

# ***Semantic Annotation of Computational Components***

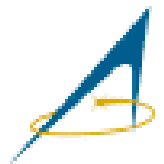
*Peter Vanderbilt*

*Piyush Mehrotra*

**NASA Ames Research Center**



***June 2004  
GGF11 -- Semantic Grid Workshop***



# Overview

- Motivation
- CRADLE Architecture
- Dataspace model
- Plan generation algorithm
- Status & futures
- Relation to RDF



# Motivation

*Users want to be relieved of the responsibility of keeping track of what tools are available and how to use them*

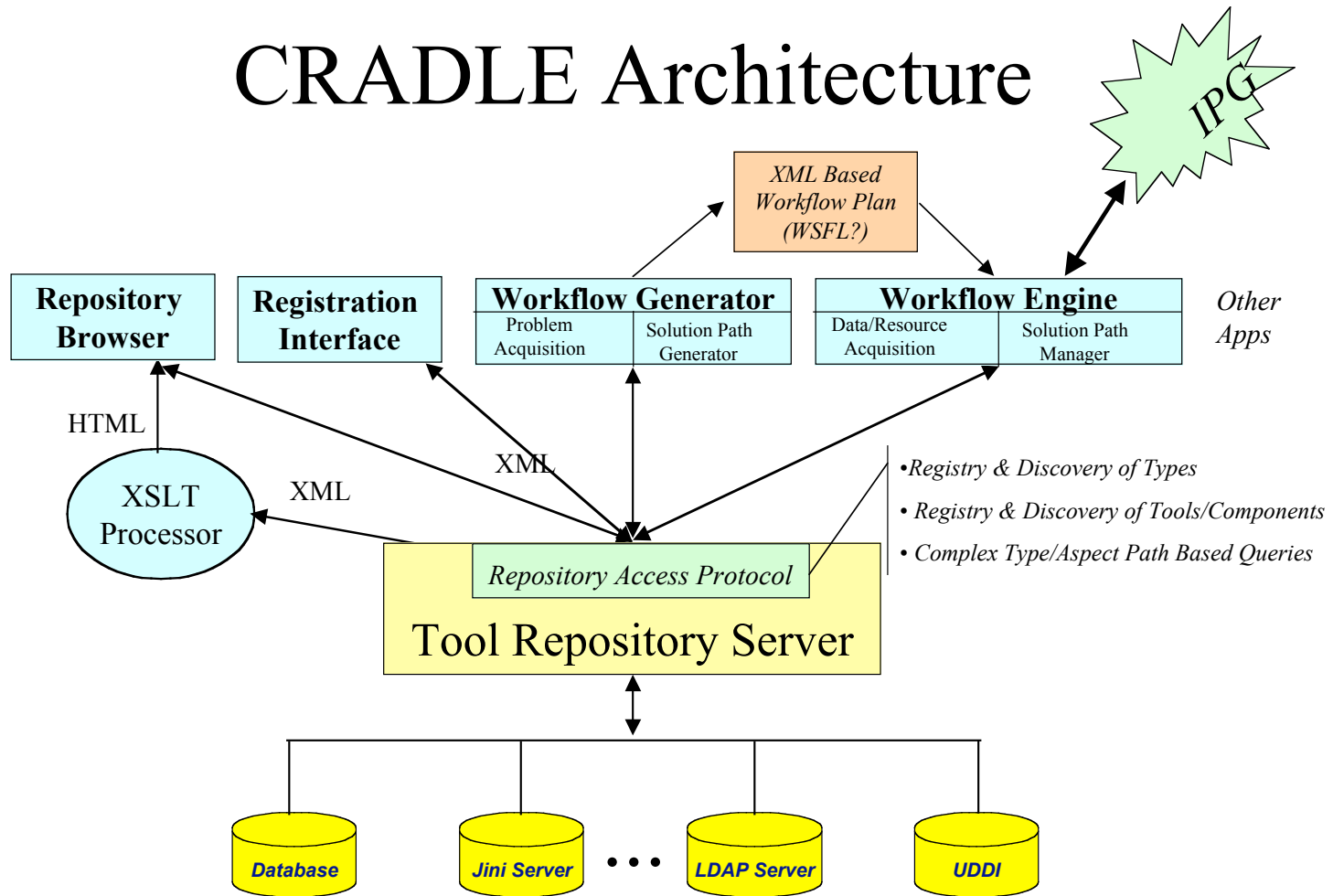
*Goal of the project:*

To provide an infrastructure and the associated mechanisms to register and discover tools & services to support the (semi)-automatic generation of a workflow in order to solve a given user problem.



# Architecture

## CRADLE Architecture



# “Tools”

- Legacy simulation codes
- Pre- and post-processing tools
  - Data translation & extraction tools
- Data sources - UIs, sensors
- Visualization tools



# Use Case

- Tool owners register their tools
- User poses problem:
  - Compute the FEM (triangulated) grid of a 2D cross-section of a wedge
  - Given length, halfspan, nose radius, wedge angle
  - (Using tools with certain fidelity)
- Plan is generated & run



# Issues

- Issues
  - Representation of problem
  - Description of tools
  - Rules for composing tools
- Ideal: Support *automatic* plan generation
  - Input & output names significant
  - Need composite names (not flat)
  - Type is significant



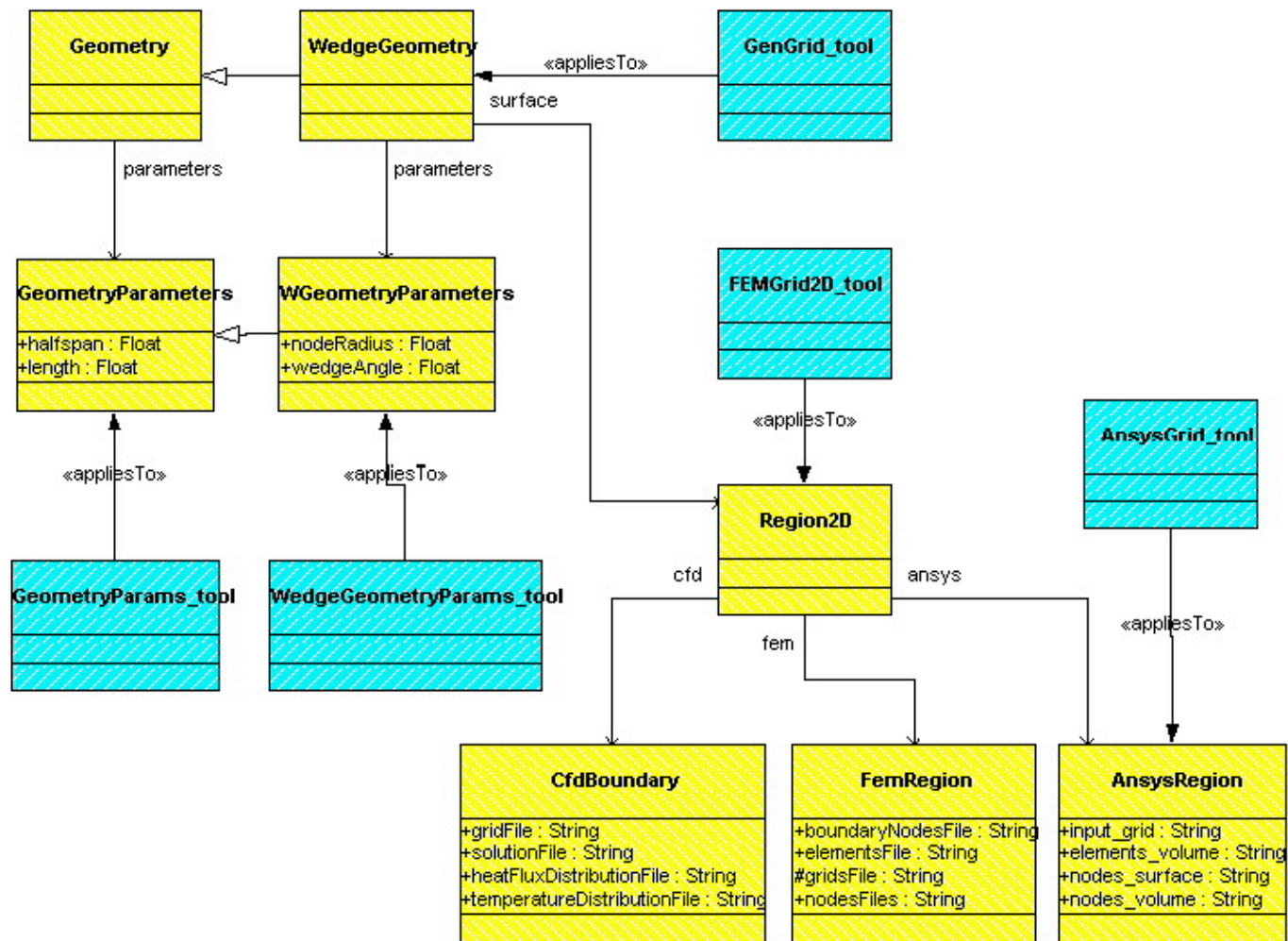
# *Dataspace Model*

- Dataspace: tree of "slots"
  - About some scientific entity
  - A slot may contain a value of an "aspect"
  - Named by aspect names / paths
- Dataspace type
  - Vocabulary of aspect names
  - Types & interpretation of aspects
  - Interdependencies/invariants
  - Inheritance (multiple)
- Tool "appliesTo" dataspace type
  - Inputs & outputs as aspect paths



# Example Types & Tools

Simple Problem Mapping  
8/15/01



# Plan Generation Algorithm

- Params: *problemType, goals, givens*
- While *subgoal in needed - produced*
  - Find tools
    - applying to *problemType* (or supertype) &&
    - yielding *subgoal* &&
    - with OK attributes
  - Select tool (by callout)
  - *produced* += outputs; *needed* += inputs
- Keeps track of dependencies
- Also considers subdataspaces



# *Status & Future Work*

- **Current Status:**
  - XML Schemas for types & tools
  - Client/server repository based on mySQL
  - Simple Perl & Java based plan generators
  - But limited use
- **Futures**
  - Richer data model & parameterized types
  - Apply to web services & data collections
  - Look at using RDF technology



# *Relation to RDF*

- Ontology as dataspaces type
  - Tool "appliesTo" RDF class; inputs & outputs are property paths
- KB for type & tool descriptions?
  - Can KB technology do it?
  - Is it efficient enough?
  - Is it the "right" model?
- Dataspace as a KB?
  - Dataspace is logical, not necessarily instantiated
    - script, interconnected components, workflow
  - Don't need KB technology



# Summary

- CRADLE Architecture
  - Motivation: Simplify use of tools
  - Dataspace model & "appliesTo"
  - Plan generation
  - Status, futures & relation to RDF

