Development of Kepler/CORE – A Comprehensive, Open, Reliable, and Extensible Scientific Workflow Infrastructure

Summary

The Kepler scientific workflow system is the product of a grass-roots collaboration of research projects primarily funded by NSF and DOE to apply workflow automation to a broad range of scientific disciplines. Kepler's strengths relative to other workflow systems and its promise for catalyzing future research projects derive from this multidisciplinary origin. However, prior to the Kepler/CORE effort described here, no project had been funded specifically to coordinate development of Kepler with the goal of making it a well-engineered software product with the system functions and attributes required for broad adoption and long-term sustainability. Kepler/CORE, in contrast, is funded to take responsibility for developing and maintaining those features of the system that are needed by *all* scientific domains and across the various projects, while at the same time increasing the role of the current and future user community in specifying requirements and priorities.

The Kepler/CORE project aims to reengineer and significantly enhance Kepler as follows. To better serve current and envisioned user communities, the system must be *independently extensible* by groups *not* directly collaborating with the team that develops and maintains the core capabilities of Kepler. Facilitating extension in turn requires that the Kepler architecture be *open* and that the mechanisms and interfaces provided for developing extensions be well designed and clearly articulated. For Kepler to serve as a viable starting point for developing workflow-oriented applications, and as middleware for developing user-oriented scientific applications, Kepler must be *reliable* both as a development platform and as a run-time environment for the user. Finally, to fulfill the greater promise of accelerating scientific discovery, Kepler must represent a *comprehensive* system with first-class support for managing data passing through and between workflow runs, decoupling workflow definitions from the specific technologies used to provide run-time services, and for providing a rich environment to manage data, workflows, and results in the context of specific projects.

Key to our development plans will be the active participation of the greater scientific community. Kepler/CORE team members will meet periodically with a council of current *Kepler stakeholders*, representing communities already committed to applying Kepler to specific research domains, as well as with potential future users in additional scientific domains, to gather requirements, ascertain relative priorities of needed features, and coordinate contributions by members of the greater Kepler collaboration. The Kepler stakeholders council also will evaluate potential business models for sustaining development and maintenance of the Kepler system beyond the funding period of this proposal.

The Kepler scientific workflow system is already used in a large number of diverse projects. We expect that our collaborative, open-source efforts will greatly enhance each of these projects by providing a reliable, core set of functionality, and enable individual projects to focus on and disseminate those additional capabilities needed for the particular domains and research questions they address.

1 Introduction

Despite enormous advances in information technology, the state-of-the-art in scientific data analysis and management is often rather bleak. Scientists use a variety of mostly isolated tools including specialized, standalone desktop applications and web-based programs. Data management is usually done in an *ad hoc* and piecemeal fashion using a mix of spreadsheets, text files, and relational databases. Data transfer between different applications is often done manually by the user via the notoriously error-prone and non-reproducible "copy-and-paste method." Use of plain text files and paper- or file-based lab noteboks is still widespread. For data-intensive and/or computationally-intensive scientific applications, "expert users" (*a.k.a.* programmers, typically non-scientists) are employed to develop one-off, custom solutions such as specialized Perl or Python scripts. While addressing to some extent the need for automation, such scripts: (i) are too low-level to be "aware" of the scientific processes, data, and metadata they operate on; (ii) are hard to comprehend, extend, share, document, and maintain in a collaborative setting; and (iii) produce analysis results whose provenance is hard if not impossible to track with the result that "debugging" automated processes and interpreting unexpected results is an unnecessarily difficult task.

Cyberinfrastructure developments in general, and *scientific workflow systems* [3, 13, 28, 33, 30, 11, 22, 32, 15] in particular, address many of the issues mentioned above, and aim to increase both the scientist's and the developer's effectiveness and efficiency in their respective roles. Scientific workflow systems allow the scientist to harness underlying cyberinfrastructure (*e.g.*, Grid middleware, metadata catalogs, and data federations) by defining a high-level, comprehensible, shareable, and executable model of a scientific process. Features envisioned for scientific workflow systems generally include: (i) means to discover, access, and organize data; (ii) high-level descriptions of analyses and data flows; and (iii) end-to-end support for scientific computing and data management. The latter includes capabilities to deal with different service and computing environments, integrated management of workflows and workflow products (datasets, databases), and provenance management.[1]

A scientific workflow ties together various software components that often have been developed independently and that may be deployed and invoked in different ways, *e.g.*, as a web service, as a local application that can be invoked from the command line, or—as in the case of database systems or other complex software packages such as R and Matlab—via custom APIs. A scientific workflow system should provide a uniform abstraction and invocation mechanism for all such components. For example, in Kepler [19], so-called *actors* are used as the uniform software components and principal building blocks of a scientific workflow. For a scientific workflow system to be most effective and useful to one or more communities, users should be able to readily share and reuse data, workflows, workflow results and their provenance, and—last but not least–workflow system extensions themselves (*e.g.*, new actors).

1.1 The current Kepler collaboration and system

The Kepler scientific workflow system [19, 4, 21] is a general purpose, multi-disciplinary, and open-source[2] computing environment for modeling, executing, and managing scientific processes and resources. Figure 1 gives two example workflows, one for infering phylogenetic trees and another for species prediction, implemented within Kepler. The Kepler system grew out of an informal collaboration between researchers and developers funded under the NSF/SEEK [26] and DOE/SDM [1] projects. Both projects required scientific workflow automation techniques, albeit for rather different application domains (including ecoinformatics, bioinformatics, and astrophysics). This informal collaboration chose to build Kepler upon the well-documented and open-source Ptolemy II

[1]Data and workflow provenance capture the processing history and evolution of data products and workflow descriptions, respectively. Data provenance is crucial for understanding and, if necessary, re-running, single-stepping, or otherwise "debugging" analysis results; workflow provenance is an important part of overall workflow life-cycle management.

[2]Kepler is distributed under the BSD open-source license.

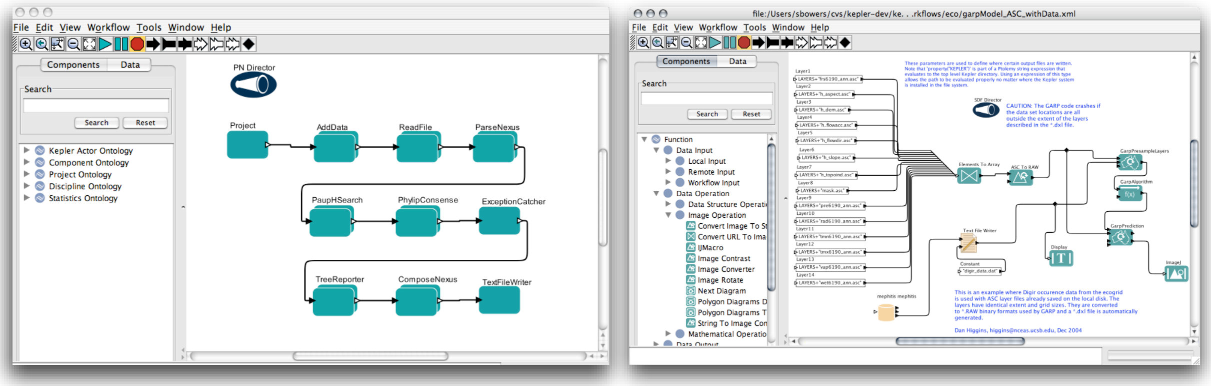

Figure 1: Two example Kepler workflows: (left) a top-level view of a bioinformatics workflow for inferring phylogenetic trees from molecular sequence data; and (right) a top-level view of an ecology workflow for running a Garp-based ecological niche model. Many of the actors shown are composites, i.e., containing sub-workflows.

system [2, 12]. Originally developed as a heterogeneous modeling and design environment for the electrical engineering community, Ptolemy II offers many advantages such as a powerful graphical user-interface and support for different workflow execution models (*e.g.*, process networks [18, 20] for stream-based pipelined execution, nested subworkflows, and models for discrete event and continuous time simulations [16]). The GUI employed by Ptolemy II (and since extended by Kepler) offers a number of benefits over commercial scientific workflow systems (such as SciTegic's Pipeline Pilot) that are more narrowly focussed on a particular domain, and the powerful and flexible execution models go beyond the simple DAG-based workflows often used in Grid-workflow applications. With the rapid success in building workflows based on PTOLEMY II extensions, developers and researchers from other projects (GEON [25], ROADNet [27], etc.) soon joined the informal collaboration in creating the current Kepler system.

1.2 The need for a coordinated engineering effort

Previous and ongoing projects contributing to Kepler have been successful in developing various Kepler extensions, *e.g.*, for provenance tracking, distributed process execution, ontology-based discovery and integration, and new workflow design methods, as required by the needs of each project. However, because of the informal nature of the Kepler collaboration, there has been *no coordinated effort to define a single, unified architecture and vision for the Kepler system*, *i.e.*, with a clearly defined kernel of capabilities applicable to all projects; well-defined extension points with support for backwards compatibility; system stability ensured by rigorous software testing; and an engineering approach capable of delivering and supporting regular software releases. Quoting one of the many PIs and Kepler supporters who have adopted the system for their projects [31]:

> *I believe the power of Kepler is that it was developed to serve a wide variety of applications simultaneously and therefore has emerged as a generic component of the nation's emerging cyberinfrastructure. However, Kepler's very success has created a problem in that with so many projects adopting Kepler and creating local improvements, there needs to be a second generation effort to create a robust and extensible software infrastructure that can continue to support the growing number of e-Science adopters.*

This project aims to fill this gap in the overall Kepler effort. In collaboration with current and future contributors to Kepler we plan to build upon the products of the efforts described above; to take responsibility for the essential, interdisciplinary software components of Kepler; and to facilitate further Kepler development by standardizing interfaces for implementing system extensions (*e.g.*, for data and metadata management, provenance tracking, and project-specific deployment scenarios). Specifically, we propose to improve and enhance the Kepler scientific workflow system to yield **Kepler/CORE**, a **c**omprehensive, **o**pen, **r**eliable, and **e**xtensible scientific workflow infrastructure suitable for serving a wide variety of scientific communities. Our Kepler/CORE team, which includes the leads and technical drivers for Kepler research and development in several large and medium scale projects, proposes to take responsibility for developing and maintaining those features of the system that are needed by all scientific domains, while at the same time increasing the role of the current and future user community in specifying requirements and priorities.

2 Development Plan

The long-term success of Kepler as a multidisciplinary, scientific computing platform depends critically on widespread contribution to the effort. It is far too difficult for a single development team to provide all of the capabilities required to satisfy the needs of all potential users. Specific communities frequently require *disparate capabilities* that depend on numerous, domain-specific methodologies, applications, and data formats. Furthermore, the requirements of different disciplines often *conflict* even for fundamental aspects of the system, including how data should be managed, how the system should be deployed and configured, and what level of information-technology competence should be expected of users. Widespread, multidisciplinary adoption thus requires that members of specific communities contribute specialized actors and system extensions suitable to their own communities.

For these reasons, the goal of Kepler/CORE is not to address the specific requirements of one or more specialized communities, but to: (i) enable multiple groups in a number of distinct disciplines to easily create, support, and make available domain-specific Kepler extensions; (ii) better support those crucial features that are needed by all disciplines; and (iii) provide for the wide range of deployment scenarios required by different disciplines and distinct research settings. Our strategy will be to make the significant architectural, design, and functional improvements required to make Kepler a *comprehensive*, *open*, *reliable*, and *extensible* (CORE) scientific workflow infrastructure. These key Kepler/CORE system attributes are described further in the following subsection.

2.1 Kepler/CORE system attributes

Independently Extensible. Current contributors to Kepler are funded by a number of distinct development and research projects. These developers contribute new features and actors needed for their respective projects to a single source-code repository shared by all projects employing Kepler. Developers currently work together closely to ensure basic interoperability of contributions and stability of the Kepler system. Even so, the resulting contributions are typically *ad hoc* (*e.g.*, for the purpose of addressing a specific need of the corresponding project), and do not necessarily follow uniform conventions for managing data, dealing with external services, or configuring workflows. The result is a system that is monolithic in the sense that it is built and deployed as a single product, but discordant in that there is no guarantee that all contributions interoperate in a semantically meaningful way. Rather than enforcing conventions that might slow progress in the various disciplines contributing to Kepler, we propose to further enable independent extensibility of Kepler while making it easy to package domain-specific contributions in a way that ensures both the stability of the overall system and clearly indicates what components are expected to work well together. More specifically, we plan to divide Kepler into a minimal set of mandatory functional components (the *Kepler kernel*); a set of extensions representing optional non-actor functionality and communicating with the kernel via well-defined and generic extension interfaces; and a number of actor packages for distinct disciplines (see Figure 2). This approach will enable other development teams to freely develop new extensions and actor packages as needed without endangering the stability of the kernel, and even to replace standard extensions as needed.

Reliable. System reliability is essential to the success and widespread adoption of Kepler. Kepler/CORE must be reliable both from the point of view of developers and users. For developers, point Kepler must be a reliable development platform. In particular, effective extensibility requires that the system being extended not change in ways that break previously developed extensions. Consequently, developers of new Kepler extensions and actors must be able to rely on features to remain stable over time, and new releases of Kepler/CORE should not require significant refactoring of extensions maintained by other groups. Our proposed approach of splitting Kepler into a minimal kernel and a set of standard extensions and actor packages will enable us to achieve this stability.

Kepler also must be reliable for end-users, *i.e.*, stable and robust to common faults at run time. This run-time reliability is particularly important when Kepler is used not simply as a desktop research application, but also as middleware that other domain-specific applications can be built upon. Reliability for developers and users alike will ensure that Kepler can be applied confidently as dependable cyberinfrastructure.

Open Architecture, Open Project. We propose an open system architecture and an equally open project management approach. First, the various kernel interfaces required to achieve sufficient extensibility must be clear to engineers extending Kepler, and a consistent, uniform design across the entire product will be essential to facilitating this understanding. We will take responsibility as a group for coordinating the design of the overall Kepler/CORE product architecture, providing uniform extension interfaces and APIs to make the system easy to understand and extend, and clearly documenting the nature and applicability of the kernel extension points. Second, the Kepler/CORE project will be open in its interaction with the user community. We will disseminate plans, designs, and system documentation as we develop them and provide mechanisms for suggestions and feedback throughout the course of the project. We will also actively engage the user community and gather requirements, advice, and feedback on priorities, both from those already committed to using Kepler (*i.e.*, the Kepler "stakeholders"), and from scientists who could benefit from scientific workflows but who are not currently using workflow automation technology.

Comprehensive (End-to-End) System. Scientific workflow systems will be broadly adopted only when they become sufficiently comprehensive in scope. This project proposes to widen the scope of Kepler not only through independent extensions as decribed above, but also by providing new, fundamental enhancements that will benefit all user communities. The general requirements for these enhancements are discussed in the following subsection.

2.2 Requirements for end-to-end scientific workflow support

Scientific workflow systems promise to accelerate research by providing scientists and developers with generic capabilities that either are currently difficult to perform using existing tools (such as automated provenance tracking and workflow distribution and optimization), or must be implemented repeatedly in each workflow (such as data access, format conversion, and remote service invocation). Kepler/CORE will help deliver on the promise of scientific workflow automation by enhancing Kepler with generic capabilities for *data, service, and workflow management.* Many of the tasks related to these three areas are partially supported in Kepler today, often via special-purpose actors developed for particular disciplines. Other functions currently must be performed entirely outside of the Kepler system or are impossible to carry out. The significance of the general capabilities planned for Kepler/CORE are described in the remainder of this subsection, and the specific enhancements we will deliver are described in the following subsection.

Data Management. While Kepler currently facilitates the flow of data between actors in a workflow, many other important data management tasks are not handled transparently by the workflow execution framework. Rather, these data management tasks typically are supported via special-purpose actors, *e.g.*, to control and manage the flow of data into and out of workflows, to compare and visualize data and metadata, to handle data format conversions, and to manage data references (*e.g.*, for resolving Life Science identifiers (LSIDs) [29]). When explicit actors are used to carry out these tasks, the workflow framework becomes ignorant of these operations, thus limiting the ability of the system to automate, optimize, and flexibly execute these tasks. For example, when specialized actors are used to import and export data, it becomes difficult for Kepler to automatically determine what data was used and what results were computed by a particular workflow run. Dependable access to such knowledge represents a critical requirement for a generic provenance subsystem [8, 9]. Furthermore, using special-purpose, technology-dependent actors for data management limits Kepler's ability to automatically leverage alternative data sources (*e.g.*, having different formats or using different storage technologies) in different contexts (*e.g.*, when the workflow is shared between organizations). While a number of more generic data management approaches have been prototyped within Kepler, including collection-oriented modeling and design [24, 23] integrated data access via EcoGrid [17], and semantic data mediation [6, 10, 5], a uniform and comprehensive data management framework does not yet exist. Due to the data-driven nature of scientific analyses, supporting data management tasks in a generic way within the Kepler framework is crucial for wide-scale adoption.

External Service Management. Scientific workflow systems promise to make scientific computing specifications more declarative by separating the definition of *what* computations to carry out (the workflow definition) from *how* to carry each out (handled automatically by the framework). Declarative workflows in turn promise to be easier to understand (*e.g.*, because all actors represent scientifically meaningful tasks rather than obscure data manipulation or control-flow operations) and to reuse (*e.g.*, because workflows do not require changes when different underlying technologies or implementations of algorithms need to be applied). Workflows also make it easier to effectively use services and resources external to the automation framework. However, Kepler workflows that make extensive use of external services generally use actor-oriented approaches for managing and accessing these services similar to the approaches used for managing data. Various special-purpose actors are used for moving data and for running jobs on remote servers [7]. Kepler/CORE enhancements will better enable the system to carry out computations on the optimal set of computing resources at run time, based on resource availability and preferences; and will make it easier for users to share and redeploy workflows in different environments.

Workflow Management. Workflow management encompasses tasks related to designing, storing, and validating individual workflows; organizing workflows, data, and results within the context of a particular project or research study; and capturing and querying the provenance of workflows and data. While current systems, including Kepler, typically support these tasks to varying degrees, they generally lack comprehensive workflow-management support and focus primarily on composing, and/or running *a single workflow at a time.* Kepler is largely ignorant of the scientific context

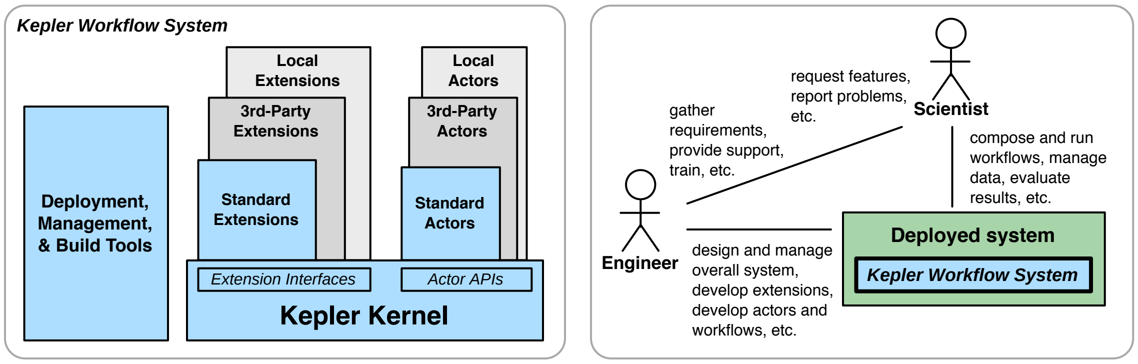
Kepler Workflow System
Local Extensions
Local Actors
3rd-Party Extensions
3rd-Party Actors
Deployment, Management, & Build Tools
Standard Extensions
Standard Actors
Extension Interfaces
Actor APIs
Kepler Kernel
request features, report problems, etc.
Scientist
gather requirements, provide support, train, etc.
compose and run workflows, manage data, evaluate results, etc.
Engineer
Deployed system
Kepler Workflow System
design and manage overall system, develop extensions, develop actors and workflows, etc.

Figure 2: Proposed high-level changes to the Kepler architecture (left), and idealized roles of scientists, system engineers, and interaction with Kepler (right)

in which workflows are being run, the flow of data through and across successive workflows (as is common in scientific research), and the origin of workflows [9]. Kepler/CORE deliverables will provide comprehensive support for end-to-end workflow management.

2.3 Kepler/CORE capabilities and deliverables

Our strategy for enhancing Kepler according to the requirements described above is to:

- Identify within the current Kepler system the minimal *kernel* of essential functional subsystems.
- Design extension points for enhancing the Kepler kernel and customizing it for use in particular research settings.
- Refactor the existing system into kernel subsystems and non-kernel components that interact with the kernel through these extension interfaces.
- Develop new components to address current limitations in data, service, and workflow management.

Figure 2 illustrates our strategy from the point of view of a research group adopting Kepler. The blue boxes indicate system components developed by the Kepler/CORE team, while the remaining boxes represent software independently developed by non-Kepler/CORE developers, either in-house (local actors and extensions) or other groups (3rd-party extensions and actors). The Kepler kernel comprises those features that cannot be replaced or overridden by extensions, while standard extensions and actors represent replaceable components provided by the Kepler/CORE team.

Design of this new Kepler architecture will be driven by the need to support current features and new capabilities required to make Kepler a more comprehensive scientific workflow environment. These new capabilities will provide support for: (1) configuring and managing Kepler installations; (2) managing project information and providing context for workflow execution; (3) decoupling the graphical user interface from workflow execution; (4) transparently storing and accessing scientific data using a variety of underlying technologies; (5) distributing workflow execution in a technology independent manner; (6) interoperating with authentication and authorization services; and (7) inserting custom services into the workflow enactment sequence. Each of these capabilities likely will be associated with one or more distinct extension points in the kernel. Design and prioritization of these new capabilities will be guided by requirements gathered from current Kepler stakeholders and from representatives of the broader scientific community. We describe these new capabilities in more detail below along with our current vision for delivering them.

Flexible configuration management. Independent extensibility requires flexible tools for deploying and configuring Kepler in a research setting. The current Kepler installer treats all included software components as mandatory. Run-time configuration information is stored in text files that must be hand-edited for non-standard installations. *Possible Deliverables: A configuration management system to support: downloading, installing, and upgrading the Kepler kernel; discovering and installing standard and 3rd-party extensions and actor packages; specifying and configuring extensions to be employed during execution; and a standard configuration store for a single-user, single-machine installation of Kepler. Third-parties will be free to develop alternative configuration stores with additional capabilities, e.g., for configuring Kepler across multiple machines and users in an organization.*

Project information management. Users commonly employ Kepler in several different contexts or distinct research projects. Users require the ability to organize and access actors, workflows, data, and results within specific projects, and to assign computing resources (and associated authentication credentials) to each. *Possible Deliverables: A project information store for persisting project definitions and associations between data and workflows within a project. The Kepler kernel will operate on data associated with the current context, and record within that context new associations between workflow results and the workflow definitions and input data used to derive them (i.e., the provenance of the results). The standard project store will be appropriate for single-user operation, but other groups will be free to develop alternative project stores and associated management tools for sharing project information between users in an organization.*

Graphical interface decoupled from the workflow engine. Different user communities and deployment scenarios often require distinct user interfaces. Furthermore, effectively exploiting distributed computing resources requires that the Kepler graphical user interface not be tightly coupled to workflow execution. Long-running workflows in particular require that users be able to detach the GUI from a workflow once it is running, and reattach to a running workflow later to monitor or steer further execution. While Kepler workflows currently may in principle be run without displaying the GUI, the Kepler workflow execution engine and graphical user interface are tightly coupled, and there is no convenient way for different user interfaces to interact with a running workflow. *Possible Deliverables: A standalone Kepler GUI that can start new workflow engines or connect to a running engine either locally or on a remote machine. Other groups will be able to develop completely new graphical user interfaces to Kepler, including domain-specific applications and web-based portals.*

Extensible workflow enactment sequence. Kepler is a powerful environment for exploring radical new approaches to modeling and automating scientific research processes. Reengineering must not put this type of research at risk. We plan to mitigate this risk and enable the development of powerful new features by making it easy to change or add to the sequence of steps involved in workflow enactment. *Possible Deliverables: A generic framework for specifying the abstract stages of workflow enactment and the concrete services to invoke for each stage. Abstract stages might include loading of abstract workflow definitions; refinement of these workflow templates to concrete workflows; validation of workflows, parameter values, and types; workflow optimization; pre-execution checks, e.g., to verify that required services are available; reservation of resources; binding* *of workflows to data sources and sinks; staging of data, actors, and subworkflows to compute nodes; execution of the workflow itself; steering of workflow execution; modifying workflows dynamically or binding to new resources as needed for fault tolerance; and storing results and releasing resources after workflow run completion.*

Technology-independent data management. Maximizing the reusability of a scientific workflow requires that the definition of what the workflow does be decoupled from the particular technologies used to manage data between workflow runs, and from the binding of the workflow engine to particular data sets it operates on during a particular run. Currently much interaction with external data stores is modeled explicitly as actors, thus requiring extensive refactoring of workflows used in new deployment contexts and making it difficult to implement comprehensive data provenance features. *Possible Deliverables: General-purpose data source and data sink components that will be loosely coupled to the technology used to store data and will definitively indicate to the system what input and output data were associated with a workflow run; a data store extension interface for transparently supporting diverse data storage systems; standard data store implementations for the Unix and Windows file systems, and standard relational database systems. Other development teams will be able to add support for additional data storage systems.*

Technology-independent distributed execution. Like data management services, distributed workflow execution services must be modeled in a technology-transparent way to make workflow definitions useful in different deployment contexts. Distribution of workflow execution currently is limited in Kepler to remote execution services modeled as explicit actors, *e.g.*, the Web Service and Globus actors. *Possible Deliverables: A general-purpose framework for distributing actors, subworkflows, or entire workflows across available processors on multiple nodes; annotation tools for specifying what parts of workflow to distribute; a distributed execution resource interface for providing resources to a Kepler installation; a default workflow distribution system based on existing open-source technologies (e.g., Globus or Condor [14]). Other groups will be able to add support for other distributed computing technologies.*

Generic authentication services Workflows that employ shared or remote services and resources require support for authenticating and authorizing users. Kepler currently has minimal support for authentication; existing workflows model authentication steps as explicit actors in the workflow and are thus tightly coupled to the particular authentication schemes employed in the local organization and/or the organizations providing the remote services. *Possible Deliverables: Integrated support for managing authentication and authorization information in workflows; extension interface for adding support for new authentication and authorization services.*

2.4 Transitioning to a new collaborative engineering approach

The majority of code currently being developed for the Kepler system is stored in a single source code repository module hosted through NCEAS, including hundreds of actors developed by the numerous contributing projects. Thus, most code developed for use *with* Kepler is also *part of* the Kepler distribution. All of this code is built and tested nightly as a monolithic unit. This approach has advantages: all developers and contributing projects benefit from a single, centrally maintained installation of collaborative development infrastructure including unified source code control, bug tracking, and communication tools; it is easy for all contributors to monitor the activities and contributions of other developers and projects and to make use of capabilities developed by everyone; and conflicts between changes, extensions, and dependencies introduced by different groups become apparent and can be resolved quickly. However, there are disadvantages: because all Kepler developers have equal access to all source code, anyone can introduce changes that affect everyone negatively; the lack of clear boundaries between different projects in the code mean that it is difficult to deliver a system that is useful to any set of users without delivering the entire system (including all actors); and it is unclear who should be responsible for what parts of the software, particularly the overall design of the system.

We plan to introduce a new engineering strategy that will preserve the advantages of the current, unconstrained approach, while eliminating its disadvantages. First, we will analyze the existing code and partition it, in cooperation with other Kepler development teams, into distinct repository modules representing common functionality required by all projects (the initial versions of the Kepler kernel, standard extensions, and standard actor packages), and code associated primarily with particular scientific domains, contributing projects, and technologies. We will continue to host code developed by other groups for use with Kepler and ensure that all such software continues to be developed in public and to the benefit of the entire scientific community. However, we will also restrict write access to particular repository modules to the development teams that take responsibility for them. Write privileges to the Kepler kernel, standard extensions, and standard actor packages will be restricted to the Kepler/CORE development team that will include developers funded by Kepler/CORE as well as additional developers contributing to this effort. We will encourage new and existing projects using Kepler to take advantage of our engineering infrastructure by creating new repository modules as needed, granting access to our bug tracking system, and performing continuous builds of each such module automatically, either against the current kernel and standard packages, or against a particular release of the Kepler/CORE packages.

A key milestone will be the first successful build of the Kepler system following this project reorganization effort. Continous and nightly builds and tests are currently carried using ant, JUnit, and CruiseControl. We will migrate these build and testing tasks to the *NMI build and test system*, and once the first set of builds and tests succeed we will begin to develop new system tests as needed to confirm that future refactoring does not break existing functionality. NMI builds and tests will cover all supported platforms (OS X, Windows, and Linux) and span resources hosted by the three participating Kepler/CORE institutions (UCD, UCSB, and UCSD).

2.5 Project Coordination and Management

The Kepler/CORE team. The general organization of the Kepler/CORE team is shown in Figure 3. Project PIs and co-PIs comprise the Kepler/CORE management group. The Kepler/CORE development group includes four software engineers including a CORE system architect and three feature engineers. The system architect will articulate the overall vision of the Kepler/CORE system and its architecture. Responsibilities will include gathering, documenting, and managing requirements; leading development of the overall architecture and design; and ensuring system architecture and design consistency. The software architect also will be responsible for overall project management tasks, facilitating communication between CORE software developers and managers (Kepler/CORE project PIs and co-PIs), setting and updating development schedules, assigning development tasks, and leading developer meetings. The CORE Feature Engineers will be dedicated to designing and developing new features and extensions and refactoring existing code. The Kepler/CORE team will meet three times a year to review project progress, discuss overall system architecture and related issues, perform required design work, and identify tasks for the following four months.

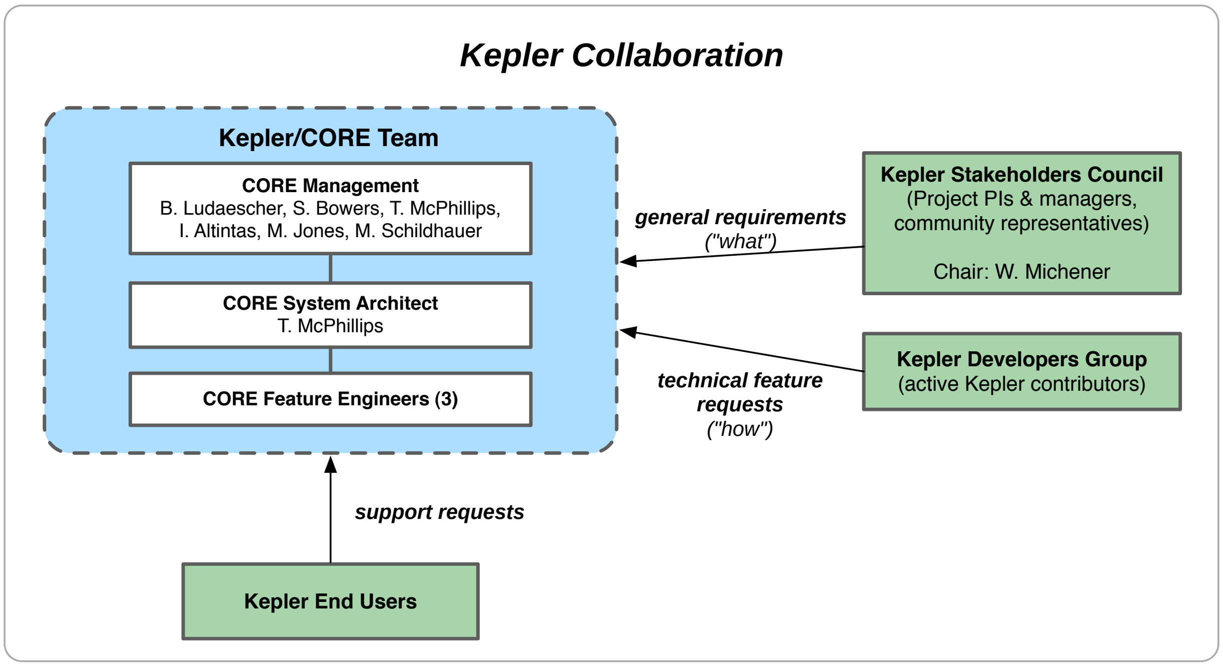
Kepler Collaboration
Kepler/CORE Team
CORE Management
B. Ludaescher, S. Bowers, T. McPhillips, I. Altintas, M. Jones, M. Schildhauer
CORE System Architect
T. McPhillips
CORE Feature Engineers (3)
general requirements ("what")
Kepler Stakeholders Council (Project PIs & managers, community representatives)
Chair: W. Michener
technical feature requests ("how")
Kepler Developers Group (active Kepler contributors)
support requests
Kepler End Users

Figure 3: The Kepler/CORE Team and interactions with the greater Kepler Collaboration and Kepler end users

Kepler Stakeholders Council meetings. All software developed under this proposal will be guided by an ongoing analysis of the needs of current Kepler stakeholders and representatives from the broader scientific community. We will hold annual Kepler Stakeholder meetings to collect requirements and feedback from authoritative representatives of existing projects using Kepler. These meetings will serve as a forum for discussing Kepler/CORE progress with the community and enabling the users of Kepler to guide future development. Each one-day stakeholders council meeting will be co-located with a Kepler developer meeting and will be open to PIs and managers for projects dependent on the success of Kepler/CORE. The meetings will be convened and facilitated by the Stakeholders Council chair. The chair also will be responsible for investigating options for sustaining the Kepler/CORE effort beyond the funding period of this grant and will use the Stakeholders Council as a forum for reporting and gathering feedback on this effort (described futher in the next section). We will provide travel support for ten participants (in addition to the Kepler/CORE project members) for each council meeting.

3 Sustaining Kepler in the Long Term

One measure of overall success for the Kepler/CORE project will be the extent to which community adoption of Kepler increases. Increased adoption will indicate that the overall Kepler effort is bringing the benefits of scientific workflow automation to a greater number of scientists and their respective communities. We believe that delivering the Kepler/CORE system attributes and technical capabilities outlined above, particularly those that will lead to Kepler becoming a dependable platform that can be easily extended by a wide range of research communities, is critical to achieving broad adoption.

The expectation of widespread adoption carries with it the responsibility to provide for the long-term sustainability of Kepler beyond the funding period of this project. At a minimum, the Kepler/CORE software must be maintained, and ideally there would continue to be a central effort to further develop the Kepler/CORE system. The issue of long-term sustainability must be addressed both organizationally and financially. We expect that the Kepler project organization proposed here will not only benefit the current Kepler/CORE effort, but will be a model for sustainability of the Kepler collaboration in the long term, ensuring that community needs will continue to be prioritized and met. If successful, the approach of engaging a council of Kepler stakeholders tasked with clearly articulating the requirements of their respective projects and providing feedback on Kepler/CORE project progress could be maintained in the long-term. Similarly, continuing to provide development infrastructure (distinct source code repository modules, automated builds and tests, and centrally managed groupware) for all groups desiring to extend Kepler would help maintain communication and synergy between different groups while avoiding conflicts between disparate requirements.

While the above approach likely could be sustained as a community effort, it would greatly benefit from additional resources for supporting new feature development and continued community outreach efforts. Among other options, we plan to investigate forming a 501(c)(3) tax-exempt corporation qualified to accept funding from a variety of sources to continue Kepler/CORE development; provide support for further community outreach, training, and project steering meetings; and generally sustain the virtual collaboration outlined in this proposal. The Kepler/CORE Stakeholder's Council, serving as representatives of the greater Kepler Collaboration, will be responsible for guiding and approving steps taken in this direction. The chair of the Stakeholder's Council will lead this effort, investigating possible business models including those employed by similar non-profit organizations; approaching an array of non-government organizations, public agencies, and other funding sources and obtaining their input on viability of these different models; assessing the applicability of these various approaches to sustaining Kepler; and presenting these options to the Stakeholders Council for discussion and approval.

References

[1] DOE/SciDAC Scientific Data Management Center, 2004. `http://sdm.lbl.gov/sdmcenter/`.

[2] Ptolemy II project and system. Department of EECS, UC Berkeley, 2006. `http://ptolemy.eecs.berkeley.edu/ptolemyII/`.

[3] A. Ailamaki, Y. Ioannidis, and M. Livny. Scientific Workflow Management by Database Management. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, 1998.

[4] I. Altintas, E. Jaeger, C. Berkley, M. Jones, B. Ludäscher, and S. Mock. Kepler: An Extensible System for Design and Execution of Scientific Workflows. In *16th Intl. Conf. on Scientific and Statistical Database Management (SSDBM)*, Santorini, Greece, 2004.

[5] C. Berkley, S. Bowers, M. Jones, B. Ludäscher, M. Schildhauer, and J. Tao. Incorporating semantics in scientific workflow authoring. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, 2005.

[6] S. Bowers, B. Ludäscher, and K. Lin. On Integrating Scientific Resources through Semantic Registration. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 349–352. IEEE Computer Society, 2004.

[7] S. Bowers, B. Ludäscher, A. H. H. Ngu, and T. Critchlow. Enabling scientific workflow reuse through structured composition of dataflow and control flow. In *Proceedings of the IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow)*, ICDE Workshops. IEEE Computer Society, 2006.

[8] S. Bowers, T. M. McPhillips, S. Cohen, and S. B. Davidson. A Model for User-Oriented Data Provenance in Pipelined Scientific Workflows. In *Proceedings of the International Provenance and Annotation Workshop (IPAW)*, volume 4145 of *LNCS*, pages 133–147, 2006.

[9] S. Bowers, T. M. McPhillips, M. Wu, and B. Ludäscher. Project Histories: Managing Data Provenance Across Collection-Oriented Scientific Workflow Runs. In *Proceedigns of the International Workshop on Data Integration in the Life Sciences (DILS)*, volume 4544 of *LNCS*, pages 122–138, 2007.

[10] S. Bowers, D. Thau, R. Williams, and B. Ludäscher. Data procurement for enabline scientific workflows: On exploring inter-ant parasitism. In *Proceedings of the 2nd International Workshop on Semantic Web and Databases (SWDB)*, volume 3372 of *LNCS (VLDB Workshops)*, 2005.

[11] L. Bright and D. Maier. Deriving and Managing Data Products in an Environmental Observation and Forecasting System. In *Conf. on Innovative Data Systems Research (CIDR)*, 2005.

[12] C. Brooks, E. A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng, editors. *Heterogeneous Concurrent Modeling and Design in Java (Volumes 1-3)*. University of California, Berkeley, 2005. Technical Memorandum UCB/ERL M05/21, M05/22, M05/23.

[13] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, and I. Wang. Programming Scientific and Distributed Workflow with Triana Services. *Concurrency and Computation: Practice and Experience, Special Issue on Scientific Workflows*, 2005.

[14] The Condor High Throughput Computing System. `http://www.cs.wisc.edu/condor/`.

[15] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny. Pegasus: Mapping Scientific Workflows onto the Grid. In *European Across Grids Conference*, pages 11–20, 2004.

[16] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming Heterogeneity – the Ptolemy Approach. In *Proceedings of the IEEE*, volume 91(1), January 2003.

[17] M. Jones. SEEK EcoGrid: Integrating data and computational resources for ecology. In *DataBits: An electronic newletter for Information Managers*, Spring 2002.

[18] G. Kahn and D. B. MacQueen. Coroutines and Networks of Parallel Processes. In B. Gilchrist, editor, *Proc. of the IFIP Congress 77*, pages 993–998, 1977.

[19] Kepler: An Extensible Scientific Workflow System. `http://kepler-project.org`.

[20] E. A. Lee and T. Parks. Dataflow Process Networks. *Proceedings of the IEEE*, 83(5):773–799, May 1995.

[21] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience*, 18(10):1039–1065, August 2006.

[22] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows*, 2005.

[23] T. M. McPhillips and S. Bowers. An Approach for Pipelining Nested Collections in Scientific Workflows. *SIGMOD Record*, 34(3):12–17, 2005.

[24] T. M. McPhillips, S. Bowers, and B. Ludäscher. Collection-oriented scientific workflows for integrating and analyzing biological data. In *Proceedings of the International Workshop on Data Integration in the Life Sciences (DILS)*, volume 4075 of *LNCS*, pages 248–263, 2006.

[25] NSF ITR: A Research Project to Create Cyberinfrastructure for the Geosciences (GEON), Award #0225673, $\approx$ \$11M (collaborative), $\approx$ \$5.6M, 10/1/02–9/30/07, PI: C. Baru, co-PIs: M. Bailey, B. Ludäscher, P. Papadopoulos.

[26] NSF ITR: Enabling the Science Environment for Ecological Knowledge (SEEK), ≈ $12.2M (collaborative), 10/1/02–9/30/07, PI: W. Michener (#0225665, UNM), co-PIs: M. Jones, M. Schildhauer (# 0225676, NCEAS/UCSB), B. Ludäscher, A. Rajasekar (#0225674, SDSC/UCSD), J. Beach (#0225635, U Kansas).

[27] NSF ITR: Real-Time Data Aware System for Earth, Oceanographic, and Environmental Applications, Award #0121726, ≈ $2.3M, 9/15/03–8/31/06, PI: J. Orcutt (SIO), co-PIs: F. Vernon (SIO), A. Rajasekar, B. Ludäscher (SDSC); K. Lindquist.

[28] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. Pocock, A. Wipat, and P. Li. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics Journal*, 20(17), 2004.

[29] OMG. Life Sciences Identifiers Specification (dtc/04-05-01). `http://www.omg.org/cgi-bin/doc?dtc/04-05-01`, 2007.

[30] SciTegic. http://www.scitegic.com/.

[31] L. Smarr. Support letter for Kepler CORE. attached to this proposal, 2007.

[32] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the Condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[33] D. Weinstein, S. Parker, J. Simpson, K. Zimmerman, and G. Jones. *Visualization Handbook*, chapter Visualization in the SCIRun Problem Solving Environment. Elsevier, 2005.