Enabling Scientific Workflow Reuse through Structured Composition of Dataflow and Control-Flow

Shawn Bowers¹, Bertram Ludäscher¹,²
Anne H.H. Ngu³, Terence Critchlow⁴

Outline

- Control-Flow in Scientific Workflows and Kepler
  - Workflow Frames and Templates
  - Managing Control-Flow using Frames and Templates
  - Conclusion
Scientific Workflows

A model of the way a scientist works with their data and tools
- Mentally coordinate data export, import, analysis via software systems

Emphasize dataflow (≠ business workflows which emphasize ctrl-flow)

Goals:
- Automation
- Component reuse
- Design & documentation

... make data analysis and management tasks easier for the scientist!

Types of Scientific Workflows

- **Modeling & Design**: Capture or reverse-engineer processes and information flows at all levels

- **Knowledge Discovery**: Automate repetitive data access, retrieval, custom analysis (e.g. Blast), generic steps (PCA, cluster analysis, ..)
  - Ex: PIW, Motif analysis, NDDP, ...

- **“Plumbing”**: Stage files, submit batch jobs, monitor progress, move files off XT3 to analysis and viz cluster, archive, steer computation, ...
  - Ex: Fusion simulation, Astrophysics (supernova simulation)

- **(Real-time) Analysis Pipelines**: processing of environmental and earth science data from sensor networks
Why not just use a Python script?

- Users who can define, reuse, modify, specialize workflows may not be able to do the same for Python scripts

- Other **advantages** to scientific workflows:
  - Modular **reuse** and application interoperability
  - Debugging and monitoring workflow execution
  - Automated data management (e.g., **provenance**)
  - Validation (e.g., structural and **semantic** typing)

  ... From integrated modeling to execution, optimization, archival, ...
The Kepler Scientific Workflow System
- Extends Ptolemy II (Berkeley), developed by a EECS community (design and simulation of complex systems)
- Open-source, Java
- Computation Models, Nested WFs, Loops
- Graphical Workflow Interface
- Workflow Execution
- Extensible Architecture
- Component Libraries
- Metadata, Discovery, Archival

The Kepler “Vision”
- End-to-end scientific workflow design and execution environment
- Data- and compute-intensive workflows
- Comprehensive component libraries for a wide range of scientific domains
- Enable collaboration, sharing across disciplines (“synergy”)

[ Digression …

• … for the benefits of this audience
• … a 3 min detour to/Tour de Force of Kepler

• For a 45 min crash-course/introduction see the EDBT’06 tutorial on Scientific Data Management …
Promoter Identification Workflow (PIW)

... or from a napkin drawing...

Step 1: MicroArray Analysis
Step 2: Clusfavor Analysis
Step 3: GenBank sequence retrieval
Step 4: NCBI BLAST search
Step 5: Transfac search
Step 6: Promoter Identification
Step 7: Promoter Model generator
Step 8: NCBIBLAST search

Microarray data → Gene ID → GenBank sequence retrieval → NCBI BLAST search → Consensus sequence → promoter data → Transcription factor binding → new candidate target genes

Source: Matt Coleman (LLNL)

... to an executable workflow (here: in KEPLER)
... to a plumbing workflow (Job Mgmt w/ NI MROD)

Authors: Wilfried Anornita, Kim Baltrick, Yohann Donetsk, Wilke Selby @ University of Zurich
        Gary Altman, Adam Bambaum. Yang Zhao @ San Diego Supercomputer Center

Authors: Bowers, Ludäscher. Ngu, Critchlow

ROADEX
time reveals

Streaming demp images from SIO nest accessed via "bohemia splot.org/banner"
A Simple Scientific Workflow

Example scientific workflow run, executed as a Dataflow Process Network

Driving the point home...

- Dataflow-oriented scientific workflows have features of
  - ... stream-processing
  - ... data-, task-, and pipeline-parallelism
  - ... signal processing systems
  - ... visual PSEs: AVS/Express, IBM DataExplorer, OpenDX, LabView,
  ...

...
GEON Dataset Generation & Registration
(and co-development in KEPLER)

Matt et al. (SEEK)

Efrat (GEON)

Ilkay (SDM)

Yang (Ptolemy)

Xiaowen (SDM)

Edward et al. (Ptolemy)

SQL database access (JDBC)

Web Services ➔ Actors (WS Harvester)

% Makefile
$> ant run

“Minute-made” (MM) WS-based application integration
• Similarly: MM workflow design & sharing w/o implemented components
Some KEPLER Actors
(out of 160+ ... and counting... last week: simple Condor support)

Flow-Based Programming & Design for SWF

• Just doing visual-programming by itself does **NOT lead to modular, re-usable, maintainable workflows**!

• To fully exploit the dataflow paradigm ...
  ... **Think Dataflow!**

→ **Flow-based Programming**
  → ... combined w/ Functional, Collection-oriented Programming

• ... similar to assembly-line metaphor
Towards Flow-based Design Patterns

- Generality vs specialization of actors
  - also loosely coupled vs tightly coupled
- Data transformation pipelines
  - alternate compute and data transformation steps
- Stage-execute-fetch pattern (Grid/HPC/HTC-WFs)
- Loops, higher-order functions (map, foldr, ...)
  - cf. Taverna’s automatic loop insertion based on data types

```plaintext
connect  JDBC/SRB connection tokens, proxies, certificates (from 1-time passwords?)

A |  B |  C

map  \[ f(x_1), \ldots, f(x_n) \]

producer  [x_1, x_2, \ldots, x_n]

methods functions

F-map  \[ f_1, f_2, \ldots f_n \]

producer
```

**A Bioinformatics Workflow**

- Generality vs specialization of actors
- Data transformation pipelines
- Loop-execute-fetch pattern (Grid/HPC/HTC-WFs)
- Loops, higher-order functions (map, foldr, ...)
- cf. Taverna’s automatic loop insertion based on data types

```
map  \[ f(x_1), \ldots, f(x_n) \]

producer  [x_1, x_2, \ldots, x_n]

methods functions

F-map  \[ f_1, f_2, \ldots f_n \]

producer
```
"And that’s why our scientific workflows are much easier to understand and maintain!"

Complexity in Scientific Workflow Engineering

- While many systems (including Kepler) support execution ... little support exists for **scientific workflow engineering**
  - Formal (abstract) models for scientific workflows
  - End-to-end workflow development, e.g., methods, frameworks, management
  - Mechanisms for discovery, **reuse**, and adaptation of existing data, workflows, and actors
### Complexity in Scientific Workflow Engineering

The use of “control-flow” primitives
- Managing complex data structures (select/filter/transform)
- Provenance, logging, data management
- **Fault-tolerance** and exception handling

![Custom actors, hand-crafted control flow limited to sequential execution (SSDBM’03)](image1)

![Fault-tolerance control-flow “wired-in”, e.g. via Boolean switches, complex branching and looping](image2)

### Modeling Control-Flow Constructs in Dataflow

- **Dataflow** in Kepler
  - Based on dataflow process networks (Kahn et al, Lee et al)
  - Supports pipeline parallelism (streaming data)
  - Natural paradigm for data-driven workflows
  - Efficient analysis and scheduling
  - **Intuitive model** for workflow designers

- **Control-Flow** in Kepler
  - Branching via if-then-else and switch-case statements
  - Iteration with multiple entry and exit points
  - Low-level actors for manipulating structure (e.g., record-to-array)

- Problems modeling Control-Flow directly using Dataflow
  - Overly complicated workflows; hard to understand (low-level programming)
  - Maintenance, debugging, extending
  - limited reusability; complex re-configuration

![Dataflow&Ctrl-flow Marriage](image3)
**Approach**

- **Design Abstractions**
  - **Workflow Frames:**
    - Abstract actor placeholders, denoting set of possible implementations
  - **Workflow Templates:**
    - A (reusable) workflow which includes frames (for easy “plug-in”)

- **Transducer Templates**
  Based on “modal models” from Ptolemy:
  - Special templates that embed finite state machines
  - States can be frames/templates
  - More convenient for many types of control-flow

- **Applying Frames and Templates**
  - Three-level pattern for generic control-flow components
  - Examples: Generic Data Transfer (GDT) and Remote Execution (GX)

**Outline**

- Control-Flow in Scientific Workflows and Kepler
  - Workflow Frames and Templates
  - Managing Control-Flow using Frames and Templates
  - Conclusion
Actor-Oriented Modeling

Actors
- single component or task
- well-defined interface (signature)
- given input data, produces output data

Ports
- each actor has a set of typed input and output ports
- they make up the actor’s signature
- produce/consume data (a.k.a. tokens)
- parameters are special (more “static”) ports
**Actor-Oriented Modeling**

**Dataflow Connections**
- actor “communication” channels
- directed (hyper) edges
- connect output ports with input ports
- merge step + distribute step

**Sub-Workflows** (aka **Composite Actors**)
- composite actors “wrap” sub-workflows
- like actors, have signatures (i/o ports of sub-workflow)
  ➔ hierarchical workflows (arbitrary nesting levels)
**Actor-Oriented Modeling**

**Directors**
- define the **execution semantics** of workflow graphs
- schedule and execute workflow graphs
- sub-workflows may be governed by different directors
- Examples: Synchronous Data-Flow (SDF), Process Networks (PN), Discrete Event (DE), Finite State Machine (FSM)

**Workflow Frames**

- **Actors** are **concrete**
  - Correspond to particular implementations

- **Frames** are **abstract**
  - Denote sets of similar actor implementations
  - making an abstract signature/API a **“first-class citizen”** (a named entity)
  - a **placeholder** for a component to be “plugged in” (akin to picture frames)
  - input, output, and parameter ports
  - an **embedding** of C in F is a set of pairs “wiring” ports of C to ports of F
  - embedded components may introduce new ports, and may ignore some existing ports of F
Advantages of Frames

- **Workflow Specification**
  - For workflow designers, frames are placeholders for components that will be instantiated and specialized later.
  - High-level, conceptual definitions of workflows.
  - Useful for when multiple methods exist (algorithms, protocols, etc.).

- **Actor Abstraction**
  - For actor (library) developers, frames can be used as abstractions for a family of components with similar function.
  - Can represent a “unified” signature (ports) for similar actors.

Workflow Templates

- Just as frames abstract actors, templates abstract workflows.
  - Partially specifies the behavior of a workflow.
  - Consist of a workflow graph, where some of the components are frames or templates.
  - Input, output, and parameter ports.

- **Transducer Templates**: Behavior modeled via finite state transducers.
  - States can be frames (or templates).
3-Layered Transducer Template Design Pattern

- In general, templates and frames can be arbitrarily nested
- A specific pattern we've found useful in practice:

  - **Level 1**: A frame representing a **particular task** (e.g., data transfer), but encapsulating a set of alternative transducer templates
  - **Level 2**: A transducer-template embedding, implementing a particular control-flow behavior
  - **Level 3**: Embeddings of transducer states, denoting particular task implementations (for the Level 1 abstract function)

Using the Pattern

A Workflow Designer first selects a **generic component** G

Then selects a **behavior** from the available templates of G

Then selects **task implementations** for the template

• The configuration is entirely performed through the generic component ...
Outline

• Control-Flow in Scientific Workflows and Kepler

• Workflow Frames and Templates
  ➢ Managing Control-Flow using Frames and Templates

• Conclusion

Data Transfer and Fault-Tolerance

• Data transfer is a common task in scientific workflows
  - The transport protocol and dynamic behavior used to operate the protocol is often **hardwired** into the workflow
  - The “Retry” composite actor for fault-tolerant data transfer: contains complex feedback loops; Boolean switches; and hand-coded actors (ConditionalLoop)
The Generic Data Transfer Component

The **GDT actor** implements the transducer template pattern:

- The GDT frame specifies a basic file-transfer I/O signature
- GDT encapsulates two transducer templates (**simple retry** and **failover**)
- The **exec** (and **failover**) states are modeled as frames
- The **SCP** and **SRB Put implementations** are shown
- The GDT configuration is performed via GDT’s configuration menu (in Kepler)
  - **ModalModel actor** (Ptolemy) is used to implement transducers in Kepler

The Generic Remote Execution Component

The **GX actor** also implements the transducer template pattern:

- The GX interface is slightly simpler than GDT
- Like in GDT, users select desired template, then corresponding task implementations
- Directly reused the GDT templates
- The **SSH2** and **Globus implementations** are shown
- Both GX and GDT can be easily reconfigured, e.g., by selecting different task implementations or template behaviors
Outline

• Control-Flow in Scientific Workflows and Kepler

• Workflow Frames and Templates

• Managing Control-Flow using Frames and Templates

➢ Conclusion

Conclusions and Future Work

• While dataflow has many advantages, control-flow modeling can be cumbersome ...

... new methods are needed for modeling and designing dataflow-based workflows requiring complex control-flow

Contributions
  - Workflow frames and templates as new design entities
  - Transducer templates for specifying control-flow
  - 3-layer pattern for modeling generic components, with configurable control-flow and underlying task implementation
  - An initial Kepler implementation, with data transfer and remote execution examples

Future work
  - Extend current prototype in Kepler
  - Promote frames and templates to first-class constructs in Kepler
  - Extend frames to support structural mappings, semantic types, etc.
  - Populate Kepler with useful templates and frames
  - Look at ways to combine transducer templates (behaviors)
Q & A … and Acknowledgements

- NSF/ITR Science Environment for Ecological Knowledge (SEEK)
- NSF/ITR Geosciences Network (GEON)
- DOE/SciDAC Scientific Data Management Center (SDM)
- U.S. Dept. of Energy, LLNL