Distributed Scientific Workflows: Techniques, Tools and Applications

Omer F. Rana
School of Computer Science and Welsh eScience Centre
Cardiff University, UK
o.f.rana@cs.cardiff.ac.uk
Thanks to:

- Bertram Ludaescher at UC Davis
- Cecilia Gomes at UNL-Lisbon
- John Domingue at Open University
- Steve McGough, John Darlington at Imperial College
- Luc Moreau and Terry Payne, University of Southampton
- Zhiming Zhao, University of Amsterdam

Some material contained in this tutorial has been obtained from the individuals mentioned above.
Overview

• Introduction + Examples of Scientific Workflows
• Constructing and Managing Workflow
• Application Example: Distributed Data Mining using FAEHIM
• Adaptive Workflows
• Workflow-related research themes
Time Division

13:00 - 13:50  Workflow examples, types, general issues, scientific vs. business workflows

13:50 - 14:00  Break

14:00 - 14:50  Workflow Optimisation, Dynamic Adaptation, Semantics, Provenance

14:50 - 15:00  Triana Demonstration

15:00 - 15:10  Discussion + Research Directions
Workflow

• '70s: Skip Ellis And Gary Nutt (OfficeTalk)
  - Xerox Parc “Office Automation Systems”
  - “to reduce the complexity of the user's interface to the [office information] system, control the flow of information, and enhance the overall efficiency of the office.” (Ellis, Nutt 1980)

  - Often seen as a technique to automate existing “processes”
  - Very popular in the business world

• Over 20 years gap:
  - Availability of Computer Networks

Adapted From:
Aleksander Slominski

Historical Perspective

- ’65-’75 Decompose Applications
  - Data And Code Separated
- ’75-’85 Database Management
  - DBMS Used To Share Data
- ’85-’95 User Interface Management
  - User Interface Separated
- ’95-’08 Workflow Management
  - Isolate Business Process
  - Emerging standards - such as those based on the Service Oriented Architecture
  - Use of Service Mashups

“Workflow Management” Aalst, van Hee
Workflow

“The automation of a business process, in whole or part, where documents, information or tasks are passed from one participant to another to be processed, according to a set of procedural rules.”

**Workflow Management Coalition (WfMC)**
WFMS And WF Engine

- **Workflow Management System (WFMS)**
  - “A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications.”

- **Workflow Engine**
  - “A software service or ‘engine’ that provides the run time execution environment for a process instance.”

A representation that shows
- precedence relationship (links) between activities

May be
- Acyclic (no loops) or cyclic
- Contain annotations associate with activities or links

From: Aleksander Slominski
Workflow (+ Enactment)

From: Aleksander Slominski

Application Services Layer

Infrastructure Services

Peer Creation & resolution Services

Discovery service

Information Routing

Information/ Naming Services

Event/Mesg Service

User Help Services

Monitoring Service

(co-)scheduling Service

Security Service

Accounting Service

Workflow Instance

Workflow Engine

Orchestration Service

Launch, configure And control

User Portals/ Science Portals

Resource layer

1000s of PCs -> massive supercomputers and data sources

Network
Scientific Workflows

• What makes it different (how it is applied)?
  - Support for large data flows
  - Need to do parameterized execution of large number of jobs
  - Need to monitor and control workflow execution including ad-hoc changes
  - Need to execute in a dynamic environment where resources are not known a priori and may need to adapt to changes
  - Hierarchical execution with sub-workflows created and destroyed when necessary
• Science Domain specific requirements.

• Triana
• Taverna/SCUFL
• GridAnt
• Condor DAG
• CoG DAG
• SWFL
• BioOpera/JOpera
• BEPL4WS
• OASIS WSBPEL
• YAWL
• GSFL
• Askalon
• OMII-BPEL, etc

Origin (?): Problem Solving Environments
(MatLab, Mathematica, SciRun, NetSolve, Ninf, Nimrod etc)

http://www.nesc.ac.uk/action/esi/contribution.cfm?Title=303
http://www.extreme.indiana.edu/swf-survey/

Problems with “Predictability”

Workflow World
A chemistry lab is a hostile environment without much room to maneuver

what can be captured automatically with sensors?
what must rely on manual annotation?

From: Jeremy Frey
very precise scales - but not connected to any recording device

Competition for space

From: Jeremy Frey
From: Jeremy Frey

critical data entry
By Making Tea!

Getting not just the what and how, but the why
Getting not just the what and how, but the *why*

---

### COSHH ASSESSMENT FORM

<table>
<thead>
<tr>
<th>SUBSTANCE NAME</th>
<th>PHYSICAL FORM</th>
<th>QUANTITY</th>
<th>NATURE OF HAZARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Water</td>
<td>liquid</td>
<td>1000ml</td>
<td>None</td>
</tr>
<tr>
<td>Dextrose</td>
<td>solid</td>
<td>&lt; 20 g</td>
<td>Possible irritation to eyes and skin</td>
</tr>
<tr>
<td>Caffeine</td>
<td>solid (tea)</td>
<td>&lt; 1 g</td>
<td>Harmful if swallowed, induce vomiting.</td>
</tr>
<tr>
<td>Milk</td>
<td>liquid</td>
<td>&lt; 100 ml</td>
<td>No particular hazards</td>
</tr>
</tbody>
</table>

**NATURE OF PROCESS**

*Liquid extraction of caffeine, followed by combination with dextrose to produce a sweet drink.*

Is there a less hazardous substance? **No**

If so, why not use it?

**CONTROL MEASURES REQUIRED**

*No specific measure required.*
Promoter Identification Workflow

**Step 1**
MicroArray Analysis

**Step 2**
Clusfavor Analysis

**Step 3**
Gene ID → GenBank sequence retrieval → cDNA sequence

**Step 4**
NCBI BLAST search

**Step 5**
genomic sequence

**Step 6**
Transcription factor binding

**Step 7**
promoter data → Promoter Identification

**Step 8**
NCBI BLAST search → Consensus sequence → Promoter Model generator → new candidate target genes

Source: Matt Coleman (LLNL)
Montage (http://montage.ipac.caltech.edu/)

- Deliver science-grade custom mosaics on demand
  - Produce mosaics from a wide range of data sources (possibly in different spectra)
  - User-specified parameters of projection, coordinates, size, rotation and spatial sampling.

The Large Magellanic Cloud (LMC) is a nearby satellite galaxy of our own galaxy, the Milky Way. At a distance of slightly less than 50 kiloparsecs (≈160,000 light-years), the LMC is the third closest galaxy to the Milky Way. It has a mass equivalent to approximately 10 billion times the mass of our Sun (1010 solar masses), making it roughly 1/10 as massive as the Milky Way. The LMC is the Fourth largest galaxy in the Local Group, the first, second and third largest places being taken by Andromeda Galaxy (M31), our own Milky Way Galaxy, and the Triangulum Galaxy (M33).
SAGE: Spitzer Survey of the Large Magellanic Cloud

<table>
<thead>
<tr>
<th>Instrument</th>
<th>Bands (μm)</th>
<th>Field-of-View (arcmin)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRAC</td>
<td>3.5, 4.5, 5.8, 8.0</td>
<td>5.2 x 5.2</td>
</tr>
<tr>
<td>MIPS</td>
<td>24</td>
<td>5.4 x 5.4</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>5.25 x 2.6</td>
</tr>
<tr>
<td></td>
<td>160</td>
<td>0.5 x 0.5</td>
</tr>
</tbody>
</table>

Images courtesy Margaret Meixner (PI)

Two epochs:
Jul/Aug 05 & Oct/Nov 05

From: G. Bruce Berriman
Montage Workflow (from Ewa Deelman)

- Transfer the template header
- Transfer the image file
- Re-projection of images.
- Calculating the difference
- Fit to a common plane
- Background modeling
- Background correction
- Adding the images to get the final mosaic
- Register the mosaic in RLS
Montage workflow

~1200 nodes
Submit scientific name; retrieve accepted name & synonyms for species

Model of climatic conditions where species is currently found
OpenModeller is a fundamental niche modelling library, providing a uniform method for modelling distribution patterns using a variety of modelling algorithms.

Welcome to the openModeller project home page!

The openModeller project aims to provide a flexible, user-friendly, cross-platform environment where the entire process of conducting a fundamental niche modeling experiment can be carried out. The software includes facilities for reading species occurrence and environmental data, selection of environmental layers on which the model should be based, creating a fundamental niche model and projecting the model into an environmental scenario. A number of fundamental niche modeling algorithms are provided as plug-ins, including GARP, Climate Space Model, Bioclimatic Envelopes, and others. Additional algorithms are planned for the future. The submission of alternative algorithms are always welcome.

The project is currently being developed by the Centro de Referência em Informação Ambiental (CRIA), Escola Politécnica da USP (Poli), and Instituto Nacional de Pescuissas Espaciais (INPE) as an open-source initiative. It is funded by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), the Incofish project, and by individuals who have generously contributed their time. Previous collaborators include the BDWorld project, the University of Kiel (KU), and other individual participants.
BioDiversity Questions

• How should conservation efforts be concentrated?
  - (example of Biodiversity Richness & Conservation Evaluation)

• Where might a species be expected to occur, under present or predicted climatic conditions?
  - (example of Bioclimatic & Ecological Niche Modelling)

• How can geographical information assist in inferring possible evolutionary pathways?
  - (example of Phylogenetic Analysis & Palaeoclimatic Modelling)
Resource used in these biodiversity studies

- **Data sources:**
  - Catalogue of Life (names of species: Species 2000, GBIF)
  - Biodiversity data
    - Descriptive data
    - Distribution of specimens and observations
  - Geographical data
    - Boundaries of geographical & political units
    - Climate surfaces
  - Genetic sequences

- **Analytic tools:**
  - Biodiversity richness assessment - various metrics
  - Bioclimatic modelling - bioclimatic ‘envelope’ generation
  - Phylogenetic analysis (generation of phylogenetic trees)
Point data from various herbaria
From: Richard White

BDWorld Architecture
GARP prediction of climatic suitability
Triana For BdWorld

Input
- PopupStringInput
- PopupTaxonGen
- SpeciesListGen
- StringArrayGen
- StringInput
- TaxonGen
- UniGen

Output
- DataCollectionDisplay
- HtmlViewer
- MapViewer
- PopupTaxonDisplay
- StringOutput
- TaxonDisplay
- VarDocViewer

Processing
- EnrichQuery
- EnrichStockListing
- EnrichSpeciesModeling
- CommandTool
- Locates
- GetLocates
- OpenModeling
- RunOpenModeling
- Sipas
- GetTaxon
- SpeciesSearch
- SpeciesTaxonSearch
- ThemeLessMapping
Workflows in Earthquake Engineering

Click on an earthquake on the map above to zoom in.

From: Philip Maechling
Observed Areas of Strong Ground Motion

From: Philip Maechling
Simulations Supplement Observed Data

From: Philip Maechling
SCEC/CME Scientific Workflow Construction

Define Scenario Earthquake

ERF Definition

IMR Definition

Gridded Region Definition

Calculate Hazard Curves

9000 Hazard Curve files (9000 x 0.5 Mb = 4.5Gb)

Extract IMR Value

Probability of Exceedence and IMR Definition

Plot Hazard Map

Lat/Long/Amp (xyz file) with 3000 datapoints (100Kb)

GMT Map Configuration Parameters

ERF: Earthquake Rupture Forecast
IMR: Earthquake Rupture Slip time function generator

Intensity Measure Relationship
OpenSHA: Open Seismic Hazard Analysis
http://www.opensha.org/index.html

From: Philip Maechling
In the first year, a Pathway-1 team led by Ned Field erected a new object-oriented architecture for seismic hazard analysis, dubbed "OpenSHA" ([http://www.opensha.org/](http://www.opensha.org/)). This Java-based code implements a number of SHA conceptual objects, such as earthquake forecast models (EFM), intensity measure relationships (IMR), and intensity measure types (IMT). The team has thus far incorporated seven different IMR’s that are applicable to Southern California and has developed an analysis tool that lets users explore the implications of the IMR’s via a Web-enabled graphical user interface. The API between the IMR’s and the analysis tool is very general and flexible, so that any new models can be plugged into the framework without having to change existing code. Along with the codes that calculate seismic hazard analysis curves, we have developed Web-based analysis tools that allow the user to explore the implications of combining various EFM’s with a number of possible IMR’s. OpenSHA will thus provide the platform for integrating the research done by the SCEC working group on Regional Earthquake Likelihood Models (RELM) ([http://www.relm.org/](http://www.relm.org/)). An overview of the OpenSHA architecture will be presented in a paper by N. Field, T. H. Jordan, and C. A. Cornell to be published soon in _SRL_.

The OpenSHA framework has provided an interesting and challenging initial application of our KR&R technology. A Pathway-1 team comprising SCEC scientists and AI researchers from ISI has developed an initial knowledge base for SHA objects such as EFM’s and IMR’s using a powerful KR&R inference engine named PowerLoom ([http://www.isi.edu/isd/LOOM/PowerLoom/](http://www.isi.edu/isd/LOOM/PowerLoom/)). A Web-based tool called DOCKER (Distributed Operations of Code with Knowledge-based descriptions for Earthquake Research) was developed to allow users to define and perform Pathway-1 calculations by accessing the SHA knowledge base. As the user sets up a computational pathway by specifying the hazard-curve variables, DOCKER checks the user’s selections for consistency by querying the SHA knowledge base and warns the user of inconsistencies. Moving the SHA system into the calculation of hazard maps will significantly increase the execution time, so this type of consistency checking should prove useful.
Flight Maneuver Simulation

- Project SikMa
  - Interactive simulation of a freely flying, fully configured, elastic warplane
- SikMa partners provided end-user requirements
Scientific vs. Business Workflow

- **Reproducability** of results at the core of the scientific method
  - Create, manage and capture dynamic workflows
- **Scientific Exploration and User-Steering**
  - Flexibility of design + **exploration** capabilities
  - Need to represent workflow variants (different workflow configurations and settings) - support complex scientific exploratory processes
  - Support for “user-steered” workflows
  - vs. more prescriptive use in business computing
- **Emphasis on Data**
  - **Heterogeneous** with different access patterns
  - Domain specific formats (textual, semi-structured, visual + varying degrees of annotations)
Scientific vs. Business Workflow ... 2

- Common Characteristics
  - Repetitive cycle of data analysis and data migration
  - Parameter sweep or “range search” operations (can lead to creation of multiple jobs)
  - Dependencies between partial results generated through a workflow
  - Data aggregation across different repositories – often in different formats
  - Composition of complex data capture + simulation engines in a single (often linear) pipeline
Pathway to Pubmed

Title: Pathway to Pubmed
Type: Taverna 1

License
All versions of this Workflow are licensed under the Creative Commons Attribution-Share Alike 3.0 License.

Credits (1)
(People/Groups)
- Paul Fisher

Attributions (0)
(Workflows Files) None

Tags (10)

Original Uploader
Omer Frana

Version 2 (latest) (of 2)
View version: 2 (latest)
Version created on: 09/03/08 @ 15:03:11 by: Paul Fisher
Revision comments
Last edited on: 09/07/08 @ 12:13:06 by: Paul Fisher
Workflow Lifecycle

- **Design**
  - Typical workflow is graph oriented (ease of use)
  - Language: how expressive is workflow
  - GUI: Visual Service Composition Environment
- **Deployment**
  - Workflow Description is sent to Workflow Engine
  - Possibly validated and compiled
- **Execution**
  - Workflow Engine enacts Workflow Description
- **Monitoring**
  - Events reflecting from workflow and services execution
- **Refinement**
Workflow refinement and execution (Ewa Deelman)

Diagram:
- User's Request
- Levels of abstraction
- Application knowledge
- Logical tasks
- Tasks bound to resources and sent for execution
- Workflow refinement
- Policy info
- Workflow repair
- Task matchmaker
- Not yet executed
- Executed
- Time
Workflow Representation

- Abstract ("design time") workflow
  - Task graph encoding data flow or control flow dependencies
  - "Scientific" reproducability
- "Concrete" (run time) workflow
  - Service bindings to an abstract workflow graph
  - "Engineering" reproducability

- Sharing of graph structures, rather than just services
  - Limited case: "composite services"
Workflow System Architecture

Composition and Modelling/Analysis

Enactment/Mapping

Execution
Workflow System Architecture

Composition and Modelling/Analysis

Enactment/Mapping

Execution

Information Service
Data Migration
User Interaction
User Services
Planning Engine
Checkpointing
Scheduling
Workflow Taxonomy

Workflow System

- Workflow design and specification
  - Structure
  - Composition
  - Model/spec
- Scheduling and enactment
- Operational attributes
- Data management
Workflow Composition

Composition

User Directed

Language-based

Markup

Logic

Functional

Process Calculi

Graph-based

DAG

UML

Petri Net

Automated

User defined

Process Calculi

Automated

Planner

Templates

Design Patterns

Sub-workflows

Factory

Textual: BPEL, SCUFL (Taverna), DAGMan, DAX

Graphical: Taverna, Triana, Kepler, SciRun, VisTrails

Planner/Semantics: Wings/Pegasus, IXI, Taverna/FETA

UML: Askalon (UML Activity Dia.)
Petri nets

- Directed cyclic graph
- 2 types of nodes: places and transitions
- Arcs: place-transition, transition-place
- Tokens: move on the graph, enable/fire transitions

Reference nets
- Tokens can be nets
- Nested structures: Parent and child nets
- Dynamic creation of tokens
- Synchronous channels
Petri Nets
(from van der Aalst und Kumar, 2000)

Task

Sequence

Choice
Petri Nets

Condition
Petri Nets

Parallel execution with synchronization

[Diagram of Petri Net with AND-split, re_1, re_2, ..., re_n, AND-join]
Petri Nets
Parallel execution without synchronization
Petri Nets

Wait any with time out
The meta model of each language vary quite a bit from one specification to another. BPML, XLANG and WSFL are all relying on the concept of Web Services. They also clearly define a data flow (as XML documents), a control flow (block structured or transition based), a message flow (web services) and transaction flow. However, they do not spend much time on specifying how users may interact with a BPMS. The WfMC has mostly focused on that problem in the past (see "user friendly" column below). On the other hand, the WfMC specification does not support a real message flow and only a very limited data flow (process variables). The following table summarizes the differences between each data model. (Please note that I do not have the UML 2.0 data points yet)

Note that Stefan Haberl provides another view of this matrix which I like better.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Control flow</th>
<th>Data flow</th>
<th>Message flow</th>
<th>Transaction</th>
<th>&quot;EAI friendly&quot;</th>
<th>&quot;B2B friendly&quot;</th>
<th>&quot;User friendly&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPML 1.0</td>
<td>Block structured</td>
<td>XML</td>
<td>Web services &amp; Global model</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>BPML 0.4</td>
<td>Block structured</td>
<td>XML</td>
<td>Web services</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>XLANG</td>
<td>Block structured</td>
<td>XML</td>
<td>Web services &amp; Contracts</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>WSFL</td>
<td>Transitions</td>
<td>XML</td>
<td>Web services &amp; Global model</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>BPDL 4WS</td>
<td>Block structured</td>
<td>XML</td>
<td>Web services</td>
<td>yes</td>
<td>yes</td>
<td>kind of</td>
<td>no</td>
</tr>
<tr>
<td>EDOC</td>
<td>Event/Notification</td>
<td>Entities</td>
<td>Events</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>XPDL</td>
<td>Transitions</td>
<td>Process variables</td>
<td>Nested and chained processes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>UML 2.0</td>
<td>Transitions</td>
<td>Transitions and data buffers</td>
<td>collaboration model</td>
<td>no</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

This table makes it pretty clear: the foundations of a modern BPMS are XML and Web Services and I totally agree with this. However, this is far from enough.

I have added "B2B EAI and User friendly" columns to measure how serious each specification is in dealing with the 3 main axes of a BPMS. As you can see, none of the new generation of PML is taking into account user interactions, though they are essential to resolve business process exceptions. None of them is considering WSFL, BPMN, or XPDL. However, BPMN is also the only one which is taking into account the interactions of the BPMS.
Process Descriptions

[Diagram showing various process descriptions and standards, including BPEL, WS-CDL, BPMN, and others, with references to IBM, Microsoft, BEA, OASIS, etc.]

YAWL notation

- Condition
- Start condition
- End condition
- XOR-split task
- OR-split task
- AND-split task
- XOR-join task
- OR-join task
- AND-join task
- Composite task
- Multiple Instance task
Abstract vs. Executable

- abstract processes
  - public behaviour
  - define “business protocols”
  - hide things that do not affect partner
    - constrain only the message exchange
    - what the possible replies are, not why one is chosen

- executable processes
  - private behaviour
  - fully define behaviour
  - portable between compliant environments

- WS-Choreography
  - Defines abstract behaviour

From: Peter Furniss, Choreology
BPEL Concept: Basic Activities

- Do a blocking wait for a matching message to arrive / send a message in reply
- Invoke a one-way or request-response operation
- Update the values of variables or partner links with new data
- Validate XML data stored in variables
- Generate a fault from inside the business process
- Forward a fault from inside a fault handler

- Immediately terminate execution of a business process instance
- Invoke compensation on all completed child scopes in default order
- Invoke compensation on one completed child scope
- Wait for a given time period or until a certain time has passed
- No-op instruction for a business process
- Wrapper for language extensions

From: Michael Illiger and Simon Moser
**BPEL Concept: Structured Activities**

- Contained activities are executed in parallel, partially ordered through control links.
- Contained activities are performed sequentially in lexical order.
- Contained activity is repeated while a predicate holds.
- Contained activity is repeated until a predicate holds.

- **Flow**
  - Block and wait for a suitable message to arrive (or time out).

- **Sequence**
  - Contained activity is performed sequentially or in parallel, controlled by a specified counter variable.

- **While**
  - Select exactly one branch of activity from a set of choices.

- **RepeatUntil**
  - Associate contained activity with its own local variables, partner links, etc., and handlers.

*From: Michael Illiger and Simon Moser*
activities

• **structured activities** – can contain other activities
  
  `<sequence>` one after the other
  `<flow>` in parallel
  `<pick>` choose by inbound message
  `<switch>` choose by expression evaluation
  `<while>` iteration
  `<scope>` nest, with declarations and handlers, synchronize

• **communication**
  
  `<invoke>` send msg to partner; possibly receive response
  `<receive>` accept msg from partner
  `<reply>` send msg to partner as response to `<receive>`

• **other**
  
  `<assign>` manipulate variables
  `<wait>` for duration / until time
  `<terminate>` end the process
  `<compensate>` run compensation handler of inner scope
  `<throw>` exit with fault to outer scope
  `<empty>` do nothing

From: Peter Furniss, Choreology
“Legacy” Code Handling

Pre-existing codes, mostly in Fortran

- Generally domain-specific
- Hard to re-use in other applications
- They are still useful
- They are often large, complex monoliths with little structure.

- Support Re-use
- Support Remote Execution
- Support Remote Discovery
- Support Remote Data Input/Output

Re-write? - try convincing App Scientists
Wrapping Approaches

- **Wrapping executables - “As-Is” Approach**
  - No source available (or provided)
  - Maintain execution environment
- **Wrapping Source - “Source-Update” Approach**
  - Some source provided (generally I/O)
  - Executable can relinquish some control
  - Data type conversions
- **Source split Wrapping - “Unit-Mapping” Approach**
  - Split source into units -- wrap units
  - Maintain unit execution environment + overall manager
- **Application Supported Wrapping - “App-Wrap”**
  - Steering support
  - Data management support
Wrapping Approaches

- Wrapping executables - “As-Is” Approach
  - No source available (or provided)

- Provide Isolation between existing code, in its present form, and need to re-use and execute code remotely

- Enable properties of code to be specified (in terms, perhaps of its interface), to enable a discovery mechanism to utilise in, say, a particular application.

- Sustain performance, correctness of results, ownership, and availability
Automating Wrapping

- Time consuming and error prone process
- Automate the implementation of interfaces to access code
  - via a system wide data model
- Automate interactions between wrapped components
  - via a discovery service
  - Registry/Lookup service
- Can have
  - same interface, different implementation
Component Model and Extensions

Existing Code
Component Model and Extensions

Existing Code

Data Type Translation
Component Model and Extensions

<def>
<preface>
  <name alt="MD1" id="MD01"> MDComponent </name>
  <pse-type> Molecular Dynamics </pse-type>
  <component-directory>/home/scmlm1/wgen/Component</component-directory>
  <legacy-code>/home/scmlm1/md/moldyn</legacy-code>
  <ORB-Compiler>idl2java</ORB-Compiler>
  <processors>8</processors>
  <host-name>sapphire.cs.cf.ac.uk</host-name>
</preface>

<outports>
  <outportnum> 6 </outportnum>
  <outport id="1"> int </outport>
  <outport id="2"> float </outport>
  <outport id="3"> float </outport>
  <outport id="4"> float </outport>
  <outport id="5"> float </outport>
  <outport id="6"> float </outport>
  <href name="file:/home/scmlm1/wgen/Component/output.data" value="output" />
</outports>

XML Data Model
Component Model and Extensions

Adding additional control inputs

Existing Code

External Control Input (for Steering)
Component Model and Extensions

Adding “container” services

Data Manager

Existing Code

Runtime support
Component Model and Extensions

Adding an execution policy

Data Manager

Existing Code

Runtime support

Execution Rules
Control Flow vs. Data Flow

• Control Flow
  - Managed via use of specialist control constructs (conditions -- may be simple conjunction/disjunction, or more complex operators)
  - Unit/component execution managed through these control constructs
  - Types
    • Transition only
    • Switch, flow, while, etc

• Data Flow
  - Execution managed via transmission of data
Data/Control-Flow Spectrum

“clean” data(-ctl)-flow    special tokens flow    message passing, control flow

- Data (tokens) flow
  - (almost) no other side effects
  - WYSIWYG (usually)
- References flow
  - token reference type may be “http-get”, “ftp-get”
  - generic handling still possible
- Application specific tokens flow
  - e.g. current Nimrod job management in Resurgence
  - “invisible contract” between components
  - Director (Kepler) is unaware of what’s going on
- Specific messages passing protocols (e.g., CSP, MPI)
  - for systems of tightly coupled components
Dealing with Loops and Conditionals

- Often difficult to achieve - often ignored
- Conditional
  - Specified as control-blocks
  - Implemented through the use of scripts
- Loops
  - Specified as “meta-blocks” - blocks implemented over sub-workflows
  - Implemented through the use of scripts
- Must be supported in the Enactment Engine
- YAWL → defines the concept of “worklets” - sub-workflows over which control constructs can be applied
In Triana and Kepler – use of specialist “Loop” components

- Components can be explicit
- Implemented as “hidden” command
Loops ... 3

init() iteration() isExitLoop(Object[] data)
(Allows for user defined objects to specify loop exit condition)

iterations >= total iterations / 2
ConstGen1.constant < ConstGen2.constant + 2 / 3
$data0 = iterations % (10 + $var2)
$var1 = $data0 + $data1
Control-flow Patterns

- **Basic Control-flow Patterns**
capture elementary aspects of control-flow (similar to the concepts provided by the WFMC).

- **Advanced Branching and Synchronization Patterns**
describe more complex branching and synchronization scenarios.

- **Iteration Patterns**
describe various ways in which iteration may be specified.

- **Termination Patterns**
address the issue of when the execution of a workflow is considered to be finished.

- **Multiple Instances (MI) Patterns**
delineate situations with multiple threads of execution in a workflow which relate to the same activity.

- **State-based Patterns**
reflect situations which are most easily modelled in WF languages with an explicit notion of state.

- **Cancellation Patterns**
categorise the various cancellation scenarios that may be relevant for a workflow specification.

- **Trigger Patterns**
catalogue the different triggering mechanisms appearing in a process context.
Data Pattern Categories

- **Data Visibility**: The extent and manner in which data elements can be viewed and utilised by workflow components.

- **Internal Data Interaction**: Data communication between active elements within a workflow.

- **External Data Interaction**: Data communication between active elements within a workflow and the external operating environment.

- **Data Transfer**: Data element transfer across the interface of a workflow component.

- **Data Routing**: The manner in which data elements can influence the operation of the workflow.
Workflow Resource Patterns

• Focus on the manner in which work is offered to, allocated to, and managed by workflow participants

• Consider both the system and resource perspectives

• Assume the existence of a process model and related organisational model

• Take into account differing workflow paradigms:
  - richness of process model (esp. allocation directives)
  - autonomy of resources
  - alternate routing mechanisms
  - work management facilities
# The Workflow Patterns Framework

## Evaluations

<table>
<thead>
<tr>
<th>Time</th>
<th>Control-flow P:s 20</th>
<th>Resource P:s - 43</th>
<th>Data P:s - 40</th>
<th>Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jun 2005</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oct 2005</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Language Development:

- YAWL/newYAWL

- BPEL4WS
- UML AD 1.4, UML AD 2.0, BPMN
- XPDL, BPEL4WS, BPML, WSFL, XLANG, WSCI, UML AD 2.0, BPMN
### Control-Flow Perspective: Evaluation

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Control-flow</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>Termination</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Sequence</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>11 Implicit Termination</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>2 Parallel Split</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>43 Explicit Termination</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>3 Synchronisation</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>12 MI without Synchronisation</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>4 Exclusive Choice</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>Multiple Instances</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 Simple Merge</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>13 MI with a priory Design Time Knlg</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td><strong>Advanced Synchronisation</strong></td>
<td></td>
<td></td>
<td></td>
<td>14 MI with a priory Runtime Knlg</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>6 Multiple Choice</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>15 MI without a priory Runtime Knlg</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7 Str. Synchronising Merge</td>
<td>+/-</td>
<td>-</td>
<td>+</td>
<td>27 Complete MI Activity</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8 Multiple Merge</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>34 Static Partial Join for MI</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9 Discriminator</td>
<td>+/-</td>
<td>+</td>
<td>-</td>
<td>35 Cancelling Partial Join for MI</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10 Arbitrary Cycles</td>
<td></td>
<td></td>
<td></td>
<td>16 Deferred Choice</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>11 Cancellation</td>
<td></td>
<td></td>
<td></td>
<td>17 Interleaved Parallel Routing</td>
<td>+/-</td>
<td>-</td>
<td>+/-</td>
</tr>
<tr>
<td>12 Structured Loop</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>18 Milestone</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>13 Generalised AND-Join</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>20 Cancel Case</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>14 General Synchronizing Merge</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>25 Cancel Region</td>
<td>+/-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>15 Critical Section</td>
<td></td>
<td></td>
<td></td>
<td>19 Cancel Activity</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>16 Iteration</td>
<td></td>
<td></td>
<td></td>
<td>26 Cancel MI Activity</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>17 Transient Trigger</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>23 Transient Trigger</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>18 Persistent Trigger</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>24 Persistent Trigger</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Data Visibility</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>-----------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Task Data</td>
<td>+</td>
<td>+</td>
<td>+/-</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Block Data</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Scope Data</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MI Data</td>
<td>+/-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Case Data</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Folder Data</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Workflow Data</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Environment Data</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Data Interaction (Internal)</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>between Tasks</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Task to Sub-workflow Decomp.</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sub-workflow Decomp. to Task</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>to MI Task</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>from MI Task</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Case to Case</td>
<td>-</td>
<td>-</td>
<td>+/-</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Data Interaction (External)</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Task Precondition Data Exist.</td>
<td>+</td>
<td>+</td>
<td>+/-</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Task Precondition Data Value</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Task Postcondition Data Exist.</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Task Postcondition Data Value</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Event-based Task Trigger</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Data-based Task Trigger</td>
<td>+</td>
<td>-</td>
<td>+/-</td>
<td>+/-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Data-based Routing</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Data Visibility**
- Task Data
- Block Data
- Scope Data
- MI Data
- Case Data
- Folder Data
- Workflow Data
- Environment Data

**Data Interaction (Internal)**
- between Tasks
- Task to Sub-workflow Decomp.
- Sub-workflow Decomp. to Task
- to MI Task
- from MI Task
- Case to Case

**Data Interaction (External)**
- Task Precondition Data Exist.
- Task Precondition Data Value
- Task Postcondition Data Exist.
- Task Postcondition Data Value
- Event-based Task Trigger
- Data-based Task Trigger
- Data-based Routing
Resource Patterns Classes

- **Creation patterns**: design-time work allocation directives
  生成パターン: 設計時でのリソース割り当て
- **Push patterns**: workflow system proactively distributes work items
  プッシュパターン: ワークフローシステムが積極的に作業を提供
- **Pull patterns**: resources proactively identify and commit to work items
  プルパターン: リソース(人など)が積極的に作業をコミットする
- **Detour patterns**: re-routing of work items
  回り道パターン: 作業の流れを変える
- **Auto-start patterns**: automated commencement
  自動スタートパターン: 自動開始のパターン
- **Visibility patterns**: observability of workflow activities
  可視化パターン: 作業の監視性
- **Multiple resource patterns**: work allocation involving multiple participants or resources
  複数リソースパターン: 複数リソースにまたがる作業の割り当て

Click here for a FLASH animation of Delegation Pattern
### Resource perspective: Evaluation

**1 – BPMN, 2 – UML AD, 3 – Oracle BPEL PM**

*(from [Mulyar 2005])*

<table>
<thead>
<tr>
<th>Creation Patterns</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Direct Allocation</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>2 Role-Based Allocation</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>3 Deferred Allocation</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>4 Authorization</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5 Separation of Duties</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6 Case Handling</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>7 Retain Familiar</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>8 Capacity-based Allocation</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>9 History-based Allocation</td>
<td>-</td>
<td>-</td>
<td>+/-</td>
</tr>
<tr>
<td>10 Organizational Allocation</td>
<td>-</td>
<td>-</td>
<td>+/-</td>
</tr>
<tr>
<td>11 Automatic Execution</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td><strong>Push Patterns</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 Distribution by Offer-Single Resource</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>13 Distribution by Offer-Multiple Resources</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>14 Distribution by Allocation-Single Resource</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>15 Random Allocation</td>
<td>+</td>
<td>+/-</td>
<td></td>
</tr>
<tr>
<td>16 Round Robin Allocation</td>
<td>+</td>
<td>+/-</td>
<td></td>
</tr>
<tr>
<td>17 Shortest Queue</td>
<td>+</td>
<td>+/-</td>
<td></td>
</tr>
<tr>
<td>18 Early Distribution</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>19 Distribution on Enablement</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>20 Late Distribution</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td><strong>Pull Patterns</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21 Resource-Init. Allocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>22 Resource-Init. Exec. - Allocated WI</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>23 Resource-Init. Exec. - Offered WI</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pull Patterns (cont.)</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 System-Determin. Work Queue Cont.</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>25 Resource-Determin. Work Queue Cont.</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>26 Selection Autonomy</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>27 Delegation</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>28 Escalation</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>29 Deallocation</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>30 Stateful Reallocation</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>31 Stateless Reallocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>32 Suspension/Resumption</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>33 Skip</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>34 Redo</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>35 Pre-do</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>36 Commencement on Allocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>37 Commencement on Allocation</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>38 Piled Execution</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>39 Chained Execution</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>40 Config. Unallocated WI Visibility</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>41 Config. Allocated WI Visibility</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Multiple Resource Patterns</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42 Simultaneous Execution</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>43 Additional Resources</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

| Detour Patterns                                                                 |
|----------------------------------------------------------------------------------|---|---|---|
| 27 Delegation                                                                    | - | - | + |
| 28 Escalation                                                                    | - | - | + |
| 29 Delegation                                                                    | - | - | + |
| 30 Stateful Reallocation                                                         | - | - | + |
| 31 Stateless Reallocation                                                        | - | - | - |
| 32 Suspension/Resumption                                                         | - | - | + |
| 33 Skip                                                                          | - | - | + |
| 34 Redo                                                                          | - | - | - |
| 35 Pre-do                                                                        | - | - | - |
| 36 Commencement on Allocation                                                    | - | - | - |
| 37 Commencement on Allocation                                                    | - | - | - |
| 38 Piled Execution                                                               | - | - | - |
| 39 Chained Execution                                                             | - | - | - |
| 40 Config. Unallocated WI Visibility                                            | - | - | - |
| 41 Config. Allocated WI Visibility                                               | - | - | - |

| Auto-start Patterns                                                              |
|----------------------------------------------------------------------------------|---|---|---|
| 36 Commencement on Creation                                                      | + | + | - |
| 37 Commencement on Allocation                                                    | - | - | - |
| 38 Piled Execution                                                               | - | - | - |
| 39 Chained Execution                                                             | - | - | - |
| **Visibility Patterns**                                                          |
| 40 Config. Unallocated WI Visibility                                            | - | - | - |
| 41 Config. Allocated WI Visibility                                               | - | - | - |

| Multiple Resource Patterns                                                      |
|----------------------------------------------------------------------------------|---|---|---|
| 42 Simultaneous Execution                                                        | + | + | + |
| 43 Additional Resources                                                          | - | - | + |
Enactment Engines

- Employ a variety of techniques for enactment

- Integrated with a Portal – others based on a command line interface (some also provide a scripting language)

- Generally for constructing graphs – others also support execution of components within a graph

- Support for third-party services
  - Monitoring, Registry, etc

- Can take workflow as input, process this, and return another workflow (equivalent to treating workflow graphs as data)
Enactment Strategies ... I

- **Centralised Enactor**
  - Single graph coordinated through a centralised enactor
  - The enactor manages execution of components in some sequence

- **Distributed Enactors**
  - Graph divided into sub-graphs and handed to different enactors
  - Each enactor responsible for executing local graph
  - Divide graph across enactors
    - Undertaken by a user
    - Undertaken automatically using rules (more later)
Enactment Strategies ... II

• Event-based
  - Each component on completion generates an event
  - Use of publish-subscribe mechanism
  - Each component also activated through the generation of an event
  - Can have multiple event types

• Blackboard/Shared memory
  - Component/Enactor writes to a shared space
  - Monitored by components/enactor
  - Blocks on availability of particular data items in shared space
.NET Services – Windows Workflow Foundation

- Hosting of Workflow:
  - Own Host
  - “Dublin”
  - .NET workflow Services
- Hosting supported in a Microsoft Cloud (via Microsoft Azure)
- Supports multiple instances of a workflow instance (for fault tolerance) - through a multicast Service Bus
Activities in Microsoft Workflows

• Out-of-Box WF Activities
  - IfElse, Sequence, Suspend, Terminate, While
• .NET Workflow Service Activities (for Azure)
  - Delay, HTTPSend, HTTPReceive, ServiceBusSend (for Events), XPathRead, XPathUpdate (content-based routing)
• Execution supported through a .NET execution engine
  - Workflow status (terminated, suspended, running, etc)
  - Can use portal to change workflow definition
Trident & NEPTUNE (Roger Barga, Microsoft)

- Trident: Scientific workflow tools using Microsoft Workflow Foundation
- Distributed registry service for sensor + simulation/model data
- Trident enables:
  - Automate tedious data cleaning and analysis pipelines.
  - Explore and visualize data, regardless of source.
  - Compose, run and catalog experiments, save results.
  - Explore and visualize ocean & model data.
- Also, utilize collaboration facility in MSW
  - through the use of .NET Services portal
Populate Windows WF with custom activities
- Introduce gridded data structures;
- Define basic operators (data transformations);
- Implemented as custom activities;

Introduce parameterized activities
- Easier for users to design workflows
- Tool to convert custom to parameterized activities

Invoke and author workflows via web browser

Persistent workflows, checkpoints (*stop-revise-rerun*)
Yahoo! Pipes

- Exports data to RSS, XML and JSON (data aggregation)
- Mainly provides support for aggregating and manipulating RSS feeds
  - Feeds can come from Google Base, Flickr, Yahoo! Local, CSV files etc.
- Provides a variety of functions for this
- Allows
  - Translation between feeds
  - Aggregation of feeds
  - Integration with map
- Focus primarily on a data driven approach
[PERSPECTIVES] DEVELOPMENTAL BIOLOGY: Brain Vats for Blood Vessels
[PERSPECTIVES] CHEMISTRY: Interrogating Molecules
[NEWS] ASTROPARTICLE PHYSICS: Excess Particles From Space May Hint at Dark Matter
Editors' Choice
[NEWS] CRIMINOLOGY: Study Shows How Degraded Surroundings Can Degraded Behavior
[SPECIAL FEATURES] SCIENCE CAREERS: Finance's Quantum Mechanics
[POLICY FORUM] POLICY: A Force for Peace in the Middle East
[NEWS FOCUS] SCIENCE IN ROMANIA: At Home in Bucharest, for Better and for Worse
+52 more...
DoodleMap with JOpera

YahooLocal

ConvertY2D

DoodleCreatePollText

GetPollID

GetPoll

Wait

Convert

ShowGMapDoodle

Poll: hamburger
CP has created this poll.

*16001*

<table>
<thead>
<tr>
<th>Gray's Papaya</th>
<th>Gray's Papaya</th>
<th>Hilton-Millenium</th>
<th>Island Burgers &amp; Shakes</th>
<th>Le Parker Meridien</th>
<th>New York</th>
<th>Corner Bistro</th>
<th>ESPN Zone</th>
<th>Warwick New York Hotel</th>
<th>The Jekyll &amp; Hyde Club</th>
<th>Dow the Hatc</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>PA</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>
JOpera Example: Doodle Map Mashup

- Setup a Doodle with Yahoo! Local search and visualize the results of the poll on Google Maps

From: Cesare Pautasso
ActiveSheets/EnFuzion

- Extend MS-Excel to support execution of functions
  - Excel → Nimrod Cache → Nimrod-based execution
  - Use OLE extensions (VB + ActiveX DLL)
- Support for parallel evaluation of cells within a spreadsheet
  - Results of one cell may feed as input into another
- Use of custom functions (rather than built in ones in Excel) - evaluate cells in a data flow manner
  - Cells must be “functionally” independent
- Table used to maintain current state:
  - not evaluated, under evaluation, evaluated
- Custom function returns “before” evaluation completes - causing other functions to be evaluated in parallel

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>=A1+B1</td>
<td></td>
<td>=C1+D1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>=A2+C2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Trapezoidal Integration of External Function "SlowSine" using Nimrod over Globus
Enactment for Automated Composition (more later)

- Enactment engine enlists use of other components
  - Discovery Service
  - Planning Engine
- Enactment may be “goal-oriented”
  - Define requirement, rather than components
  - Conflict detection support
  - Mechanisms to chose between alternatives (constraints)

Difficult to do in practice
GridLab Implementation

http://www.trianacode.org/
**Java GAT Prototype**

- **GAP (Java Prototype)**
  - **Jxta**
  - Web Services
  - P2PS
  - OGSA
  - Jxtaserve
  - GSI Enabled
  - NS-2
  - And more...

- Features:
  - Set of generic Java interfaces
  - High level abstractions to Grid services
  - Factory design - dynamic pluggable services

- GridLab GAT (www.gridlab.org)

- Functions:
  - Advertising
  - Discovery
  - Communication
  - Generic Job Submission
  - Virtual filename data access
Plug-in Applications
- flexible: apps can use Triana in various ways, as a:
  - GUI
  - remote control GUI
  - or in full inc. GAP/GAT

Triana Architecture

Plug-in Applications
- flexible: apps can use Triana in various ways, as a:
  - GUI
  - remote control GUI
  - or in full inc. GAP/GAT
Triana Distributed Workflow

- flexible distribution: based around Triana Groups
- HPC and Pipelined distribution

Triana Controlling Service (TCS)

 workflows, e.g. BPEL4WS

Action Commands

Network

Other Engine

Triana Engine

Triana Gateway

Triana Service & Engine

Triana Service & Engine

Triana Service & Engine
Distributing Triana Taskgraphs

- **Mapping tasks or groups of tasks to resources**
- **Two stages:**
  - Taskgraph annotation, XML definition for each task or group of tasks
  - Extended to specify resources and message channels
  - Data distribution, annotated sub-sections of taskgraph passed to resources
Custom Distribution

- The workflow is cloned/split/rewired to achieve the required distribution topology

- Custom distribution units allow sub-workflows to be distributed in parallel or pipelined

- Distribution units are standard Triana tools, enabling users to create their own custom distributions
Remote Deployment

- User can distribute any task or group of tasks (sub-workflow)

- Using the GAP Interface, Triana automatically launches a remote service providing that sub-workflow.

- Input, Output and Control Pipes are connected using the current GAP binding (e.g. JXTA Pipes)
Deploying and Connecting To Remote Services

- Running services are automatically discovered via the GAP Interface, and appear in the tool tree.

- User can drag remote services onto the workspace and connect cables to them like standard tools (except the cables represent actual JXTA/P2PS pipes).
Web Service Discovery 1

- Triana allows users to query UDDI repositories

- Alternatively, users can import services directly from WSDL
Web Service Discovery 2

- Discovered/Imported Web Services are converted into Triana tools
  (service name = tool name)
  (input message parts = in nodes)
  (output message parts = out nodes)
  etc...

- Web Service tools are displayed in the user's Tool Tree (alongside local tools)
Connecting Workflows

- Web Service tools can be dropped onto the user’s workspace and connected like local tools
- A workflow can contain both local and Web Service tools
Complex Data Types

- Users can build their own interface for creating/mediating between complex types
- Alternatively, Triana can dynamically generate an interface from the WSDL2Java generated bean class
GEMSS: Maxillo-facial Surgery Simulation
GEO 600 Inspiral Search

• Background
  - Compact binary stars orbiting each other in a close orbit
    • among the most powerful sources of gravitational waves
  - As the orbital radius decreases a characteristic chirp waveform is produced - amplitude and frequency increase with time until eventually the two bodies merge together

• Computing
  - Need 10 Gigaflops to keep up with real time data (modest search..)
    • Data 8kHz in 24-bit resolution (stored in 4 bytes) -> Signal contained within 1 kHz = 2000 samples/second
    • divided into chunks of 15 minutes in duration (i.e. 900 seconds) = 8MB

• Algorithm
  - Data is transmitted to a node
  - Node initialises i.e. generates its templates (around 10000)
  - fast correlates its templates with data
Coalescing Binary Search
Coalescing Binary Scenario

Controller

Email, SMS notification

Logical File Name

GW Data Distributed Storage

GAT (GRMS, Adaptive)
• Submit Job
• Optimised Mapping

GAT (Data Management)

GW Data

CB Search

Gridlab Test-bed
The KEPLER/Ptolemy II GUI (Vergil)

“Directors” define the component interaction & execution semantics

Large, polymorphic component (“Actors”) and Directors libraries (drag & drop)
Actor-Oriented Design

- **Object orientation:**
  - What flows through an object is sequential control (cf. CCA, MPI)
  - What flows through an object is a stream of data tokens (in SWFs/KEPLER also references!!)

- **Actor/Dataflow orientation:**
  - What flows through an object is sequential control (cf. CCA, MPI)
  - What flows through an object is a stream of data tokens (in SWFs/KEPLER also references!!)

Source: Edward Lee et al. http://ptolemy.eecs.berkeley.edu/
Object-Oriented vs. Actor-Oriented Interfaces

Object Oriented

<table>
<thead>
<tr>
<th>TextToSpeech</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialize(): void</td>
</tr>
<tr>
<td>notify(): void</td>
</tr>
<tr>
<td>isReady(): boolean</td>
</tr>
<tr>
<td>getSpeech(): double[]</td>
</tr>
</tbody>
</table>

Actor/Dataflow Oriented

AO interface definition says “Give me text and I’ll give you speech”

OO interface gives procedures that have to be invoked in an order not specified as part of the interface definition.

Source: Edward Lee et al. http://ptolemy.eecs.berkeley.edu/
Ptolemy II: Actor-Oriented Modeling

- “Director” acts as an enactor
  - In this instance, interaction semantics are not maintained within a component
  - This is equivalent to having a centralised enactor

- Different directors for different modeling and execution needs
  - Hence, a variety of directors can operate on the same components

→ Better abstraction, modeling, component reuse, ...
Behavioral Polymorphism in Ptolemy

These polymorphic methods implement the communication semantics of a domain in Ptolemy II. The receiver instance used in communication is supplied by the director, not by the component. (cf. CCA, WS-??, [G]BPL4??, … !)

Behavioral polymorphism is the idea that components can be defined to operate with multiple models of computation and multiple middleware frameworks.

Source: Edward Lee et al. http://ptolemy.eecs.berkeley.edu/
Component Composition & Interaction

- Components linked via ports
- Dataflow (and msg/ctl-flow)
- Where is the component interaction semantics defined??
  - each component is its own director!
- But still useful for special applications, e.g. parallel programs (MPI, ...)

Source: GRIST/SC4DEVO workshop, July 2004, Caltech
Domains and Directors: Semantics for Component Interaction

- CI - Push/pull component interaction
- CSP - concurrent threads with rendezvous
- CT - continuous-time modeling
- DE - discrete-event systems
- DDE - distributed discrete events
- FSM - finite state machines
- DT - discrete time (cycle driven)
- Giotto - synchronous periodic
- GR - 2-D and 3-D graphics
- PN - process networks
- SDF - synchronous dataflow
- SR - synchronous/reactive
- TM - timed multitasking

For (finer-grained) concurrent jobs?

For (coarse grained) Scientific Workflows!

Source: Edward Lee et al. http://ptolemy.eecs.berkeley.edu/
Polymorphic Actor Components Working Across Data Types and Domains

- **Actor Data Polymorphism:**
  - Add *numbers* (int, float, double, Complex)
  - Add *strings* (concatenation)
  - Add *complex types* (arrays, records, matrices)
  - Add *user-defined types*

- **Actor Behavioral Polymorphism:**
  - In *dataflow*, add when all connected inputs have data
  - In a *time-triggered model*, add when the clock ticks
  - In *discrete-event*, add when any connected input has data, and add in zero time
  - In *process networks*, execute an infinite loop in a thread that blocks when reading empty inputs
  - In *CSP*, execute an infinite loop that performs rendezvous on input or output
  - In *push/pull*, ports are push or pull (declared or inferred) and behave accordingly
  - In *real-time CORBA*, priorities are associated with ports and a dispatcher determines when to add

By not choosing among these when defining the component, we get a huge increment in component re-usability. But how do we ensure that the component will work in all these circumstances?

Source: Edward Lee et al. [http://ptolemy.eecs.berkeley.edu/](http://ptolemy.eecs.berkeley.edu/)
Directors and Combining Different Component Interaction Semantics

Behavioral Polymorphism: Hierarchical Heterogeneity and Modal Models

Possible app. in SWF:
- time-series aware ...
- parameter-sweep aware ...
- XY aware ...
... execution models

Source: Edward Lee et al. http://ptolemy.eecs.berkeley.edu/ptolemyII/
“Minute-made” (MM) WS-based application integration
  • Similarly: MM workflow design & sharing w/o implemented components
KEPLER: Actors
FAEHIM

• Use of Web Services composition – with distributed services
  - Wrap third party services (Mathematica, GNUPlot)
  - WEKA Service template
  - Triana Workflow
• Services provided by third parties
  - WSDL interfaces (avoid use of specialist languages – unless really necessary)
  - SOAP-based message exchange
    - Use of attachments
• Access to local and remote data sets
  - Support for data streaming
• Wrapping of existing algorithms (important requirement)

http://users.cs.cf.ac.uk/Ali.Shaikhali/faehim/
FAEHIM Architecture
StringViewer

Value =

- Append values

- pruned tree

- play = yes
  - humidity <= 70: sunny (3.01, 0)
  - humidity > 70
  - temperature <= 71: rainy (2.0)
  - temperature > 71: overcast (4.01, 0)
  - play = no
  - temperature <= 71: rainy (2.0)
Inside the FAEHIM Toolbox

- **Deals with Dataset files**
  - e.g. load datasets, converts dataset formats

- **Deals with data that need manipulation**

- **Visualize data**
Classifier

- **J48 Classifier**

- **Operations**
  
  - `classify()`
    **Input:** DataHandler *dataset*, String *attributeName*
    **Output:** DataHandler *decisionTree*

  - `classifyRemoteDataset()`
    **Input:** String *url*, String *attributeName*
    **Output:** DataHandler *decisionTree*
Clustering Support

Cobweb Web Service

Operations

cluster( )
Input: DataHandler dataset
output: String result

clusterRemoteInstance( )
Input: String datasetURL
output: String result

clusterByPercentage( )
Input: DataHandler dataset, int percentage
output: String result
Graph Plotting Service

Plotting Web Service

Operations

plot3D( )
Input: DataHandler data, String plotType
output: DataHandler graph

getPlotTypes( )
Input: null
output: String plotTypes
Registry Usage

1. Discover
2. Select
3. Invoke

UDDI
Registry of Algorithms

Web Service

Execution Resource

Algorithm 1
Algorithm 2
Parallel Execution

1. Discover
2. Select
3. Invoke

- Resource Allocation Manager
- UDDI
- Registry of Algorithms
- Algorithm 1
- Algorithm 2
- Execution Resource
Workflow Optimisation

• Types of workflow optimisation
  - Through service selection
  - Through workflow re-ordering
  - Through exploitation of parallelism

• When is optimisation performed?
  - At design time (early binding)
  - Upon submission (intermediate binding)
  - At runtime (late binding)
Workflow Partitioning (Pegasus)

Full Graph vs Partial Graph Scheduling

Schedule
- Total workflow Graph
- Sub-graph
- Each node
• Late binding of abstract service to concrete service instance means:
  - We use up-to-date information to decide which service to use when there are multiple semantically equivalent services
  - We are less likely to try to use a service that is unavailable.
Late Binding Case

• Search registry for all services that are consistent with abstract service description.
• Select optimal service based on current information, e.g., host load, etc.
• Execute this service.
• Doesn’t take into account time to transfer inputs to the service.
• In early and late binding cases we can optimise overall workflow.
Work at Cardiff has focused on implementing a late binding model for dynamic service discovery, based on a generic service proxy, and service discovery and optimisation services.
Service Discovery Issues

• Service discovery and optimisation is based on service metadata.
• Could store in a database.
• Could obtain by interrogating service.
Use of Registry Services

Specialization vs. late-binding of function creates three options:

- No fixed actions
- One fixed action: the matching service comes with
  - A set of registries or
  - A set of matchers or
  - A selection function
- Two fixed actions: the matching service comes with
  - A set of registries and a set of matchers or
  - A set of registries and a selection function or
  - A set of matchers and a selection function
- Three fixed actions: a set of registries and a set of matchers and a selection function
• User writes a knoogle script
• commands:
  • submit-query: submit a query to be processed by the broker
  • get-query-status: get the identities of the query jobs currently being run by the broker
  • get-IDs: get the String id of all the queries being currently processed by the broker
  • triple-store: get the status of a query
  • terminate-query: terminate a query
  • validate-broker: validate the given broker
  • broker-status: return the status of a broker

• Accessible also as an API – enabling embedding in applications.
• Keep the API simple.

<querySubmission>
  <!-- the query document -->
  <query>
  <repositories>
    <!-- the repositories to be used -->
    <repository>* <!-- repository URL -->
    <matchers>?
    <!-- the matchers to be used -->
    <matcher>* <!-- match service URL -->
    <!-- the selection policy script to use -->
    <selectionPolicy>?
    <!-- selection policy script language -->
    <selectionPolicyLanguage>?
  ...
Taverna Configuration

- All inputs specified in knoole.properties File
  knoole.wsdlBroker =
  http://alis.cs.bath.ac.uk:9050/axis/services/Broker

- knoole.wsdlRepository =
  http://chost-comsc.grid.cf.ac.uk:8080/grimoires/services
  knoole.wsdlMatcher =
  http://host1:9050/axis/services/basicMatcher,
  http://host2:9050/axis/services/gridSAMmatcher

- knoole.wsdlSPSScriptName = SeRQL
  knoole.wsdlSPSScriptFile = SP1.serql

- Provision for multiple Repositories and Matchers
myGrid
Taverna Scufl Workbench v1.4

Advanced model explorer

Workflow model
Workflow inputs
Workflow outputs

Processors
Knoogle_GridSAM_Broker

Read_Text_File

Knoogle_GridSAM_Broker

Workflow Outputs

output gridsamWSDL

Available services

Search

Available Processors
Local Services
WSDL @ http://www.ebi.ac.uk/collab/mygrid
WSDL @ http://ails.cs.bath.ac.uk:9050/ax
WSDL @ http://chost-comsc.grid.cf.ac.uk:
WSDL @ http://chost-comsc.grid.cf.ac.uk:
WSDL @ http://www.ebi.ac.uk/xembli/XEM
WSDL @ http://soap.bind.ca/wsdl/bind.wsd
WSDL @ http://www.ebi.ac.uk/ws/services
WSDL @ http://seqhound.seqhound.blueprint.org
Optimisation by Re-Ordering

- Optimise the runtime execution of workflow before it is executed
- Achieves the goal through:
  - Re-ordering of components
  - Addition of components
  - Substitution of components
  - Pruning of the workflow
  - Performance and workflow aware Scheduling
  - Runtime Optimisation
    - through monitoring, check-pointing and migration
Component Manipulation

- Re-ordering: Workflows (often composed from composite workflows) may contain non-optimal ordering of components
  - Use re-ordering to improve performance
Pruning

- Workflow Pruning:
  - Workflows may contain unused components. Especially when composed from other sub-workflows
- Remove redundant components
Workflow Patterns

• Identify and reuse common “idioms” in some scientific domain and across different scientific domains.

• An “idiom” captures common knowledge and experience and describe how a similar set of experiments are to be set-up and managed.

From: Cecilia Gomes
Usage

1. To allow computational scientists and developers to capture *design patterns* that express common usage of software infrastructure within scientific domains.

2. To provide a software engineering tool that supports:
   • application configuration,
   • execution control, and
   • reconfiguration of software services.

From: Cecilia Gomes
Approach

• Patterns are divided in two categories for flexibility:
  - Co-ordination (Behavioural) patterns
    • Capture interactions between software sub-systems
  - Structural patterns
    • Capture connectivity between particular types of Grid software/hardware components

From: Cecilia Gomes
Approach

- **Patterns as first class entities** both at design, execution, and reconfiguration times
- **Pattern templates** are manipulated through **Pattern Operators**:
  - Structural operators
  - Behavioural operators

From: Cecilia Gomes
Structural Pattern Templates

- **Encode component connectivity.** Ex: Pipeline, Ring, Star, Façade, Adapter, Proxy.

From: Cecilia Gomes
Structural Operators

- Manipulate structural patterns keeping their structural constraints.
- Examples:
  - Increase, Decrease,
  - Extend, Reduce,
  - Embed, Extract,
  - Group,
  - Rename/Reshape, ...
Structural Operators

- Manipulate structural patterns keeping their structural constraints.
- Examples:
  - Increase, Decrease,
  - Extend, Reduce,
  - Embed, Extract,
  - Group,
  - Rename/Reshape, ...
Increase Structural Operator

Pattern

Increase(Pipeline,2)

Result Pattern

Real Subject
Proxy

Real Subject
Proxy
Proxy
Proxy

From: Cecilia Gomes
Extend Structural Operator

Pattern

Real Subject → Proxy

Extend(Proxy, element)

Facade

Result Pattern

Real Subject → Proxy → Proxy

Facade

Extend(Facade, element)

Facade

From: Cecilia Gomes
Behavioural Pattern Templates

- Capture temporal or (data/control) flow dependencies between components.
- Examples:
  - Client/Server,
  - Master/Slave,
  - Streaming,
  - Service Adapter,
  - Service Migration,
  - Broker Service
  - Service Aggregator/Decomposer, ...

From: Cecilia Gomes
Behavioural Operators

• Act over the temporal or flow dependencies for execution control and reconfiguration.

• Examples:
  - Start, Terminate,
  - Log,
  - Stop, Resume,
  - Restart, Limit,
  - Repeat, ...

From: Cecilia Gomes
• Applications are built by connecting services available in a toolbox
• The execution follows the dataflow model
3- Implementation over Triana - example
example
example
a second configuration
Workflow Planning/Adaptation

- Goal-oriented
- Abstract $\rightarrow$ Concrete workflow translation
  - May utilise a number of different infrastructure services (Pegasus)
- Level of automation can vary
  - Find components
  - Find sub-workflows
  - Find infrastructure services
  - Publish output data at specific locations
Planning

• *Situated so actions, percepts, time*
• An enactment process:
  - Monitors “Percepts”
  - Executes one or more “Plans”
  - Leading to “Actions”
  - Leading to new “Perceopt”

From: Michael Winikoff, RMIT
Planning ... 2

- **Reactive so events**
  (significant occurrence)
  - Percepts lead to internal events
  - Events need to be monitored with reference to “Goals”
  - **Goals** act as filters to decide “Actions”

From: Michael Winikoff, RMIT
Planning ... 3

• Implementation uses plans and beliefs
  ✐ Cache for means, and world information respectively
  ➢ Beliefs: contains information about current state of resources
  ➢ Plans: chose a schedule to meet a specific deadline

From: Michael Winikoff, RMIT
Chimera is developed at ANL
By I. Foster, M. Wilde, and J. Voeckler

From: Ewa Deelman

Chimera

Grid

Execution

Workflow

planning

Request

Manager

Concrete

Workflow

Dynamic
information

Replica

and Resource
Selector

Replica

Management

Data

Abstract Workflow

Available Resources

Data Publication

Transformation Catalog

Globus Monitoring and Discovery Service

Globus Replica Location Service

Workflow Reduction

from:

Monitoring information

Information and
Models

Raw data

detector

lesx

Virtaual Data Language

By I. Foster, M. Wilde, and J. Voeckler

Chimera

Grid

Workflow executor (DAGman)
Wings ... 2

- Uses: workflow templates, workflow instances and executable workflows
- Data
  - File
  - DataCollection (objects or files)
- Computation
  - ComponentType
  - ComponentCollection (hasComponentType property)
- Node
  - Node in a workflow
  - uses hasComponent to specify contained component
- Link
  - hasDestinationNode and hasOriginNode
  - hasDestinationFileDescription and hasOriginFileDescription
  - Subclasses: InputLink, InOutLink, OutputLink
  - Data collections carried in links with Skolem instances (stand in for actual data to be used in the instance)
HTN Planning (Activity Composition)

HTN Planning: Use of “Methods” (task decomp) and “Operators” (task execution)

Augment to describe Grid/web services
Augment to describe Grid/web requirements

“Initial” Plan

A1 → A2 → A4 → A5

“Final” Plan

A1 → A2.1 → A2.2 → A4 → A5

Plan Library

A2 Refinement
S1 ← S2

Introduce activities to achieve preconditions
Resolve interactions between conditions and effects
Handle constraints (e.g. world state, resource, spatial, etc.)

From: Austin Tate (Edinburgh)
HTN Planning (Initial Plan Stated as “Goals”)
Composer & Enactor

Collection of Available Semantic Web Services

I-Plan (Planning Service)

Goal

Policies Constraining Usage of Services

Partial Plan

Enforcement (e.g. via KAoS)

Final Plan

Enactment (e.g. via I-P2)

Partial Plan Amended with Policy Related Commentary

From: Austin Tate (Edinburgh)
BDI agents (based on AgentSpeak(L))

\[ \text{goal} : B_1 \land \ldots \land B_n \leftarrow S_1; \ldots; S_m \]

where each \( B_i \) is a belief, and each \( S_i \) is either an action (\( a \)), or a subgoal (\( a_{\text{sub}} \)).

The execution model of AgentSpeak consists of the following steps:

1. The agent selects an event \( e \) (note that goals are an event type)

2. The agent generates all plans with matching invocation conditions

3. From these relevant plans the agent identifies those with satisfied preconditions

4. If there are several plans, one is chosen non-deterministically

- Chosen plan added to “intention” stack (can be either an event (posted) or action (executed))
**BDI-based Enactor**

- Enactor can maintain local plan library
  - update of plan library as new conditions are detected
  - Useful in a dynamic environment (Grid) -- as agents are goal directed
- Execution of a plan leads to update of beliefs
  - useful mechanism to adapt agent behaviour in a Grid context
- Potentially useful to allow detection of plan conflicts
- Traditional approach:
  - number of tasks fixed, resources identical
  - fixed number of resources, tasks pre-defined
- Delegate scheduling priorities to each resource and task agent (no central schedulers)
Planning as Model Checking

• Planning based on:
  - **Non-determinism**: cannot predict interactions with external processes - i.e. cannot predict whether answer to a request for availability will be positive or negative
  - **Partial Observability**: can only observe external interactions (as BPEL) not internal status
  - **Extended Goals**: behaviour of the “process” is important, and not just the final goal
  - **Conditional Preferences**: may require multiple conditions to hold for goal to be satisfied

• **Given current state, evaluate possible likely states** (may require an exhaustive checking of possibilities)
Planning as Model Checking

- Planning under uncertainty
- Support different degrees of “run-time observability”
  - domain state partially visible via sensing
- “Temporally extended planning” goals
  - conditions on states that arise when a plan is executed
  - “goal” specifies conditions/constraints on intermediate states, and not just on the final outcome
• Domain \( \rightarrow \) Model of generic system
• Plan \( \rightarrow \) monitors evolution of domain via “observations”
  - Controls evolution of domain via “actions”
• Planning domain defined in terms of:
  - States, Actions (it accepts), Observations (domain can exhibit)
  - Transition function: action execution changes domain state
  - Observation function: observations associated with each state
Definition 1 (planning domain). A nondeterministic planning domain with partial observability is a tuple $D = (S, A, O, I, T, X)$, where:

- $S$ is the set of states.
- $A$ is the set of actions.
- $O$ is the set of observations.
- $I \subseteq S$ is the set of initial states; we require $I \neq \emptyset$.
- $T : S \times A \rightarrow 2^S$ is the transition function; it associates to each current state $s \in S$ and to each action $a \in A$ the set $T(s, a) \subseteq S$ of next states.
- $X : S \rightarrow O$ is the observation function.

Definition 2 (plan). A plan for domain $D = (S, A, O, I, T, X)$ is a tuple $P = (C, c_0, \alpha, \epsilon)$, where:

- $C$ is the set of plan contexts.
- $c_0 \in C$ is the initial context.
- $\alpha : C \times O \rightarrow A$ is the action function; it associates to a plan context $c$ and an observation $o$ an action $a = \alpha(c, o)$ to be executed.
- $\epsilon : C \times O \rightarrow C$ is the context evolutions function; it associates to a plan context $c$ and an observation $o$ a new plan context $c' = \epsilon(c, o)$.
Action Execution and Beliefs

- **Context**: internal state of plan
  - Account for knowledge gathered during previous steps
  - Actions: depend on observation and on the context
- Due to partial observability, a set of domain states need to be considered (given initial knowledge and current plan state)
  - Executing an action “a” evolves $B \rightarrow B'$ (contains all possible states that can be reached through “a” from “B”)
  - If after executing “a” observation “o” still holds, then filter out states for which “o” is not valid

$$Evolve(B, a, o) = \{ s' : \exists s \in B.s' \in T(s, a) \land X(s') = o \}.$$
Application to Web Services Composition

- First model the process undertaken within each service involved
- Synthesise, using planning, a process that interacts with the three processes (each service) in order to reach a particular state
- Aim to reach some “ideal” state – defined as an overall goal
Fig. 1.1: The user protocol

Fig. 1.2: The shipper protocol
Composed Web Service
Semantic Approaches

- Component/Services have “rich” annotations to aid discovery
- Descriptions also contain support for composition of components

From EU NextGRID project
Web Services Modelling Ontology (WSMO)

• Use of Semantic Web Services to aid automated composition
• Given a goal, identify how services could be composed to achieve the goal
• Specifies a complete set of infrastructure that is necessary to achieve this
• Provides three main components:
  - Web Services Modelling Ontology
  - Web Services Modelling Language
  - Execution Environment

From: John Domingue, Open University
WSMO Working Groups

A Conceptual Model for SWS

A Formal Language for WSMO

A Rule-based Language for SWS

Execution Environment for WSMO

From: John Domingue, Open University
WSMO Top Level Notions

Objectives that a client wants to achieve by using Web Services

Provide the formally specified terminology of the information used by all other components

Semantic description of Web Services:
- **Capability** (functional)
- **Interfaces** (usage)

Connectors between components with mediation facilities for handling heterogeneities

From: John Domingue, Open University
WSMO Mediators Overview

From: John Domingue, Open University
Mediator Structure

WSMO Mediator uses a Mediation Service via Source Component 1..n Source Component Target Component

- as a Goal
- directly
- optionally incl. Mediation

Mediation Services

From: John Domingue, Open University
DAML-S (Similar to OWL-S)

- Primarily aimed at Software Agents community
- Enable reasoning/planning about services
  - With particular support for automated composition
  - Integration with other information services

- Key aspect:
  - Notion of a service “profile”
  - Used to register service + support automated discovery via a matchmaking infrastructure
  - Use of “service advertisement”
  - Specifies inputs, outputs, pre-conditions, effects (post-conditions)
The **provenance of a piece of data** is the **process** that led to that piece of data.

- We represent the provenance of some data by **documenting** the process that led to the data:
  - documentation can be complete or partial;
  - it can be accurate or inaccurate;
  - it can present conflicting or consensual views of the actors involved;
  - it can provide operational details of execution or it can be abstract.

From: Luc Moreau
(U Southampton)
Provenance constituents

- The provenance of a data item is composed of several elements:
  - **Interaction provenance**: the set of all interactions between actors involved in the computation of the data
  - **Actor provenance**: the documentation provided by a particular actor pertaining to an interaction
  - **Grouping**: notion that allow us to give a scope, in terms of execution semantics and application's needs.
Provenance Questions

• After completion of workflow:
  1. Did the services I use actually fulfil my overall application requirement?
  2. Two of the analysis were performed on the same initial data but have different results – did I alter the services between these experiments?
  3. Did I perform each service on the type of data that the service was intended to analyse, i.e. were the inputs and outputs of each activity compatible?
  4. Did I use data sources from the same site?
  5. Why did it take much longer to run the analysis in the second instance?

Particularly significant in the context of Distributed Services
p-assertion

- A given element of process documentation referred to as a p-assertion
  - p-assertion: is an assertion that is made by an actor and pertains to a process.
- Types
  - Interaction p-assertion
    - relates to content of received/sent message
  - Actor p-assertion
    - Relationships between actors
    - State of an actor

From: Luc Moreau
(U Southampton)
Provenance architecture

Application

Client (actor 1)

Service A (actor 2)

Service B (actor 3)

Provenance Store

Record Documentation of Execution

Results

Validate workflow execution using provenance tools

Get results

User
A given element of process documentation referred to as a p-assertion

- **p-assertion**: is an assertion that is made by an actor and pertains to a process.

**Types**

- Interaction p-assertion
  - relates to content of received/sent message
- Actor State p-assertion
  - State of an actor
- Relationship p-assertion
  - Relationships between interaction p-assertions

From: Luc Moreau
(U Southampton)
From these p-assertions, we can derive that M3 was sent by Actor 1 and received by Actor 2 (and likewise for M4).

If actors are black boxes, these assertions are not very useful because we do not know dependencies between messages.
These assertions help identify order of messages, but not how data was computed.
These assertions help identify how data is computed, but provide no information about non-functional characteristics of the computation (time, resources used, etc).
I used IBM cluster
Request was in queue for 6 min

I used SPARC processor
I used algorithm x version x.y.z
Types of p-assertions (1)

- **Interaction p-assertion**: is an assertion of the contents of a message by an actor that has sent or received that message.

  - I received M1, M4
  - I sent M2, M3
- **Relationship p-assertion**: is an assertion, made by an actor, that describes how the actor obtained an output message sent in an interaction by applying some function to input messages from other interactions (likewise for data)

\[
\begin{align*}
\text{M2 is in reply to M1} \\
\text{M3 is caused by M1} \\
\text{M2 is caused by M4} \\
\text{M3 = f1(M1)} \\
\text{M2 = f2(M1,M4)}
\end{align*}
\]
Types of p-assertions (3)

- **Actor state p-assertion**: assertion made by an actor about its internal state in the context of a specific interaction

I used sparc processor
I used algorithm x version x.y.z
Actor State Capture

- **Instrumented Output**: Service information obtained from instrumented points within an actor.
- **Monitoring Sources**: Service information derived from hosting platform via monitoring sources (e.g., Ganglia).

Diagram:
- **Service**
- **Enactment Engine**
- **M1** and **M2**: Monitor Output
- **B1** and **B2**: Instrumented Output

The diagram illustrates the flow of information from the service to the enactment engine, highlighting the integration of monitoring sources and instrumented actor data.
Metrics for Actor State Assertion

• **Static**
  - No variation in value over actor lifetime
    • Per Node - Node identity, Operating system
    • Per Actor - Actor identity, Name, Owner, Version

• **Dynamic**
  - Variation in value over actor lifetime
    • Per Node - Memory usage, Network traffic
    • Per Actor - Execution Time, Availability

• **Instrumented**
  - Actor is 'Instrumented' at Key Points in its Execution
    • Description of internal data flow
  - Eg. Completion states for action events and file transfers
The p-structure (1)

- The p-structure is a common logical structure of the provenance store shared by all asserting and querying actors
- Hierarchical
- Indexed by interactions (interaction= 1 message exchange)
  - Now part of the Open Provenance Model
The p-structure (2)

All p-assertions asserted by a given actor participating in an interaction

Asserter identity
P-Assertion schemas
Implementation Diagram

From: Luc Moreau (U Southampton)
Provenance Store Components

Slide from John Ibbotson (IBM)
Portal – Tools Architecture

• Tool suite allow users of the tool to navigate and visualize provenance information beyond the capabilities provided by the Client Side Library.

- The Visualisation Tools: these tools provide Graphical User Interfaces (GUI) for visualizing p-assertions that have been submitted by an application.
Portal – Tools Architecture

- **The Processing Tools**: provide features accessible through an Application Programming Interface (API). The processing tools include the following:
  - The Analysis Engine provides reasoning capabilities over a set of passertions,
  - The Comparator Tool may be used to compare passertions that have been submitted by an application,
  - the Query Tool makes use of the Client Side Library to query one or more Provenance Store(s).
Number of interactions between actors

OTM:CollectPatientData

OTM:TestingLab

OTM:DecisionMaking

OTM:faceSender

Actor

Change View

View 1: Hierarchical
View 2: Short Distributed
View 3: Extended Distributed
View 4: Cluster
View 5: Compact Tree
View 6: Radial tree
View 7: Tree
Reset

Zoom

Group Actors
Reynolds number used as part of the process

Different views provide detail information of an interaction
Relationship Tool
References

• “A Taxonomy of Workflow Management Systems for Grid Computing”, Jia Yu and Rajkumar Buyya, GRIDS Lab, University of Melbourne


• “Putting Semantics in Grid Workflow Management: the OWL-WS approach”, Stefano Beco, Barbara Cantalupo, Nikolaos Matskanis, Mike Surridge, EU NextGRID paper

• “Planning and Monitoring Web Service Composition”, M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso (ITC/IRST, Italy)
References ... 2