grid Component Model) models (supported by ProActive), adaptation mechanisms to the execution environment are triggered by the non-functional part of the components. Interactions with execution environments may require complex relationships between controllers. In a non-deterministic execution environment, the non-functional part of components can be also possible source of programming bugs, which are hard to redetect/reply by the user without an appropriate logging and supervisory mechanism.

In case of gUSE, hard-to-reproduce errors may occur at another level due to the changes in the run-time environment (middleware failures, unavailability of previously allocated hosts, etc.) but the independent data sources can be also interesting; file catalogues, storage elements, and databases, since their content and access can be changed execution-by-execution as well.

In both cases, the necessary supervisory mechanism must detect and log the events (which are relevant during the re-execution) in order to ensure or simulate the same environment in the reply phase.

The parties have been studying debugger/tester methods for service/component oriented Grid solutions, which could be adopted in other development framework as well. The results are planned to describe in more details as a joint technical report in the frame of Coregrid Sustainability.