

Kepler: An Extensible System for Design and Execution of Scientific Workflows

Ilkay Altintas¹, Chad Berkley², Efrat Jaeger¹, Matthew Jones², Bertram Ludäscher¹, Steve Mock¹

¹San Diego Supercomputer Center (SDSC), University of California San Diego

²National Center for Ecological Analysis and Synthesis (NCEAS), University of California Santa Barbara
{berkley, jones}@nceas.ucsb.edu, {altintas, efrat, ludaesch, mock}@sdsc.edu

1. Background

Most scientists conduct analyses and run models in several different software and hardware environments, mentally coordinating the export and import of data from one environment to another. The Kepler scientific workflow system provides domain scientists with an easy-to-use yet powerful system for capturing *scientific workflows* (SWFs). SWFs are a formalization of the ad-hoc process that a scientist may go through to get from raw data to publishable results. Kepler attempts to streamline the workflow creation and execution process so that scientists can design, execute, monitor, re-run, and communicate analytical procedures repeatedly with minimal effort. Kepler is unique in that it seamlessly combines high-level workflow design with execution and runtime interaction, access to local and remote data, and local and remote service invocation.

SWFs are superficially similar to business process workflows but have several challenges not present in the business workflow scenario. For example, they often operate on large, complex and heterogeneous data, can be computationally intensive and produce complex derived data products that may be archived for use in re-parameterized runs or other workflows. Moreover, unlike business workflows, SWFs are often dataflow-oriented as witnessed by a number of recent academic systems (e.g., DiscoveryNet, Taverna and Triana) and commercial systems (Scitegic/Pipeline-Pilot, Inforsense). In a sense, SWFs are often closer to signal-processing and data streaming applications than they are to control-oriented business workflow applications.

2. Ptolemy II Features

Kepler builds upon the mature, dataflow-oriented Ptolemy II system (Ptolemy) [3]. Ptolemy focuses on visual, module-oriented programming with an emphasis on multiple component interaction semantics. Ptolemy precisely controls the execution model of a workflow via so-called *directors*. To the best of our knowledge, Ptolemy is the only available system which allows one to *plug in* different execution models into workflows. Individual workflow steps are implemented as reusable *actors* that can represent data sources, sinks, data transformers, analytical steps, or arbitrary computational steps. An actor can have multiple *input* and *output ports*, through which streams of data tokens flow. Additionally, actors may have *parameters* to define specific behavior.

Ptolemy can perform both design-time (static) and run-time (dynamic) type checking on the workflow and data. Kepler inherits and extends these advanced features from Ptolemy, adding numerous new features for scientific workflows.

3. Kepler Features

Using Kepler, scientists can capture workflows in a format that can easily be exchanged, archived, versioned, and executed. Both Kepler's intuitive GUI (inherited from Ptolemy) for design and execution, and its actor-oriented modeling paradigm make it a very versatile tool for SWF design, prototyping, execution, and reuse for both workflow engineers and end users. Kepler workflows can be exchanged in XML using Ptolemy's own Modeling Markup Language (MoML). Kepler actors run as local Java threads by default (from Ptolemy), but are extended to spawn distributed execution threads via web and Grid services, as well as through Java's foreign language interface (Java Native Interface). Kepler currently provides the following features:

Prototyping workflows: Kepler allows scientists to prototype SWFs before implementing the actual code needed for execution. Kepler's *design actor* can be seen as a "blank slate" which prompts the scientist for critical information about an actor, e.g., the actor's name, and port information. Once the user has prototyped an actor in this way, a corresponding stub is compiled at run-time and added to the user's library. The user can then use this stub on the workflow canvas to prototype a workflow.

Distributed Execution (Web and Grid-Services): Kepler's web and Grid service actors allow scientists to utilize computational resources on the net in a distributed scientific workflow. Kepler's generic *WebService actor* provides the user with an interface to seamlessly plug in and execute any WSDL-defined web service. For conveniently plugging in a whole set of (possibly related) services, a *web service harvester* has been developed. It can be used to instantaneously import all web services found on a web page or in a UDDI repository. In addition to generic web services, Kepler also includes specialized actors for executing jobs on the Grid, e.g., actors for certificate-based authentication (*ProxyInit*), grid job submission (*GlobusJob*), and Grid-based data access (*DataAccessWizard*, *GridFTP*). Third-party data transfer on the Grid can be established using the *EcoGrid* actor with Ecological Metadata Language (EML) support or through *GridFTP* and *SRB* (Storage Resource Broker).

Database Access and Querying: Kepler includes database actors, e.g., *DBConnect* which emits a *database connection token* (after user login) to be used by any down-stream *DBQuery* actor that needs it.

Other Execution Environments: Supporting foreign language interfaces via the Java Native Interface (JNI) gives the user flexibility to reuse existing analysis components and to target appropriate computational tools. For example, Kepler (through Ptolemy) already includes a Matlab actor and a Python actor. Further actors for execution of SAS and R (S+) code will be added.

Other Features: Kepler includes a suite of *data transformation actors* (XSLT, XQuery, Perl, etc.) for linking semantically compatible but syntactically incompatible web services together. Also included is the *Ecological Metadata Language (EML) ingestion* actor to access and download EML described sources. The *EML ingestion* actor allows Kepler to import a multitude of heterogeneous data, making it a very flexible tool for scientists who often deal with many data and file formats. The *browserUI* actor is used for injecting user control and input, as well as output of legacy applications anywhere in a workflow via the user's web browser. The Kepler SWFs can also be run in batch mode using Ptolemy's background execution feature. A feature to allow the user to configure his/her own actor libraries via a web interface is also being implemented.

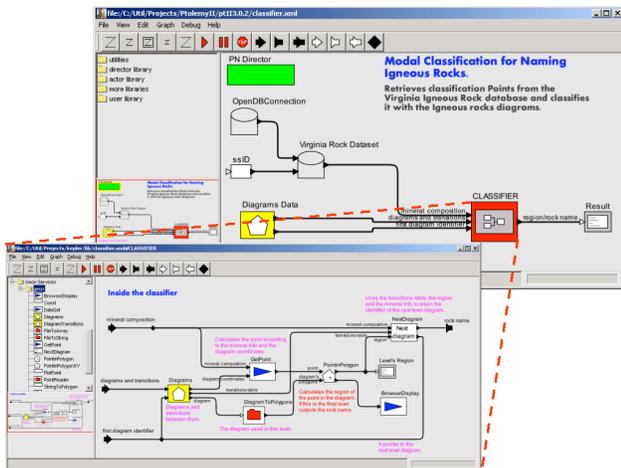


Figure 1. An example Geoscience Workflow in Kepler

The Kepler scientific workflow system has been used to design and execute various workflows in biology, ecology, geology, astrophysics and chemistry; see Fig.1 for an example workflow for rock type classification.

4. Related Systems

Other scientific workflow environments include academic systems, for example, SCIRun [7], Triana [8], Taverna [9], and commercial systems (Scitegic/Pipeline-Pilot, Inforsense). SCIRun has extensive support for large-scale simulation and data visualization. SCIRun,

Triana, and Taverna are all based on a single dataflow execution model, while Ptolemy/Kepler allows many. Triana supports distributed execution and interfaces with the Gridlab Application Toolkit. In Taverna, all computational workflow steps are web services. The system provides tools for web service harvesting and execution and numerous other novel features, e.g., automated insertion of iterators.

5. Implementation Status and Next Steps

Kepler inherits many advanced features such as variable (director-based) execution models, nested workflows and the Vergil GUI from the underlying Ptolemy system. Since Ptolemy is open source and comes with excellent and comprehensive documentation, the Kepler team [2] was able to add numerous novel features within a relatively short period of time. The extensions are driven by actual needs for scientific workflows [5] in scientific application projects such as GEON [1], SEEK [4], and SciDAC/SDM [10]. The source code is freely available through the project site [2], and the first official pre-release, packaged for end users is planned for May 2004. For additional information on Kepler see [6].

Acknowledgments.

Kepler includes contributors from SEEK [4], SDM Center [10], Ptolemy II [3] and Geon [1]. Work supported by NSF ITRs 022567 (SEEK), 0225673 (GEON), DOE DE-FC02-01ER25486 (SciDAC/SDM), and DARPA F33615-00-C-1703 (Ptolemy).

References

[1] GEON: Cyberinfrastructure for the Geosciences, <http://www.geongrid.org>
 [2] Kepler: An Extensible System for Scientific Workflows, <http://kepler.ecoinformatics.org>
 [3] Ptolemy II, <http://ptolemy.eecs.berkeley.edu/ptolemyII/>
 [4] SEEK: Science Environment for Ecological Knowledge, <http://seek.ecoinformatics.org>
 [5] I. Altintas *et al.*, A Modeling and Execution Environment for Distributed Scientific Workflows, SSDBM'03, MIT/Cambridge.
 [6] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, S. Mock, Kepler: Towards a Grid-Enabled System for Scientific Workflows, Workflow in Grid Systems, GGF10, Berlin, 2004.
 [7] SciRUN <http://software.sci.utah.edu/scirun.html>
 [8] Triana <http://www.triana.co.uk>
 [9] Taverna <http://taverna.sourceforge.net>
 [10] SciDAC/SDM: Scientific Data Management Center, <http://sdm.lbl.gov/sdmcenter/>