On the Black Art of Designing Computational Workflows

Yolanda Gil
Pedro González
Ewa Deelman

Group for Artificial Intelligence Applications
Complutense University of Madrid

Outline

- Motivation: Wings/Pegasus
- Process of Workflow Design
- Related work and conclusions

Motivation

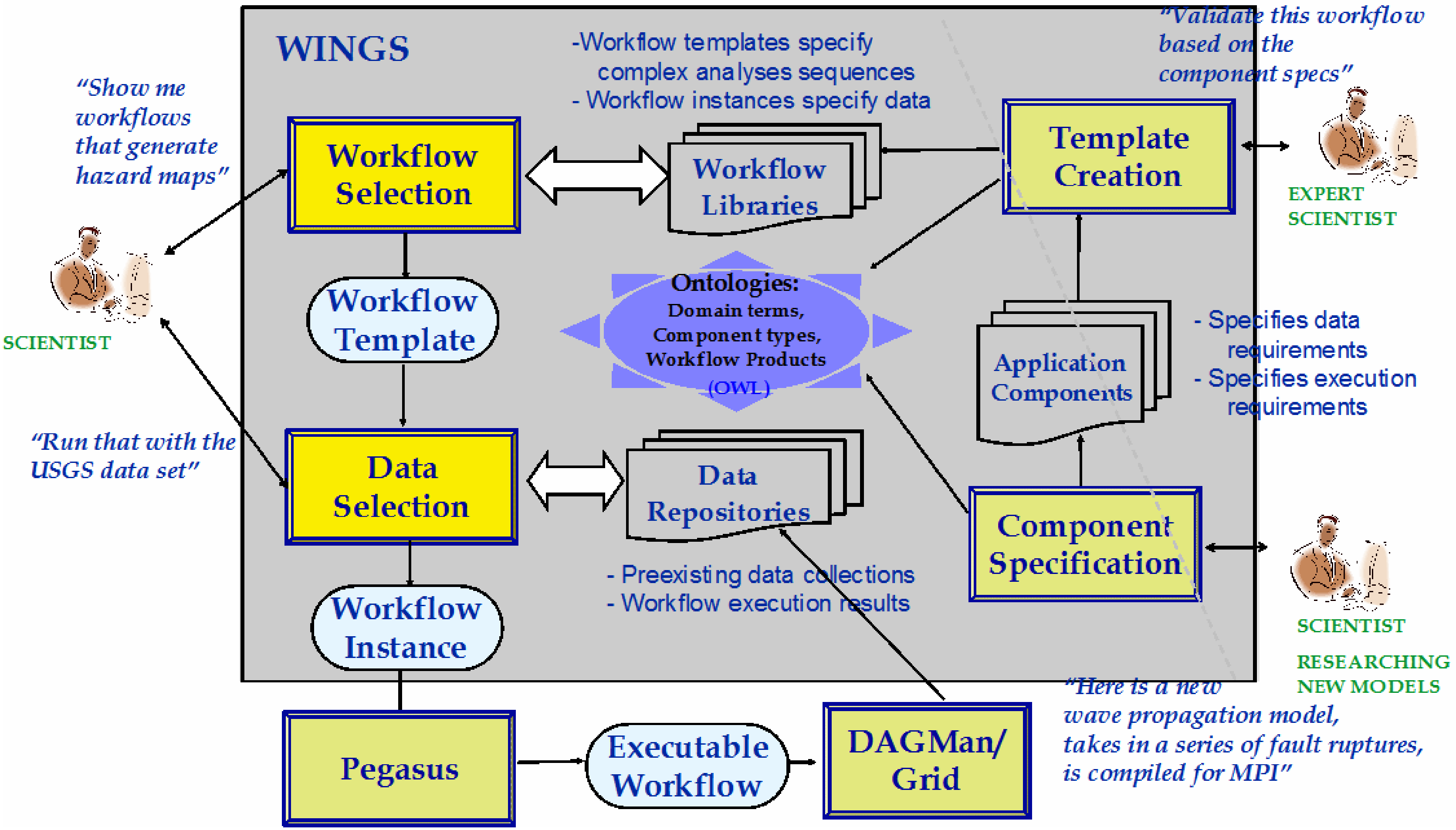
Motivation

- Computational workflows are composed of portable codes that can be submitted for execution to several alternative execution resources, process large-scale datasets, and can be easily restructured to exploit parallel data processing
- Current approaches: scripts to create thousands of jobs and the dataflow among them. Scripts are workflow-specific and costly to create, debug and evolve
- It is important to understand the sources of cost of creating a workflow-based application and investigate whether current workflow systems can be extended to assist the process and reduce the effort required

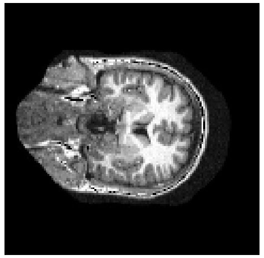
Wings/Pegasus Framework :: Creation of Large-Scale Grid Workflows

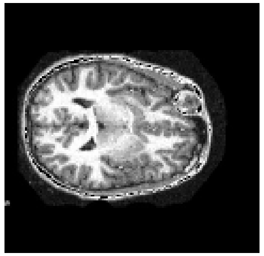
- Workflow Template (generic known-to-work recipes)
  - Specifies application components and dataflow among them
  - No data specified, just their type
- Workflow Instance (data-specific)
  - Specifies data files for a given template
  - Expands parallel data processing steps
  - Logical file names, not physical file replicas
- Executable Workflow (actual run)
  - Specifies physical locations of data files (may be in data repositories)
  - Assigned hosts/pools for execution of each component
  - Expand workflow to includes data movements among execution sites
  - Reduce workflow by reusing previously executed computations
  - Restructure workflow by grouping related executions for efficiency

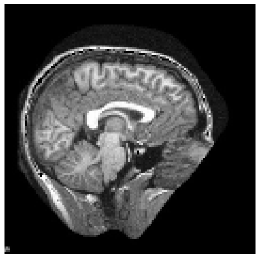
Wings: Workflow Instance Generation and Selection

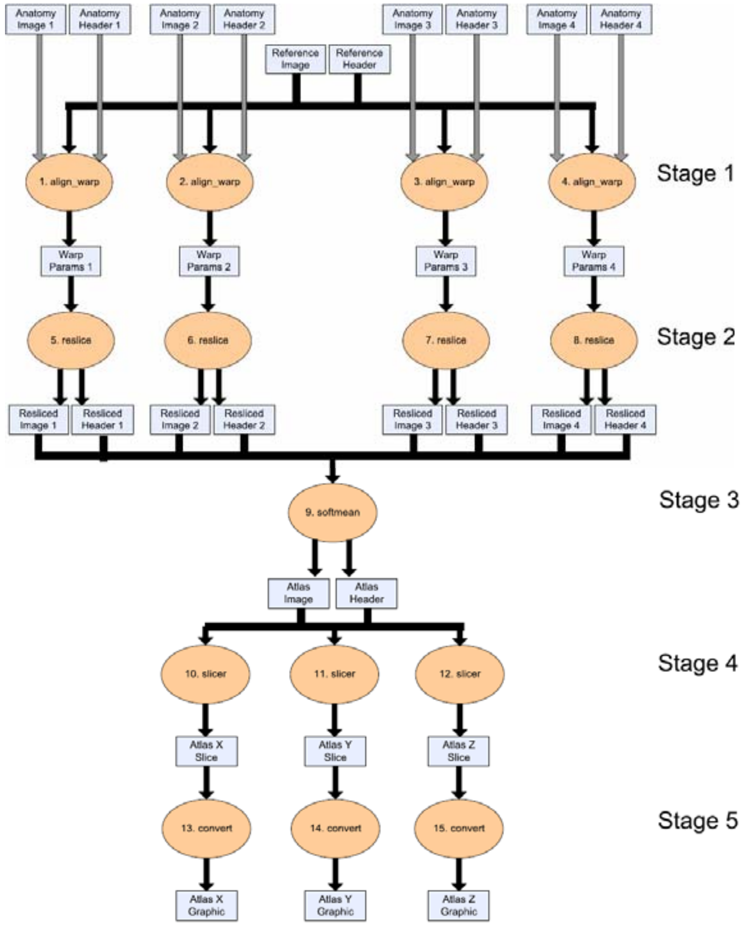


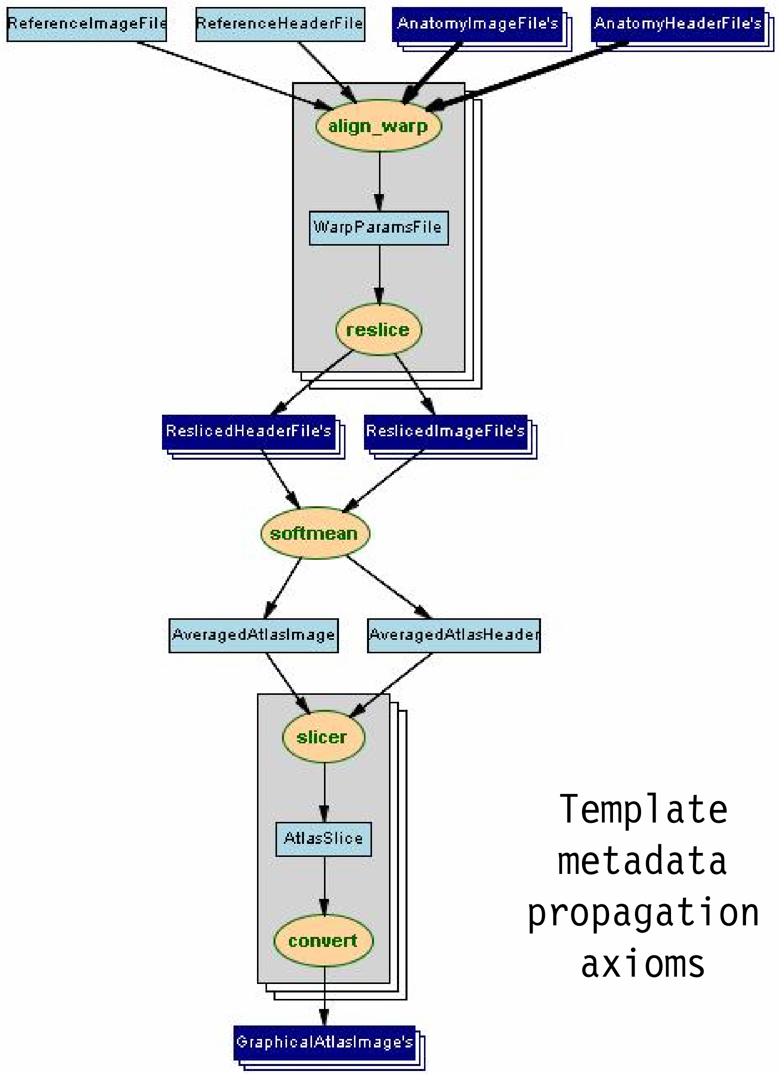
Workflow Sketch :: Workflow Template







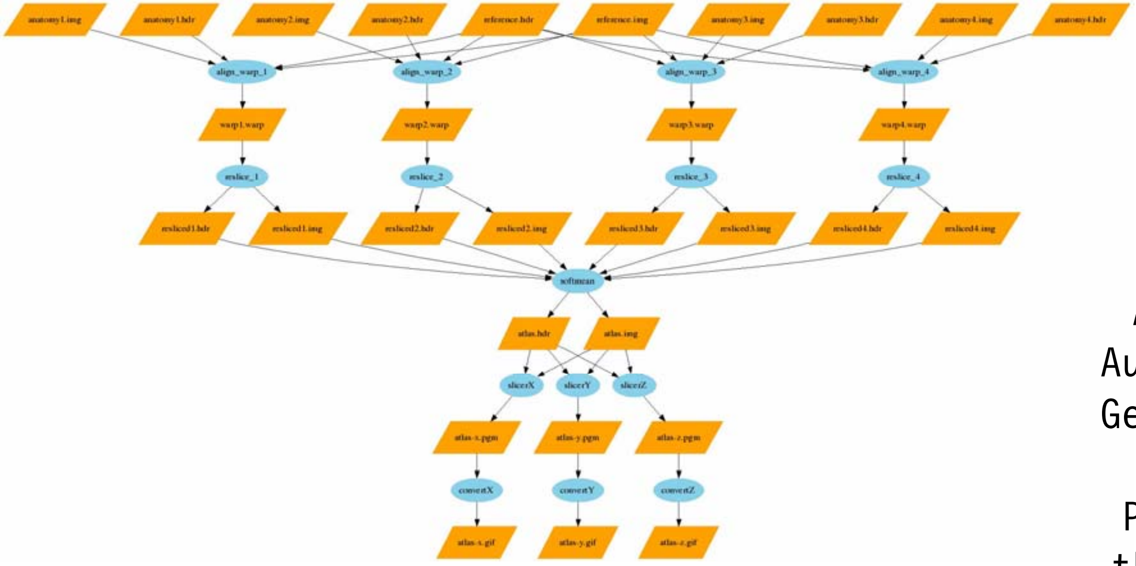




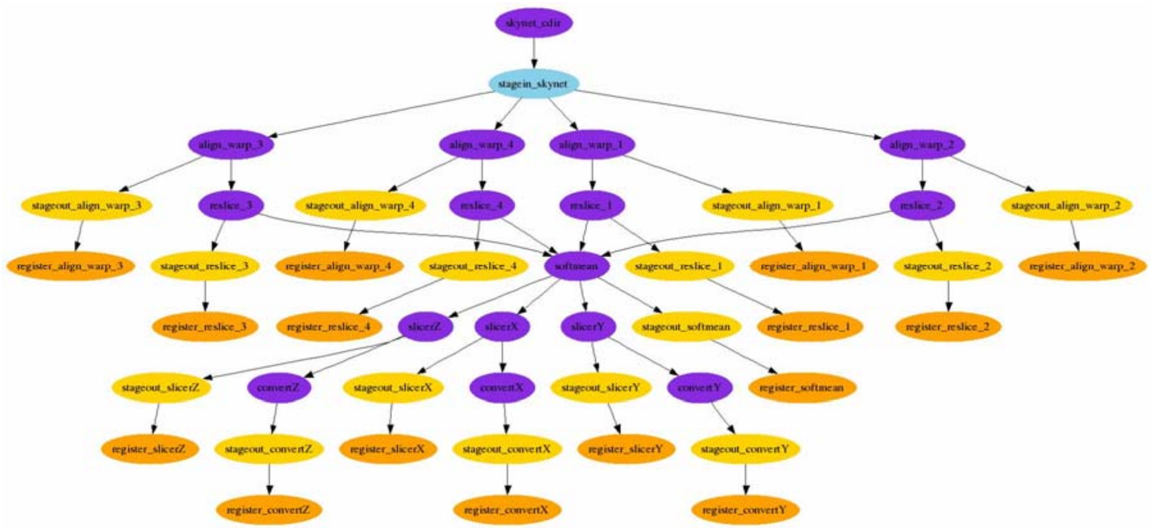
Template metadata propagation axioms

Workflow Instance :: Executable Workflow

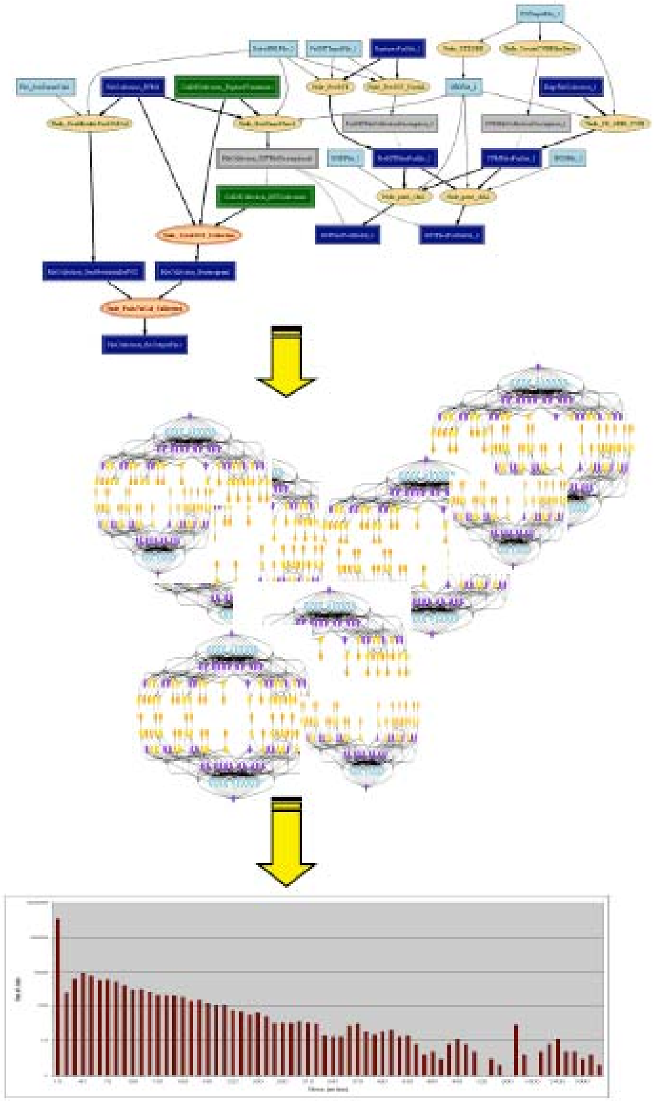
metadata of actual input data



Metadata Attributes Automatically Generated for New Data Products of the Workflow



It Works :: Seismic Hazard Analysis



- Input data: a site and an earthquake forecast model
  - thousands of possible fault ruptures and rupture variations, each a file, unevenly distributed
  - ~110,000 rupture variations to be simulated for a given site
- 8043 application nodes in the workflow instance generated by Wings
- 24,135 nodes in the executable workflow generated by Pegasus, including:
  - data stage-in jobs, data stage-out jobs, data registration jobs
- Executed in USC HPCC cluster, 1820 nodes w/ dual processors) but only < 144 available
  - Including MPI jobs, each runs on hundreds of processors for 25-33 hours
  - Runtime was 1.9 CPU years
- Significant contribution to create a more accurate seismic hazard map for SoCal
  - First integration of multiple physics-based models
  - Currently fine-tuning and cross-validating models

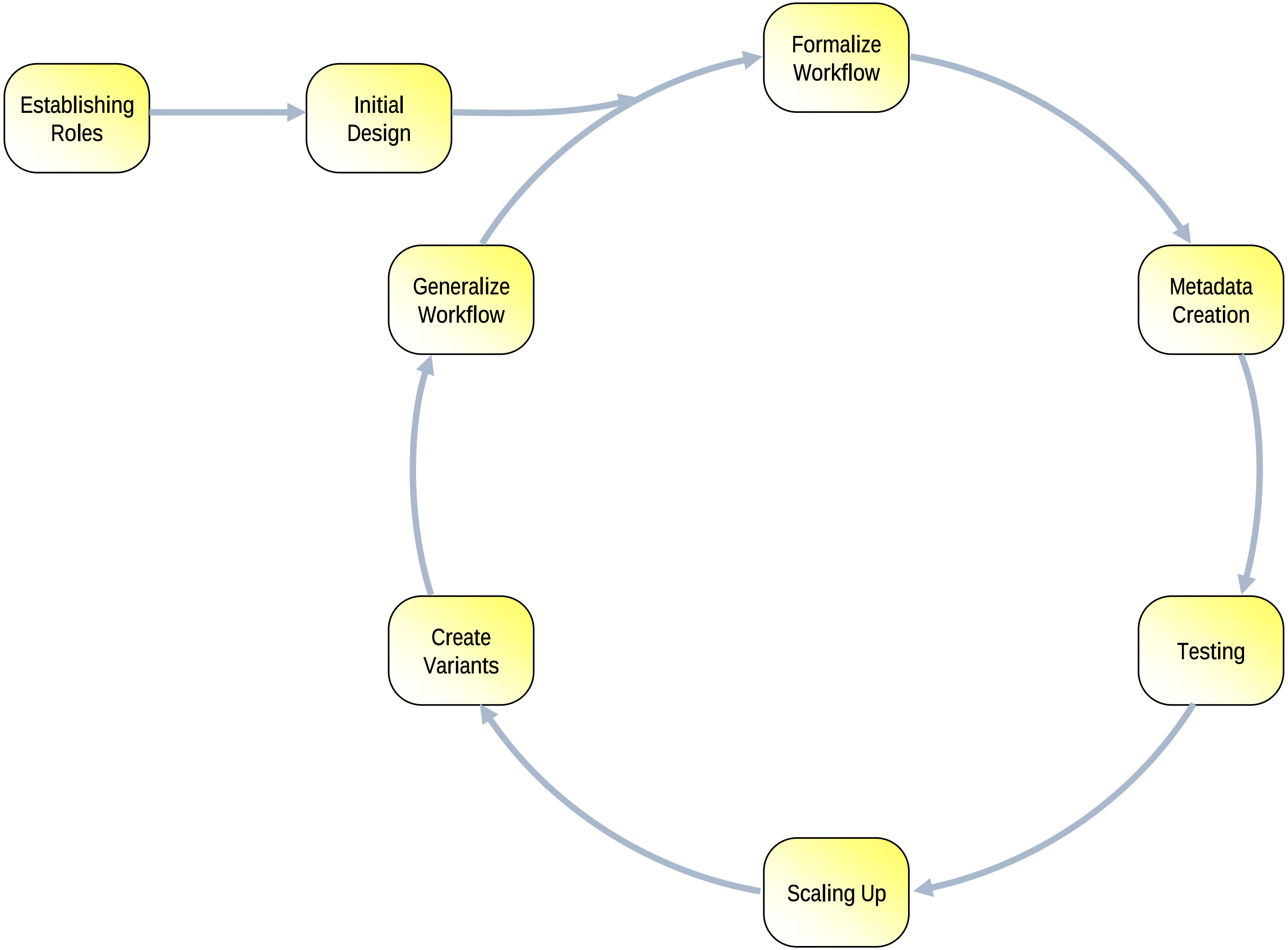
Benefits of Adopting a Workflow-based Approach

- Provide a clear separation between domain-relevant user concerns and execution details
  - Allows for complex optimizations
  - Promotes understandability to domain users
- Result validation
  - Workflow templates guarantee the results were obtained using a widely-accepted analysis methodology
- Accelerate experimental cycle
- Document experimental results
- Broaden participation in the experimental cycle
- Facilitate scaling up
  - Seamless transition to executing with high-end computing resources

Process of

Workflow Design

Process of Workflow Design



Establishing Roles

- Define scientist and engineer roles of each participant in the design process
- Define role of the system in assisting and automating various aspects of workflow creation
- Desirable tools and assets
  - a common understanding: terminology, automation required
  - examples of prototypical workflows, illustrations of bad workflow design and inappropriate uses of the workflow system

Scientist

SW Eng.

Programmer

Execution

Representation

Initial Design

- Identifying components and data and understanding their dependencies in the computations
- Clean up the codes: remove hard-coded control, remove data movement components, add failure reporting
- Design of workflow sketch with directed acyclic data flow
- Desirable tools and assets
  - workflow sketching
  - light-weight knowledge acquisition tools
  - source code analysis tools

Scientist

Programmer

Execution

Representation

Formalize Workflow

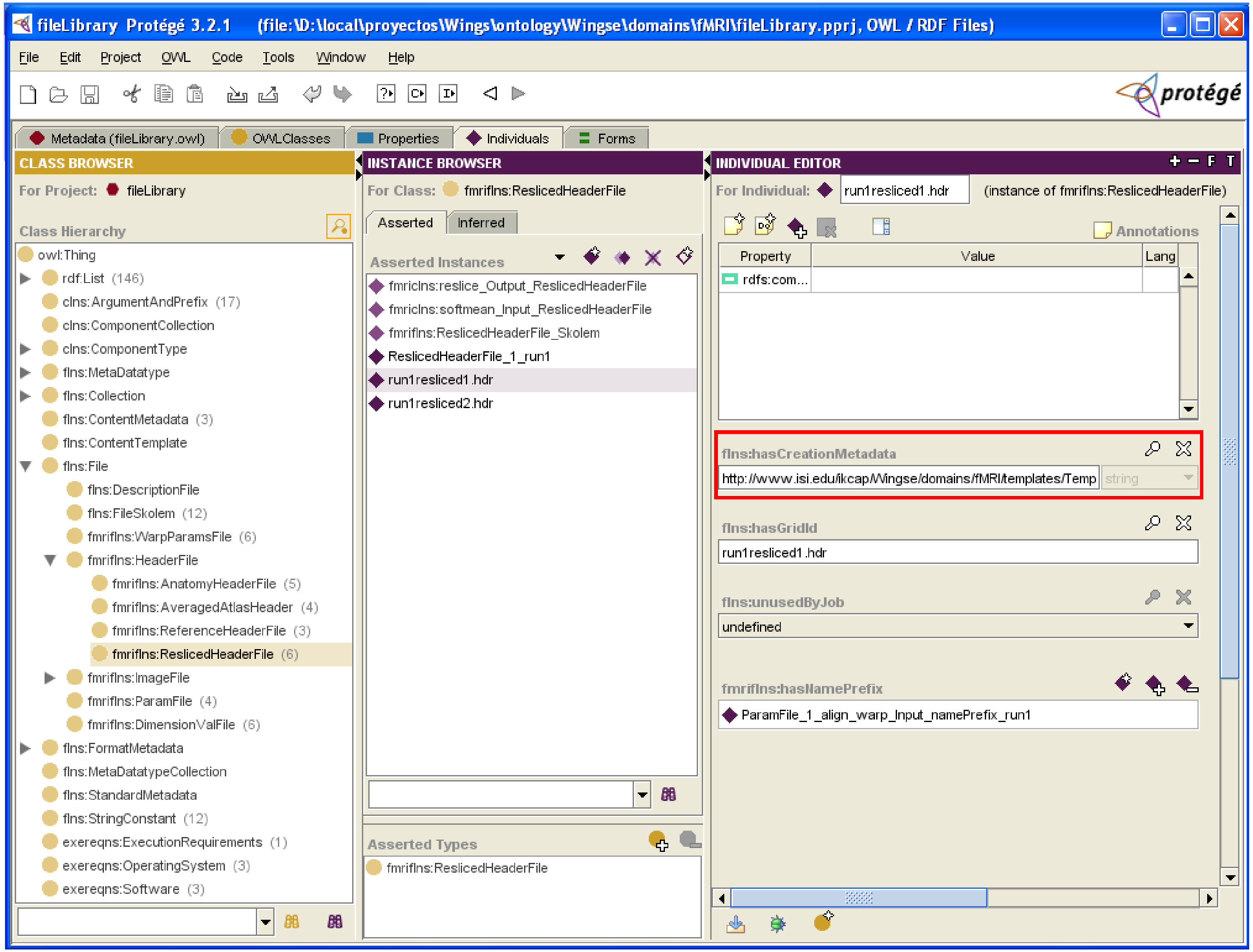
- Build workflow template
- Software Component Modeling
  - input/output
  - execution requirements
- Write additional components as needed
- Desirable tools and assets
  - Dedicated authoring tools
  - Checking and maintaining coherence between model and source code

Scientist

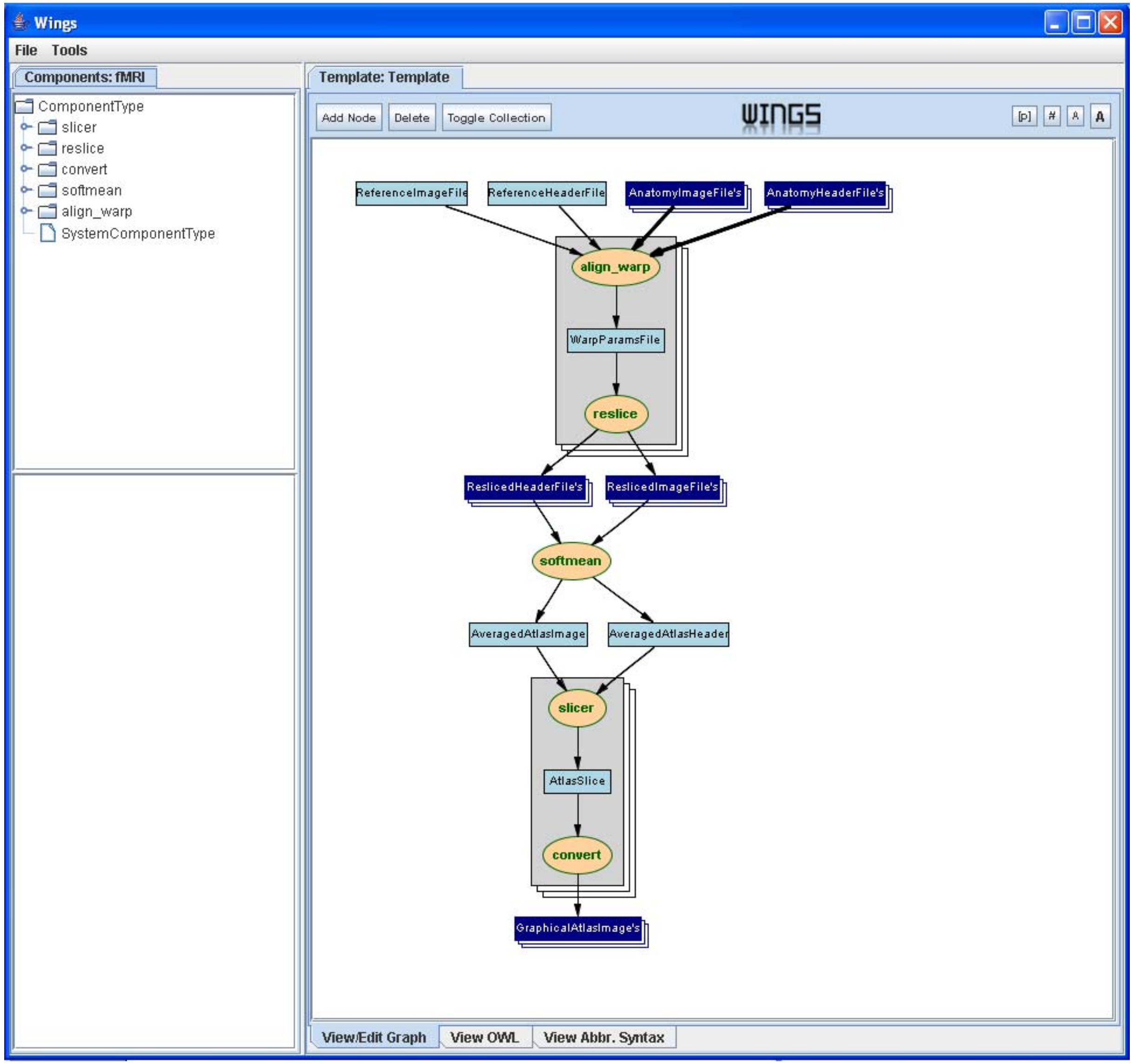
Programmer

Representation

Formalize Workflow :: Tool Support



Formalize Workflow :: Tool Support



Metadata Creation

- Creation of rules for propagation of metadata from input data through each component
- Describe using metadata constraints the requirements of the template from input datasets
- Describe using metadata constraints the characteristics of final workflow data products
- Desirable tools and assets
  - Dedicated authoring tools
  - Knowledge acquisition tools


Scientist


SW Eng.


Representation

Testing

- Verification of compliance of codes with component and metadata definitions
- Validation of models by executing workflows using small data sets
- Validation of models and workflow with known data sets and results
- Desirable tools and assets
  - benchmark datasets
  - unit tests for components and workflows

Execution

Representation

Scaling Up

- Identify bottlenecks in execution by running workflows with larger data sets
- Identify workflow strands that could process data in parallel
- Add data splitting and data merging components
- Desirable tools and assets
  - High-level analysis tools to connect execution logs to elements in workflow templates and instances

Scientist

SW Eng.

Programmer

Execution

Create Variants

- Define new workflow templates with varying parameter values
- Define new workflow templates with alternative codes for a component
- Define semi-instantiated workflow templates by specifying default datasets
- Desirable tools and assets
  - Component and workflow versioning tool support

Scientist

Representation

Generalize Workflows

- Identify commonalities between workflows that can be modeled as abstract components
- Define workflow templates using component classes and criteria to select among specializations
- Desirable tools and assets
  - workflow catalogs: indexing and reusing workflows from a shared library

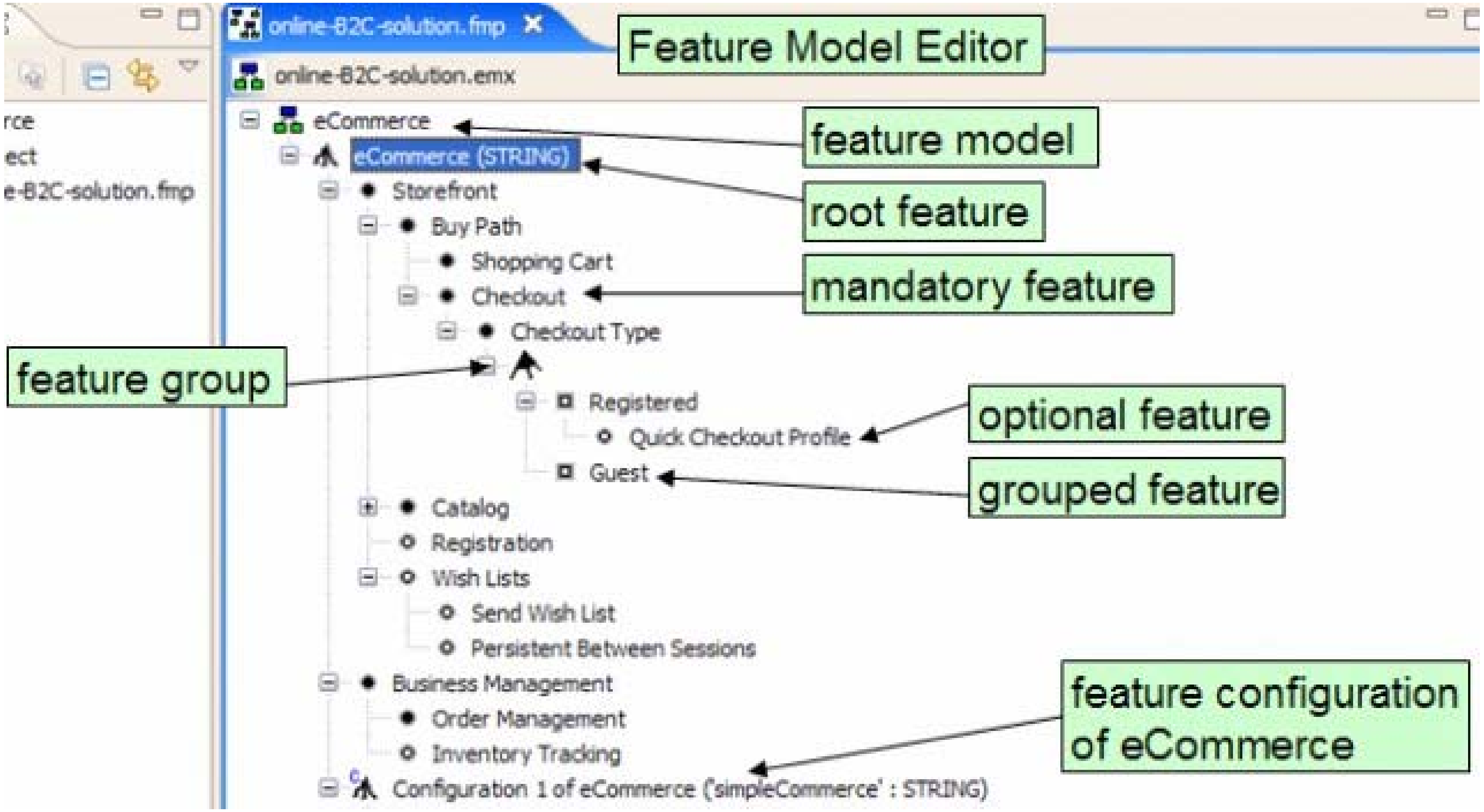

SW Eng.


Programmer


Representation

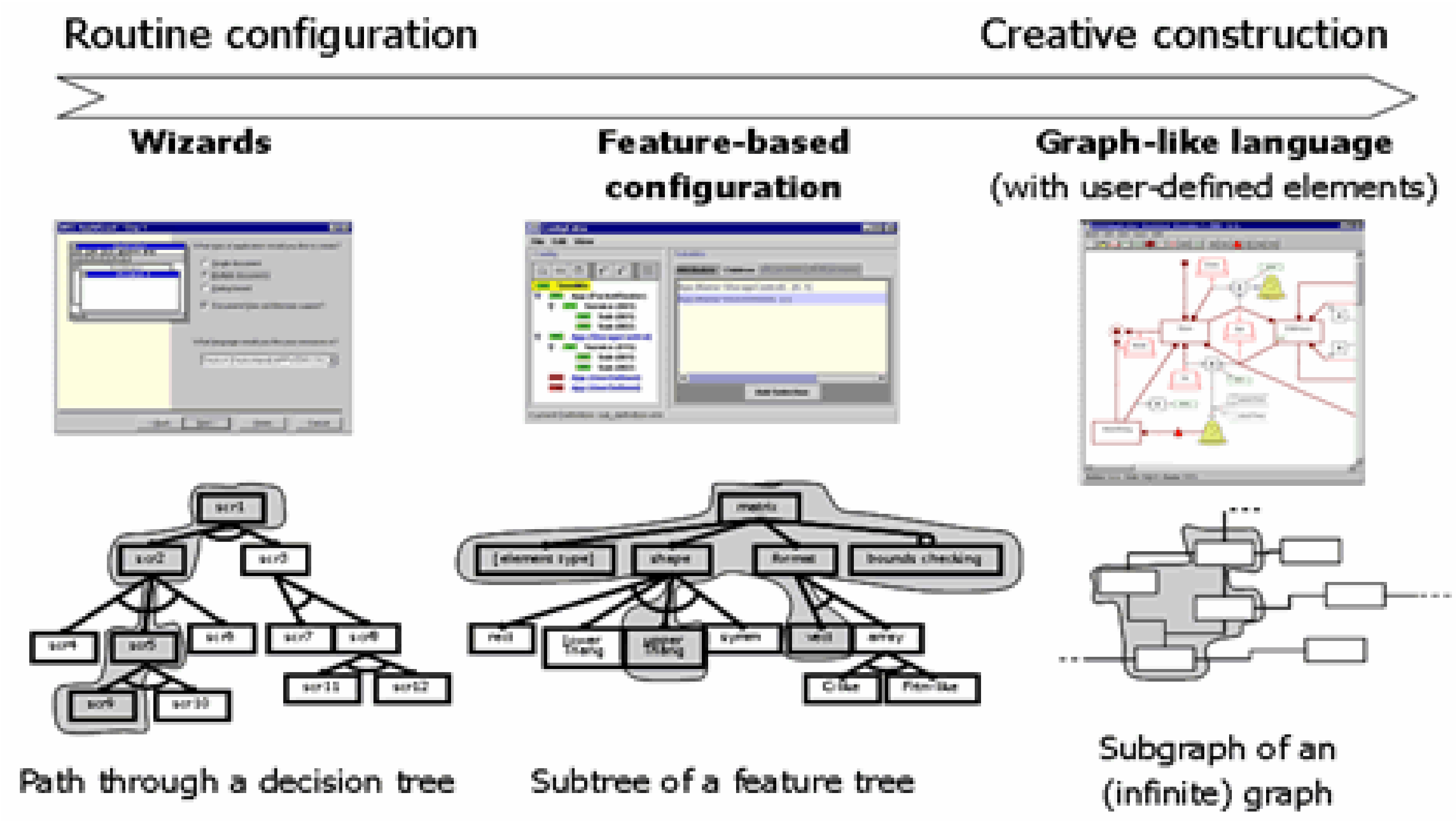
Related Work
and
Conclusions

Related Work in Software Engineering :: FODA



Krzysztof Czarnecki Eclipse plug-in for feature modeling

Related Work in Software Engineering :: Software Product Lines



Jack Greenfield, 2004:
Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools

Conclusions

- Workflow systems that support large-scale computation-intensive scientific applications could revolutionize many sciences
- Workflow systems widespread adoption depends on a design methodology that offers enough support and automation to make the process cost-effective
- By articulating the benefits of workflow applications and by reducing the cost of developing them, our goal is to make workflow technologies accessible to a broader community of users with applications where computation and scale are important issues