Enabling Model Evolution via a Repository

Dan Matheson
Robert France
James Bieman
Roger Alexander
James DeWitt
Nathan McEachen

Goals

- Support software engineering research
  - Model Driven Architecture
  - Model Driven Software Development
  - Aspect Oriented Modeling and Programming
- Develop a solution that is practical in the real world
  - Show viability of an independent repository
  - Show processes and engineering techniques that work

Motivations

- Experiences from applying MDA by hand in 2002 & 2003
- Experiences from AOM research
- Reverse engineering / product merge experiences
- Economic realities of the last few years (layoffs, offshoring), the thinking that India or China is more cost effective
- Stop re-inventing data for research

Requirements, part 1

- Manage software solution artifacts from first solution proposal through code development
- Tool independence
  - Work with existing tools
  - Able to add new tools with little disruption
- Support multiple types of data and relationships among data
- Easy to extend for new data types and new relationships

Requirements, part 2

- Allow any tool to access any data
  - No fixed tool data access rules, other than normal access control
- Support multiple tool repository interaction protocols
- High data integrity in a shared data environment
- Scalable and distributed
- Non proprietary, open research environment

Some Managed Artifacts

- Managing model transformations in MDA
- Managing multiple models and composition directives from AOM
- Managing code components
- Managing model to code relationships
- Creating a catalog of components for re-use
- Tracking design evolution for team communication

Approach

- Leverage learnings from other engineering disciplines
  - Discrete manufacturing (mechanical, electrical)
  - Civil engineering
- RM-ODP to organize project design
- Use Open Source tools as much as possible
- Use experience of team members
  - 60+ years industry product/solution development

Leveraged Learnings, Problems

- Discrete manufacturing
  - Conflicting requirements
  - Part versions
  - Assembly versions
  - Access control
  - Manufacturing processes
- Software manufacturing
  - Conflicting requirements
  - Software versions
  - Component sizes
  - Access control
  - Build processes

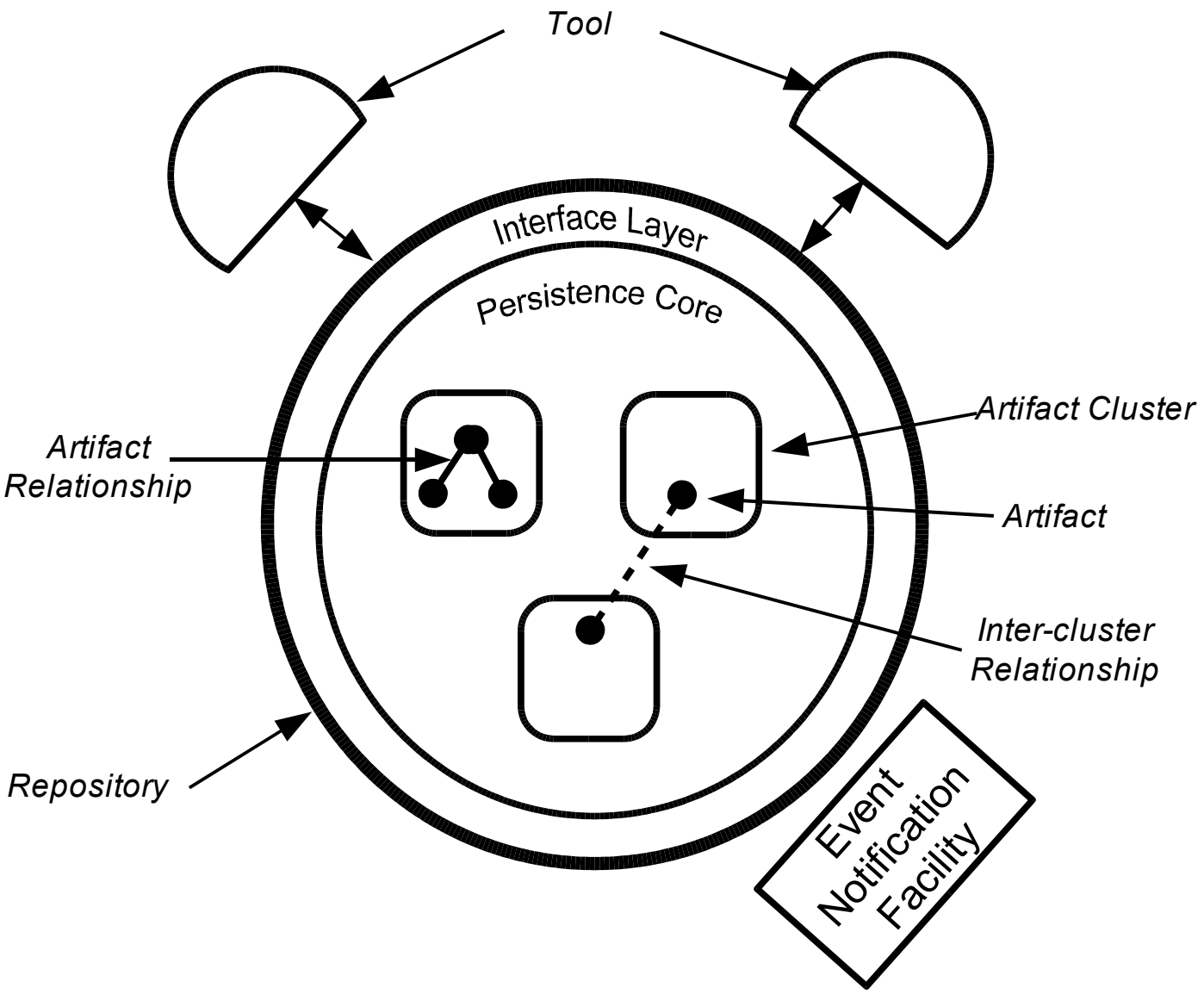
Leveraged Learnings, a Solution

- Product Data / Lifecycle Management
  - The electronic equivalent of manufacturing specs
  - Manages product structure: parts, assemblies, versions, documentation
  - Manages manufacturing processes
  - Manages product configurations
  - Manages change process and change data
- Use Standards rather than re-invent
  - OMG PDM Enablers, XML, etc.

Leveraged Learnings, Solution

- Product development tool structure
  - Independent PDM repository
  - Multiple tools from multiple vendors, from CAD to specialized analysis
- Product development process
  - Tools only depend on data availability (stable)

Responsibility Architecture



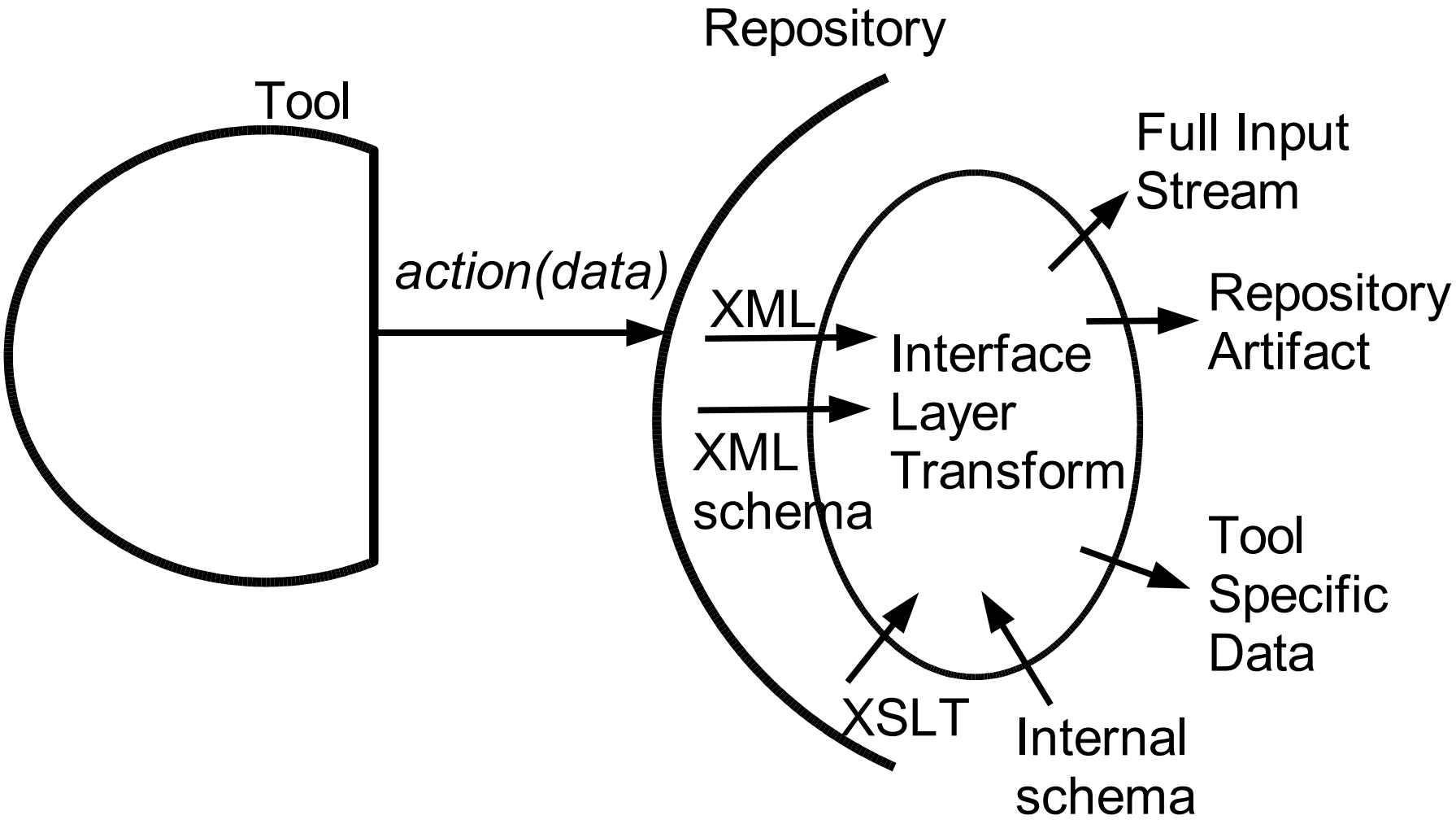
Responsibility Architecture

- Tool – creates, reads, modifies, deletes artifacts and relationships
- Repository – tool independent artifact storage (ACID on CRUD)
  - **Interface Layer** – interface between tools and the persistence core
  - **Persistence Core** – storage and basic data integrity
  - **Artifact** – atomic data storage unit
  - **Complex Artifact** – assembly level storage unit
  - **Artifact Cluster** – logic grouping of artifacts (project)
  - **Artifact relationship** – dependency between 2 artifacts
- Notification Facility – event propagation and handling

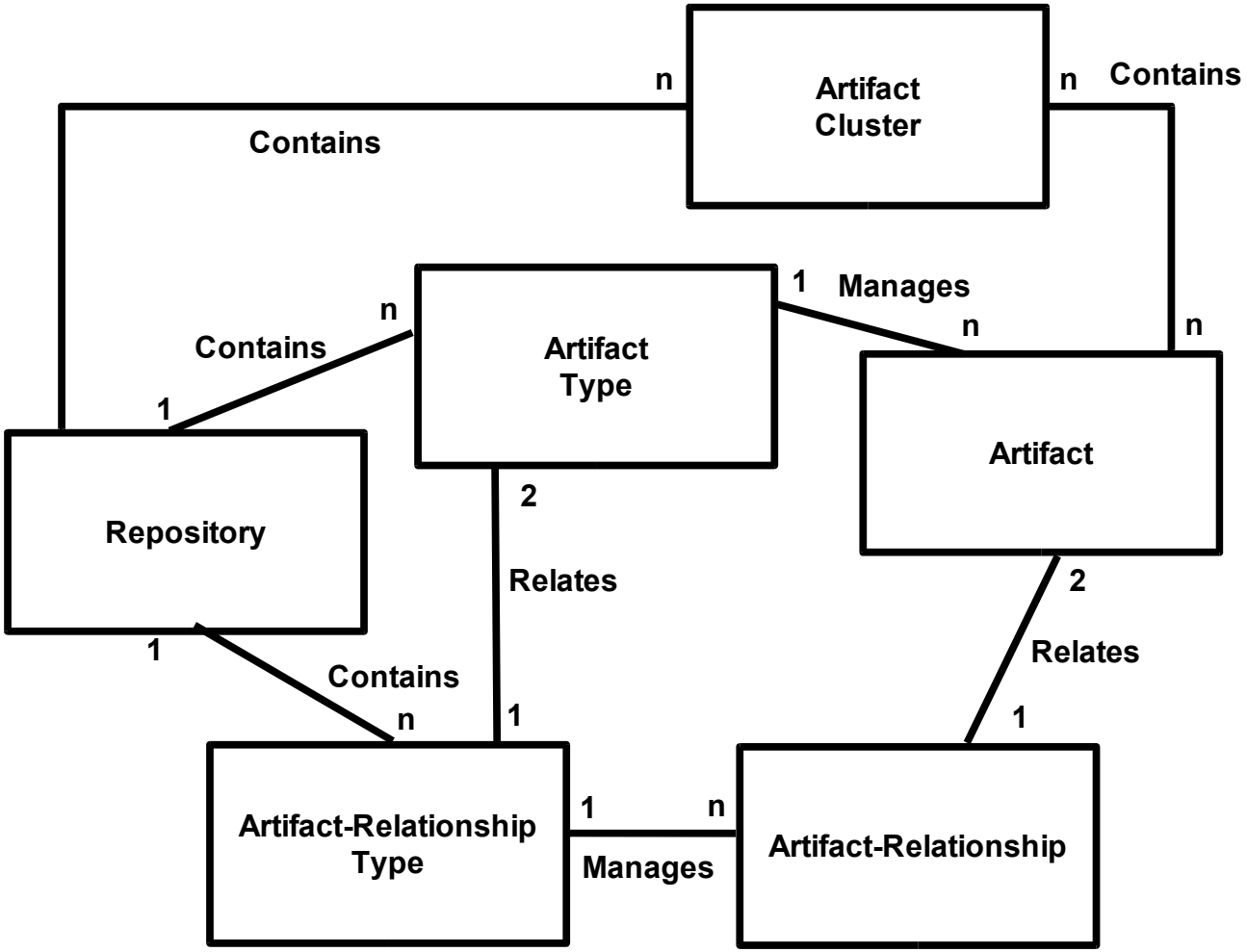
Process Flexibility

- 3 orthogonal components that can change with smaller cost (think of UNIX pipes).
  - ***Tools*** for creating and modifying data
  - A ***repository*** for preserving data and relationships
  - Multiple ***processes*** coordinating data creation and validation
    - Tool to tool, tool to repository, repository change to tool (process)
- As new tools are developed, processes can be modified or new processes created.
- As business goals change or SWE improves, processes can be modified without requiring new tools.

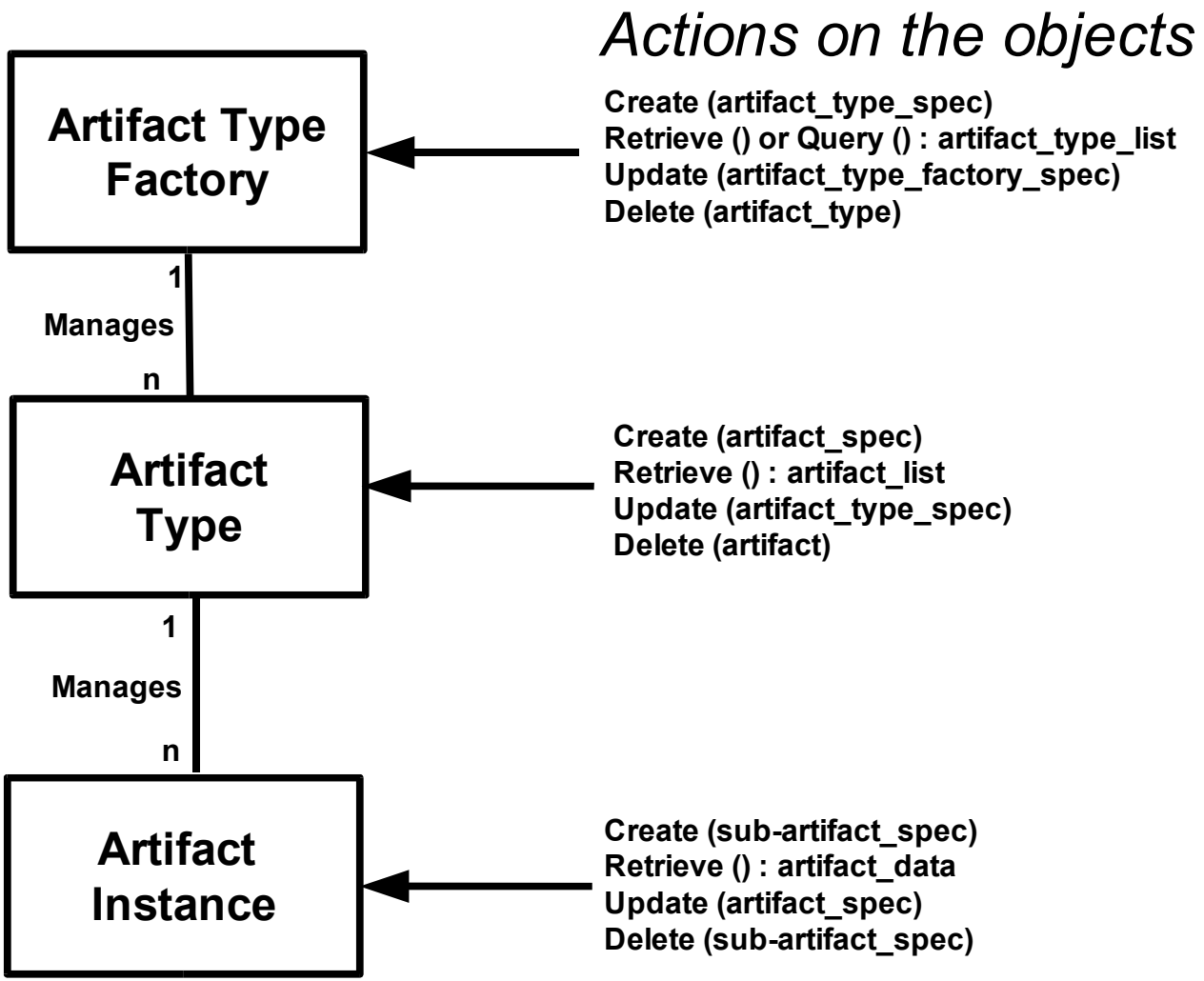
Interface Layer Interaction Design



Abstract Artifact Structure Design



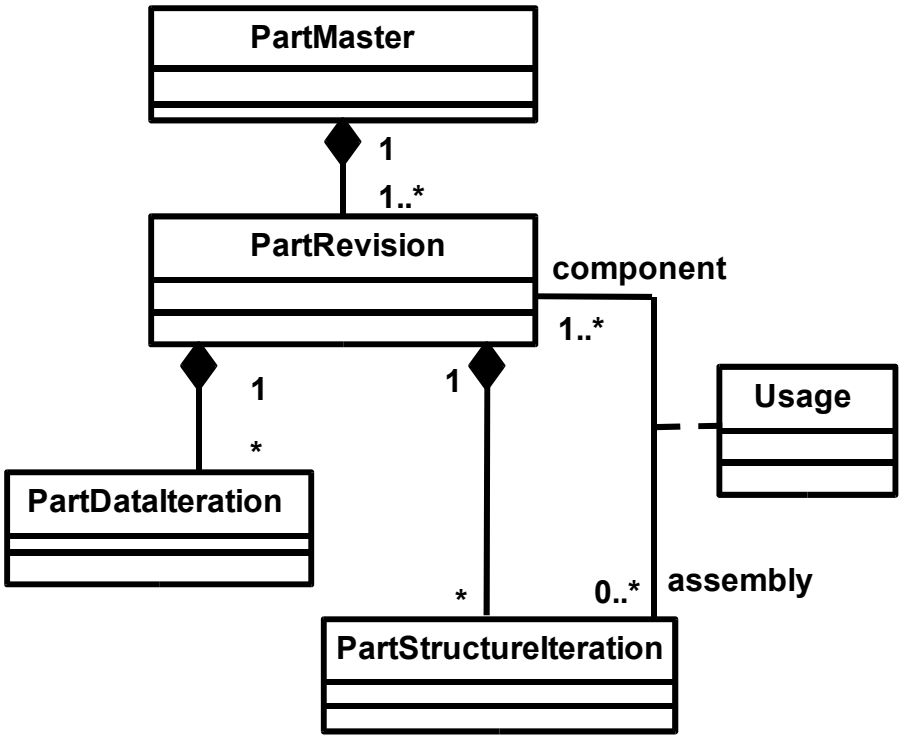
Factory/Finder Structure for Extension



PDM Part Structure Pattern

- PartMaster – unchanging attributes
- PartRevision – controlled changes
- PartDataIteration – atomic part data and work-in-progress
- PartStructureIteration – complex part (assembly) and work-in-progress
- Usage – role played by part in assembly

Applied to all artifacts



Implementation of Release 1

- Goal
  - Support of simple UML class model storage
  - Support AOM composition research
- Approach
  - XMI representation of UML data
  - XML / XML schema representation of other data
  - Use open source components – MySQL, ArgoUML
  - Code in Java as monolithic application
  - Simple GUI to launch

Implementation of Release 1

- Approach (cont.)
  - Using RM-ODP to structure design process
  - Refactoring of student project code (Nathan)
  - Use CVS to control code source

Some Future Work Directions

- Expand artifact types
  - Other UML models, interaction, activity, state, etc.
  - Java code and code analysis
- Increase Repository Robustness
  - N-tiered and distributed
- Increase tool support
  - AOM composition tools
  - Model analysis tools
  - Code analysis