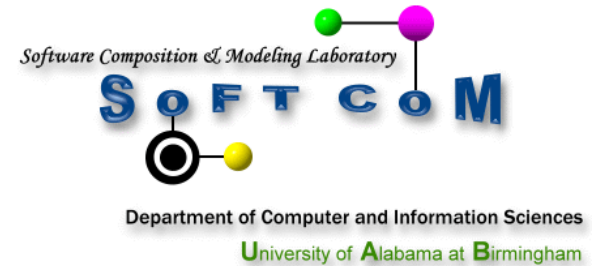


Concern Separation in Model-Integrated Computing

Jeff Gray and Aniruddha Gokhale

Software Composition and Modeling Laboratory
University of Alabama at Birmingham



Institute for Software Integrated Systems (ISIS)
Vanderbilt University



*OMG's First Annual
Model-Integrated Computing Workshop*

Arlington, VA
October 12-15, 2004

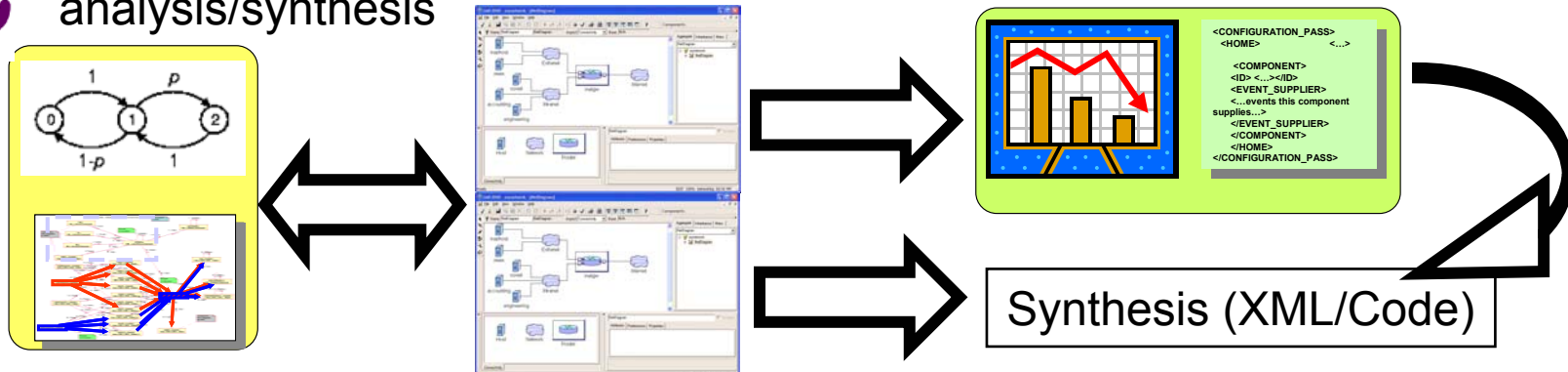


Supported by the DARPA PCES program
DARPA/AFRL Contract # F33615-03-C-4112

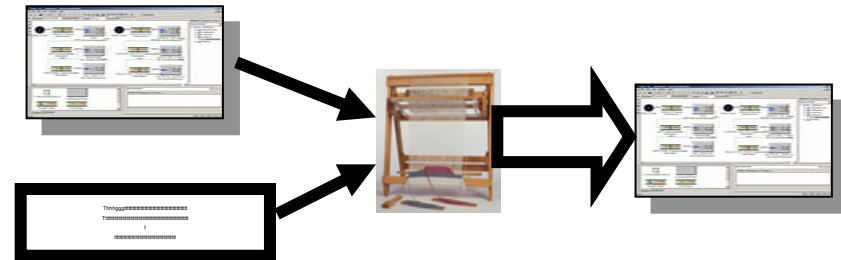
Focus Areas of this Presentation

CoSMIC

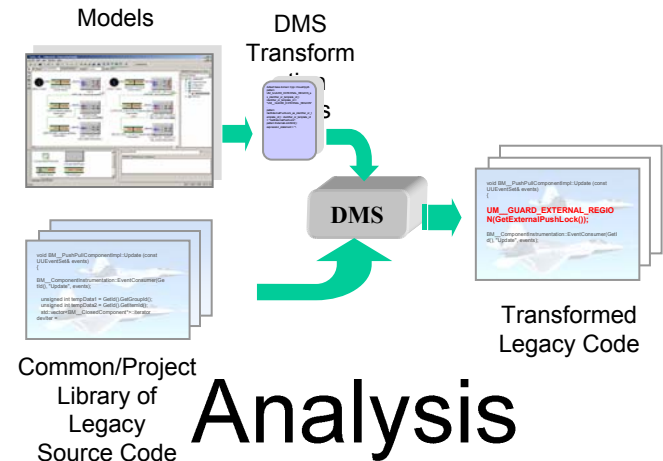
-- a suite of domain-specific modeling languages and tools for DRE analysis/synthesis



C-SAW – a model transformation tool for separating crosscutting properties in domain-specific models



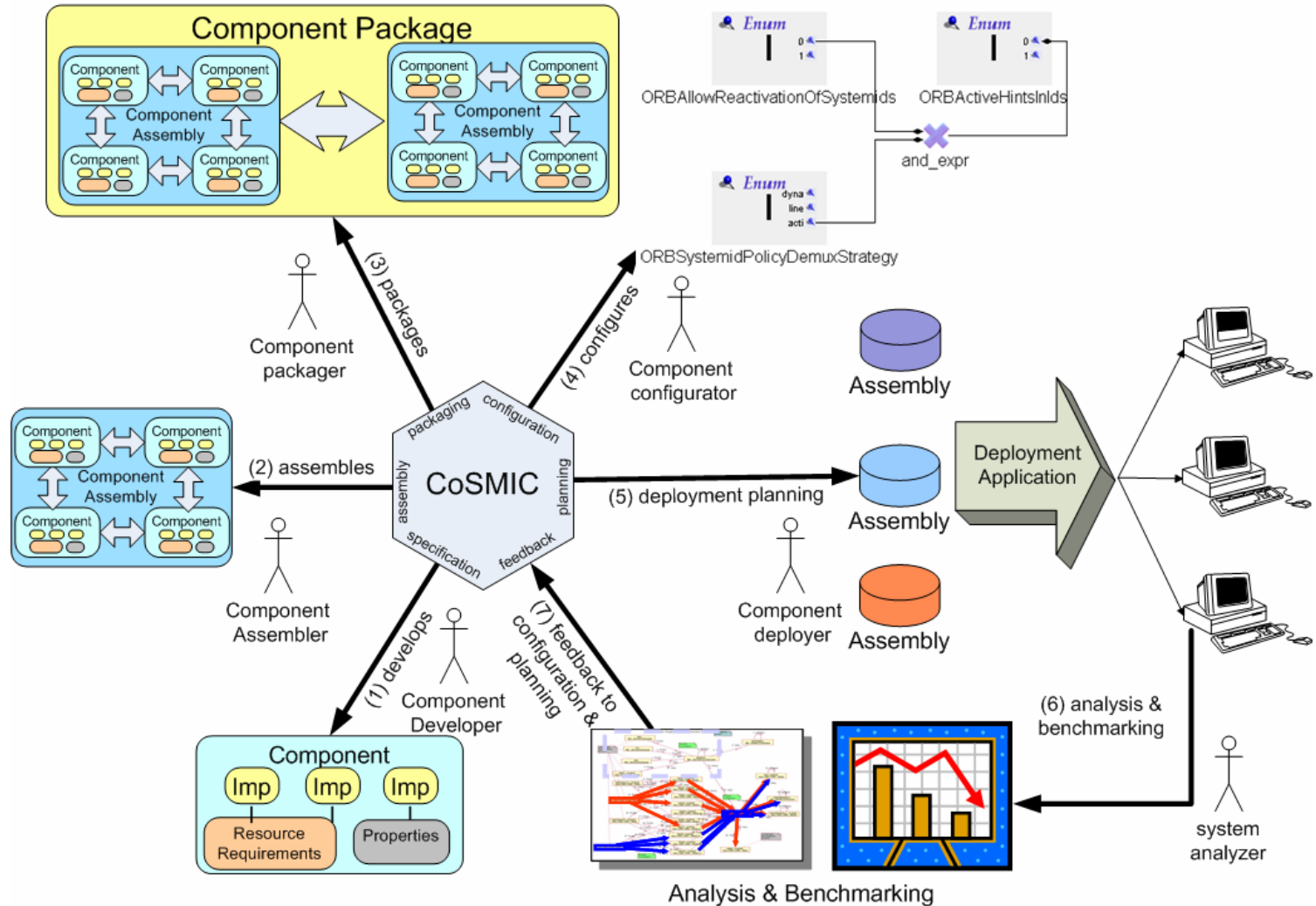
- ◆ **Goal: Maintain the fidelity between the evolving model properties and the legacy source code**
- ◆ **Challenges: Parsing and invasively transforming legacy source code from higher-level models**
- ◆ **Solution: Model-Driven Program Transformation**
 - Based on the unification of a mature program transformation system with a meta-modeling environment



CoSMIC:
Modeling Deployment & Configuration
Crosscutting Concerns

Model-Driven Middleware for DRE Systems

CoSMIC Model Driven Middleware Suite



- ◆ Addresses DRE systems configuration and deployment crosscutting concerns
- ◆ Employs MIC technology
- ◆ www.dre.vanderbilt.edu/cosmic

Addressing D&C Crosscutting Concerns with DAnCE

• Different Stages

• *Development*

- Developer
- Assembler
- Packager

• *Target*

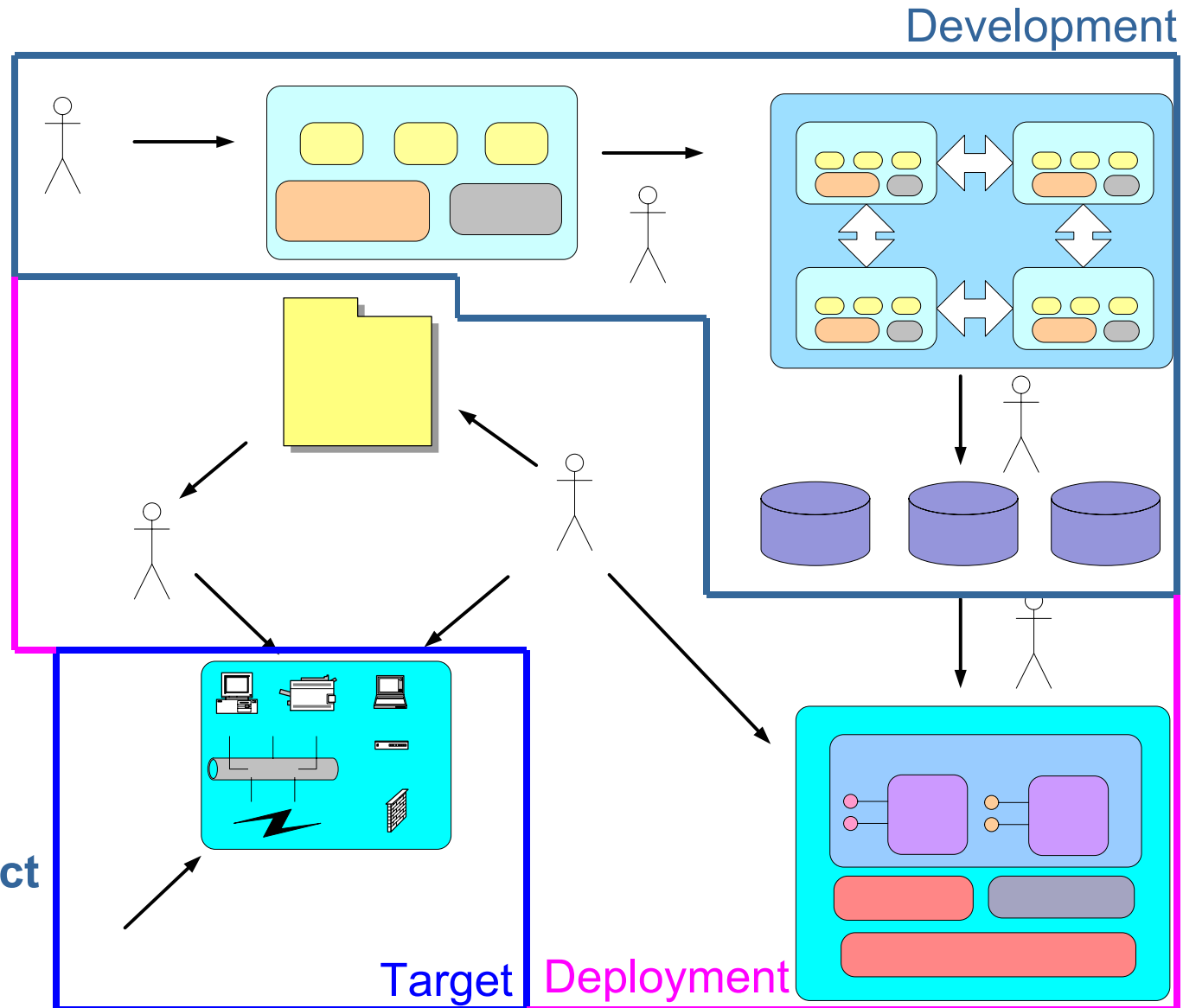
- Domain Administrator

• *Deployment*

- Repository Administrator
- Planner
- Executor

• **Actors are abstract**

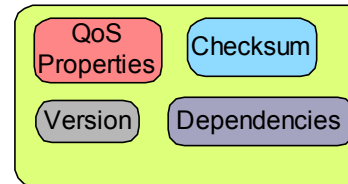
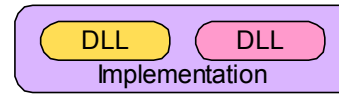
- Usually human + software tool



PICML: Capturing & Modeling D&C Crosscutting Concerns

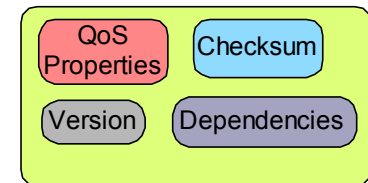
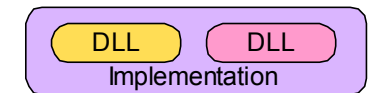
- **Context**

- Configuring & Deploying component-based applications using XML meta-data



- **Problem**

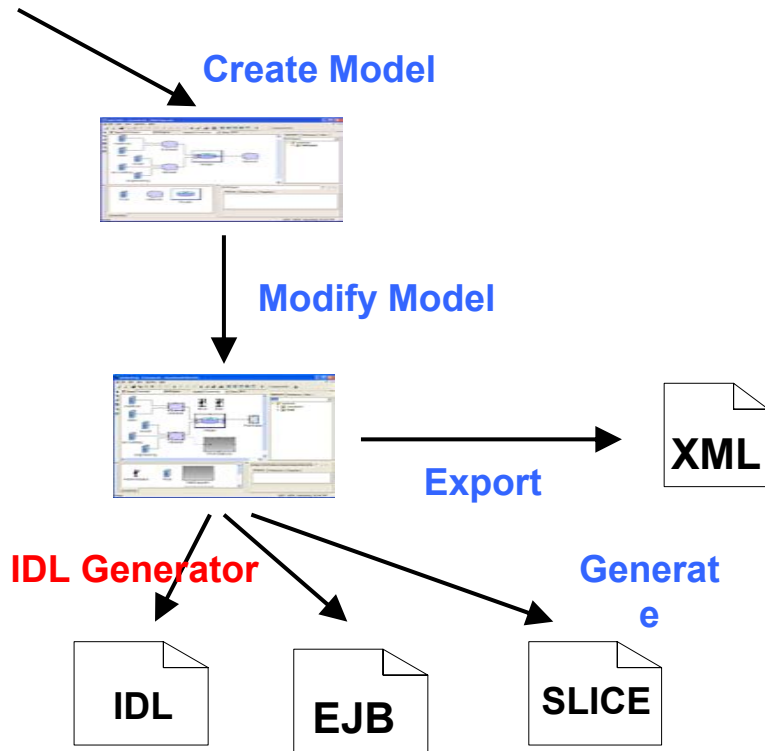
- Meta-data split across multiple XML descriptors
- Inter-dependencies between descriptors
- XML is error-prone to read/write manually
- No guarantees about semantic validity (only syntactic validation possible)
- If meta-data is wrong, what about my application?



- **Solution**

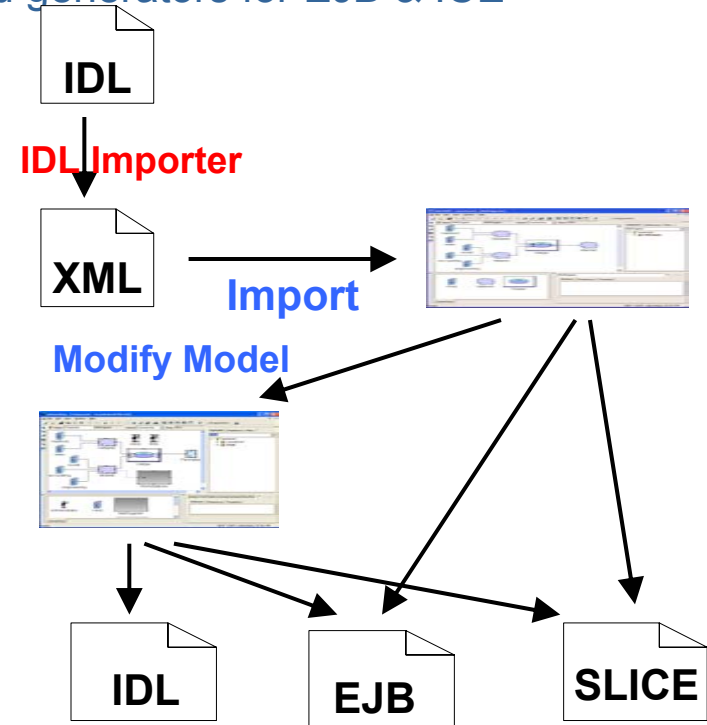
- PICML = Platform Independent Component Modeling Language
 - Modeling paradigm developed using Generic Modeling Environment (GME)
- Capture dependencies visually
- Define semantic constraints using Object Constraint Language (OCL)
- Generate domain specific meta-data from models
- Correct-by-construction

IDML: Capturing Interface Definition Aspects in PICML



- IDML = Interface Definition Modeling Language
- Graphical modeling language.
- Component middleware building blocks.
- Integrated with PICML.
- Export model to equivalent XML format.
- Generate middleware-specific application code.
 - IDL generator finished
 - Planned generators for EJB & ICE

- IDL Importer translates IDL into IDML's XML format.
- Import XML into graphical modeling tool.
 - Translate to other middleware platform.
- Develop model further
 - Regenerate IDL.
 - Generate application code for a different middleware platform.



EQAL: Capturing Event QoS Aspects in PICML

• Context

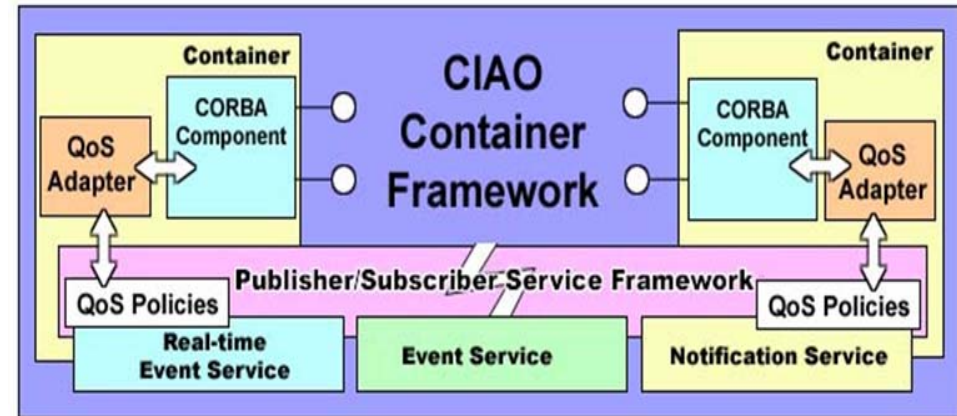
- Publisher/subscriber services are highly configurable
- XML-based specification of QoS properties

• Problems

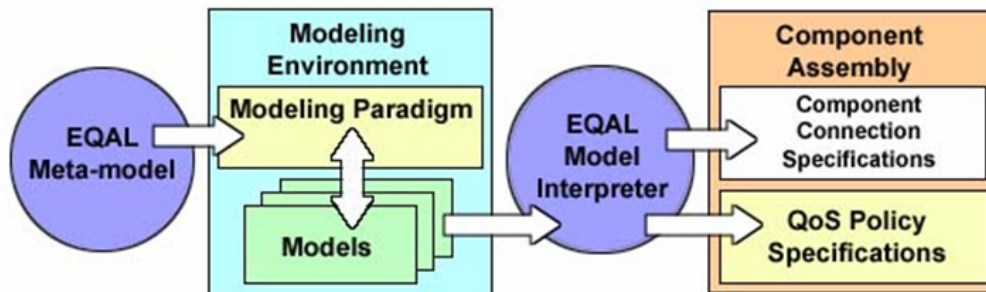
- Multiple dissimilar services
- Semantically invalid operating policies
- Error-prone handwritten XML

• Solution

- Use models to enforce policy constraints & synthesize configuration files



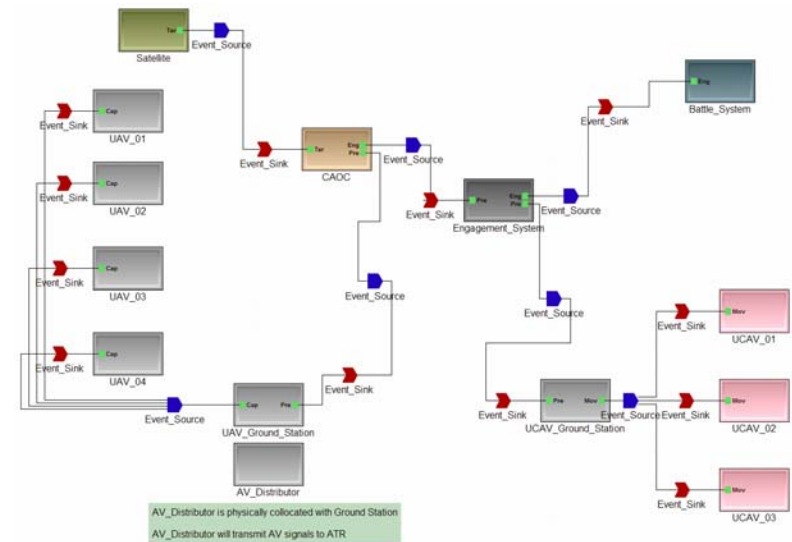
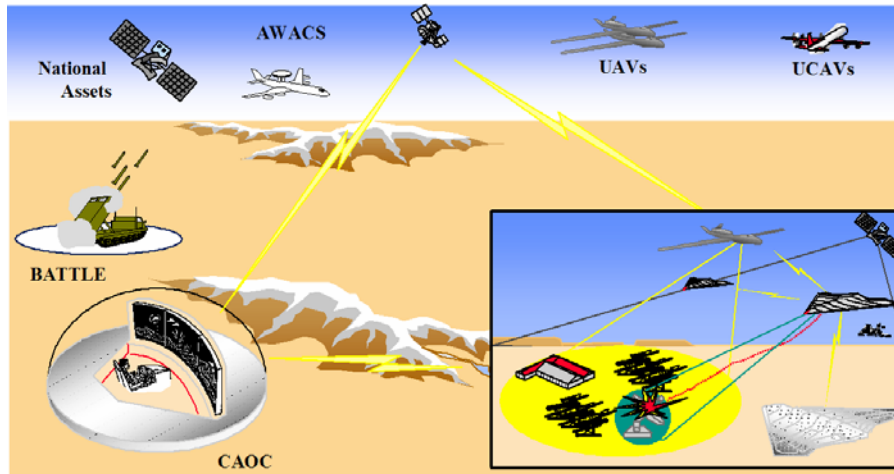
- EQAL = Event QoS Aspect Language
- EQAL is part of PICML within the CoSMIC suite
 - Built in the Generic Modeling Environment (GME)
 - Addresses publisher/subscriber service configuration and deployment challenges
 - *Models* specify service configurations and deployments
 - *Aspects* decouple D&C concerns
 - *Constraints* ensure semantic validity
 - *Interpreters* generate descriptor files



C-SAW: An Aspect Model Weaver

*Separating Crosscutting Concerns from
Domain-Specific Models*

Scaling up to Large DRE Systems

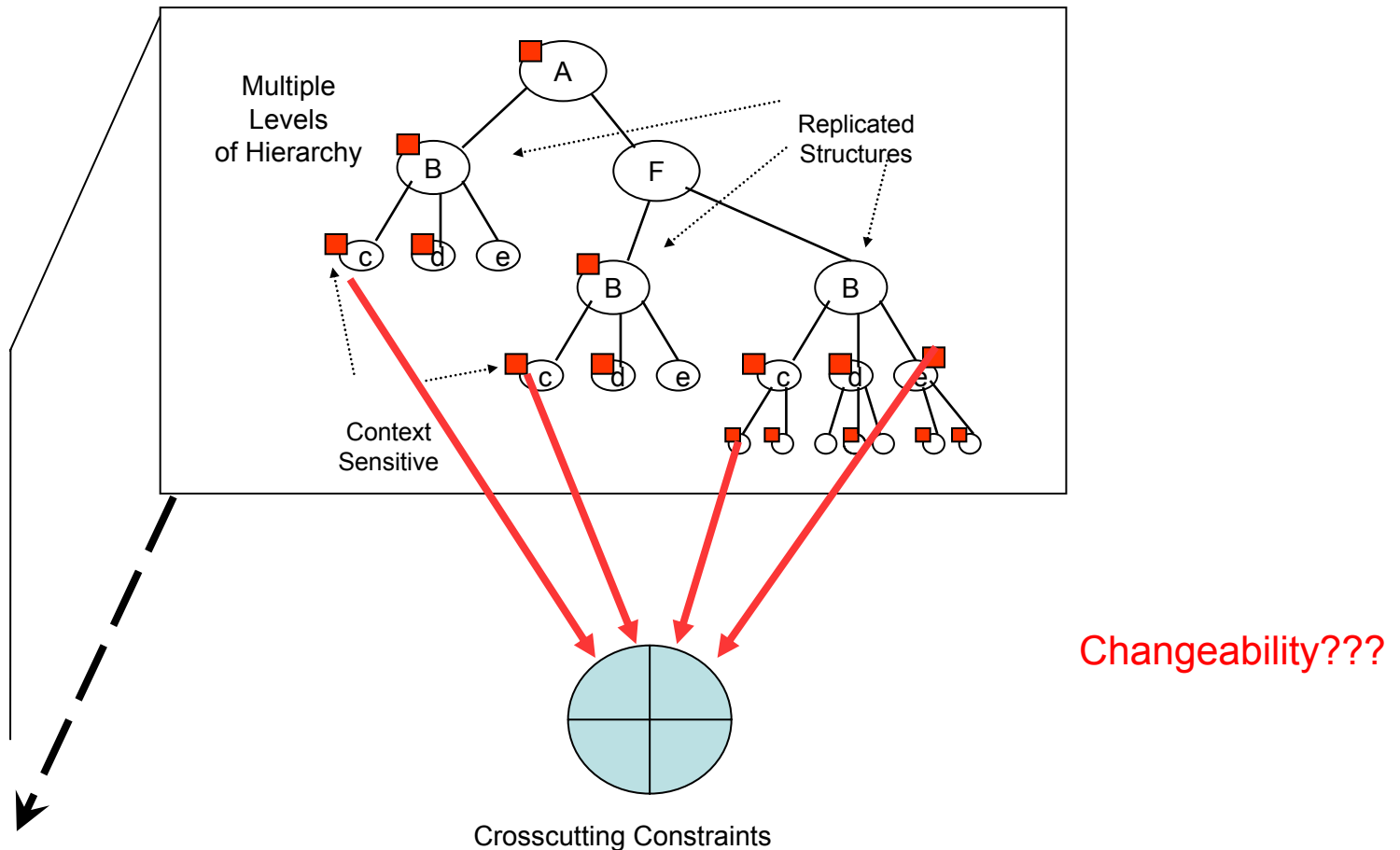


- ◆ Rich and complex interactions among modeling elements
- ◆ Changing requirements have cascading effect across multiple locations in a model
- ◆ The time to make such changes becomes infeasible to do manually; error prone nature of manual change can lead to incorrect models
- ◆ Example: Scaling a model from 3 UAVs to 30 UAVs involves a combinatorial amount of changes that becomes nearly impossible to model by hand; similar for large federations of event channels (EQAL)

Challenge: Crosscutting Constraints in Real-Time/Embedded Models

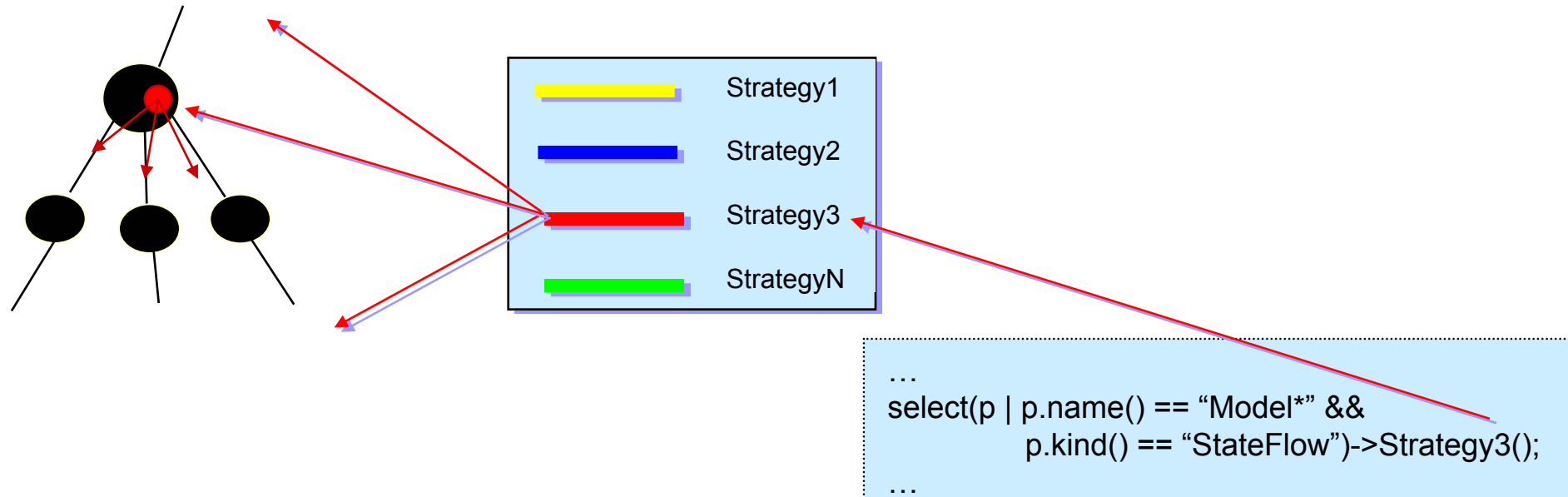
- ◆ Base models become constrained to capture a particular design

- ◆ Concerns that are related to some global property are dispersed across the model



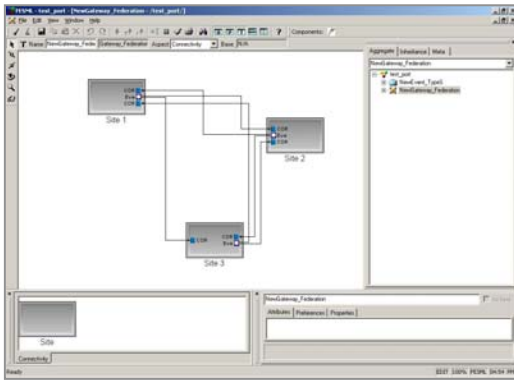
Quantification Over a Domain Model

- ◆ Apply AO Weaving concepts to Model-based systems
 - Weavers ‘Decorate’ Models with attributes & constraints
 - Weavers compose new model constructs



Using C-SAW for Aspect Weaving within EQAL Models

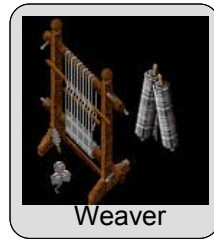
EQAL Model with 3 sites



ECL Specifications

```
aspect Start()
{
  scaleUpSites(1, 8, 4);
  strategy scaleUpSites(site_id, max, idx: integer)
  {
    iterateSite_r(idx-1, 1, max, idx);
    addSite_r(site_id, max, idx);
    addCon_r1(site_id, max, 1, 1, idx);
  }
  strategy iterateSite_r(oldmax, oldidx,
    max, idx : integer)
  {
    declare id_str:string;
    if (oldidx <= oldmax) then
      id_str := intToString(oldidx);
      rootFolder().findModel("NewGateway_Federation").
        findModel("Site " + id_str).addGateway_r(max, idx);
      iterateSite_r(oldmax, oldidx+1, max, idx);
    endif;
  }
}
```

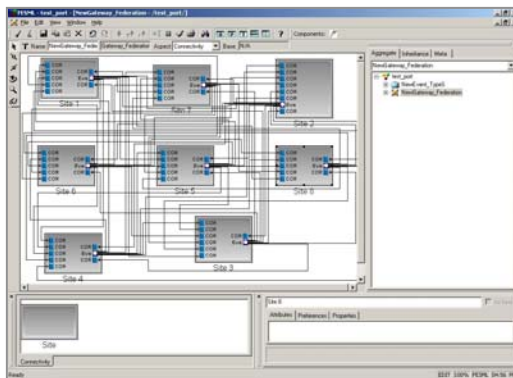
1



Weaver



3



EQAL Model with 8 sites

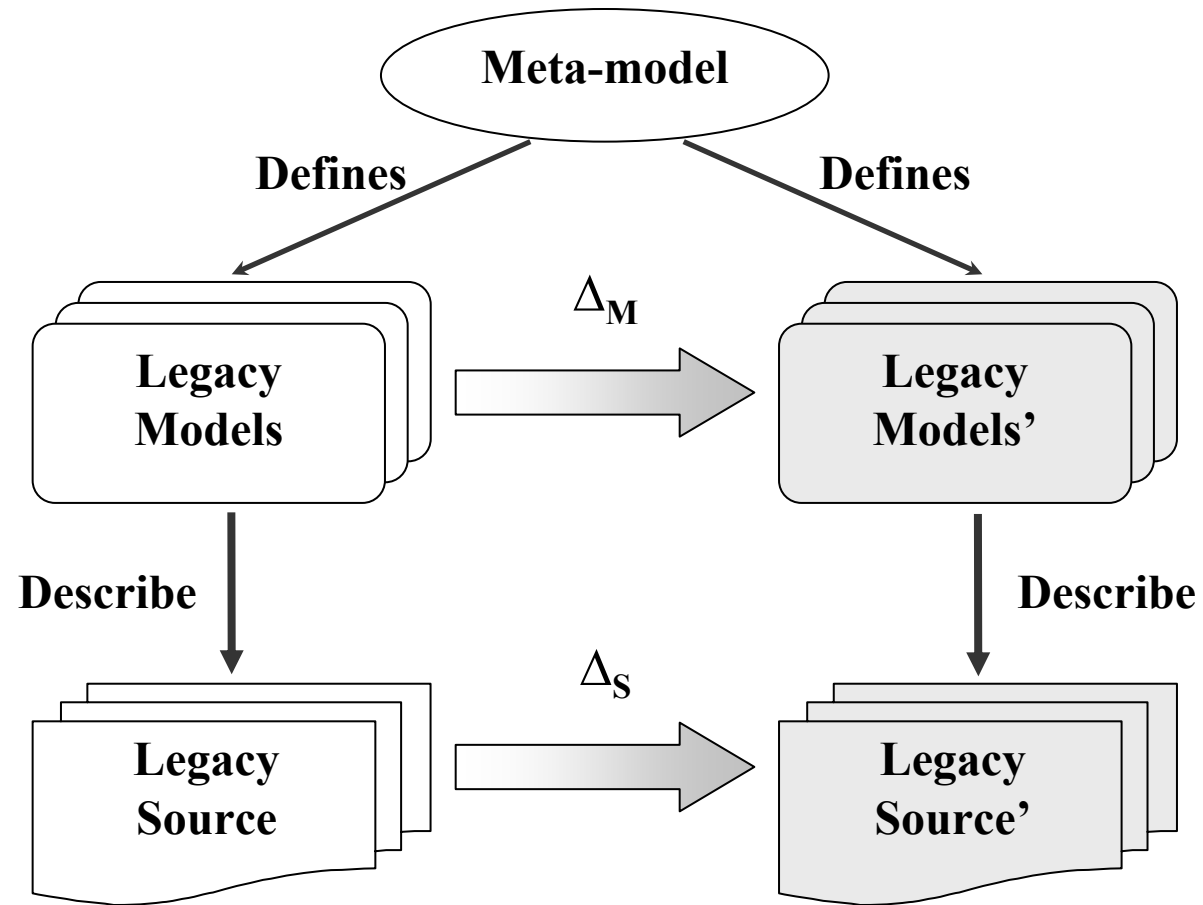
1. EQAL is used to model a federated event service with three sites
2. The ECL strategy specifications are used to scale up any site as well as the corresponding connections in the EQAL model. Three steps are included:
 - Add extra CORBA_Gateways to the existing sites
 - Repeatedly replicate the site as an instance
 - Create connections between all of the sites
3. C-SAW takes the original EQAL model and the ECL specifications, and then generates the new scaled-up EQAL model with additional sites:
 - Model weaving to explore design alternatives more rapidly
 - Design decisions crosscut model hierarchy
 - Removes manual error resulting from tedious/repetitious changes

Model-Driven Program Transformation

*Ensuring a Causal Connection Between
Concerns at Different Abstraction Levels*

Evolution of Models and Legacy Source Code

- ◆ Goal: Maintain the fidelity between the evolving model properties and the legacy source code
- ◆ Challenges: Parsing and invasively transforming legacy source code from higher-level models
- ◆ Solution: Model-driven program transformation

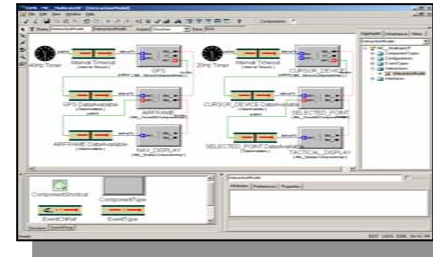


Δ_M : The changes made to the legacy models
 Δ_S : The changes reflected in the legacy source

Model-Driven Program Transformation (MDPT)

```
void BM_PushPullComponentImpl::Update (const UIEventSet& events)
{
    BM_ComponentInstrumentation::EventConsumer(GetId(), "Update", events);
    unsigned int tempData1 = GetId().GetGroupId();
    unsigned int tempData2 = GetId().GetItemId();
    std::vector<BM_ClosedComponent*>::iterator devIter = device_begin();
    std::vector<BM_ClosedComponent*>::iterator devIterEnd = device_end();
    for (; devIter != devIterEnd; ++devIter) {
        BM_ClosedComponent* component = *devIter;
        const UIIdentifier& id = component->GetId();
        if (idInEventSet(id, events))
        {
            const BM_ClosedFunctionalFacet& facet = component->ProvideClosedFunctionalFacet();
            BM_ComponentInstrumentation::SendDirectCall(GetId(), "Update", component->GetId(), "GetData1");
            tempData1 += facet.GetData1();
            BM_ComponentInstrumentation::SendDirectCall(GetId(), "Update", component->GetId(), "GetData2");
            tempData2 += facet.GetData2();
        }
    }
    data1 = tempData1; data2 = tempData2;
}
```

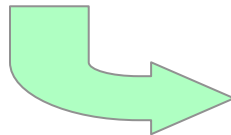
Common/Project Library of Legacy Source Code



Updated models



DMS Transformation

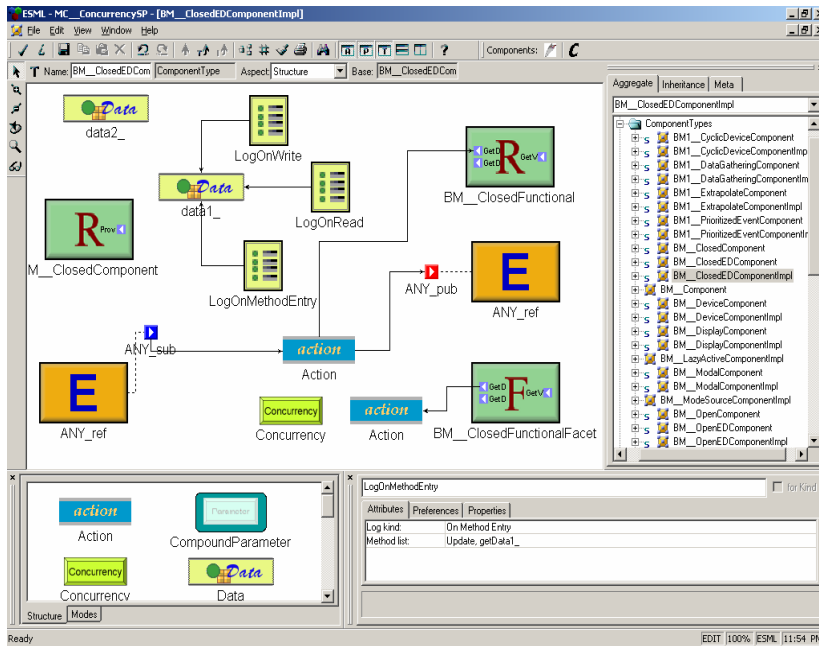


Interpreter

Transformed Legacy Source

```
void BM_PushPullComponentImpl::Update (const UIEventSet& events)
{
    UM_GUARD_EXTERNAL_REGION(GetExternalPushLock());
    BM_ComponentInstrumentation::EventConsumer(GetId(), "Update", events);
    unsigned int tempData1 = GetId().GetGroupId(); unsigned int tempData2 = GetId().GetItemId();
    std::vector<BM_ClosedComponent*>::iterator devIter = device_begin();
    std::vector<BM_ClosedComponent*>::iterator devIterEnd = device_end();
    for (; devIter != devIterEnd; ++devIter) {
        BM_ClosedComponent* component = *devIter; const UIIdentifier& id = component->GetId();
        if (idInEventSet(id, events))
        {
            const BM_ClosedFunctionalFacet& facet = component->ProvideClosedFunctionalFacet();
            BM_ComponentInstrumentation::SendDirectCall(GetId(), "Update", component->GetId(), "GetData1");
            tempData1 += facet.GetData1();
            BM_ComponentInstrumentation::SendDirectCall(GetId(), "Update", component->GetId(), "GetData2");
            tempData2 += facet.GetData2();
        }
    }
    UM_GUARD_INTERNAL_REGION; log.add("data1_" + data1);
    data1 = tempData1; data2 = tempData2; log.add("data2_" + data2);
}
```

Case Study : A black box data recorder



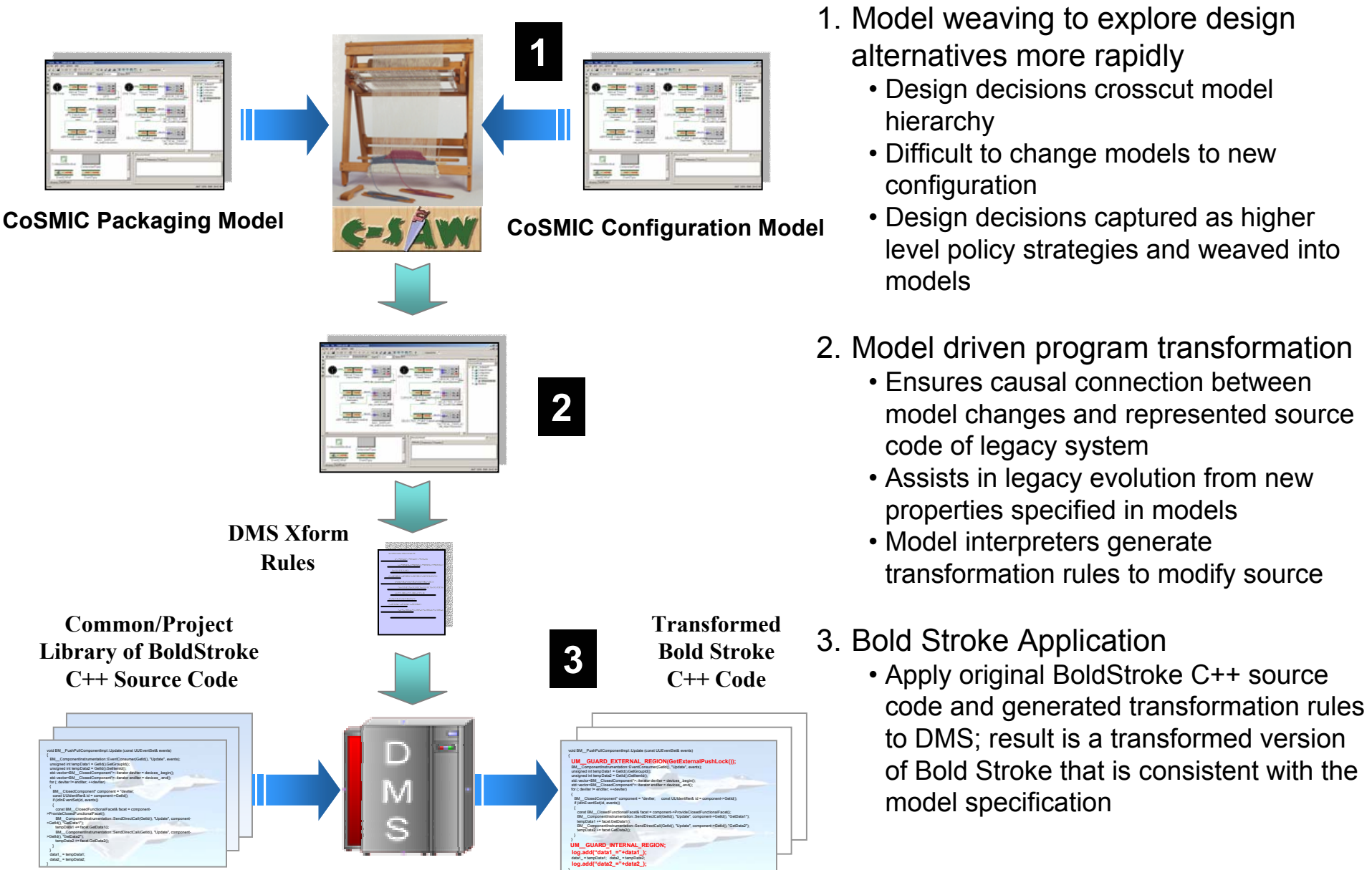
- ◆ Ensures *causal* connection between model changes and the underlying source code of the legacy system
- ◆ Large-scale adaptation across multiple source files that are driven by minimal changes to the model properties
- ◆ Model interpreters generate transformation rules to modify source

```
default base domain Cpp~VisualCpp6.  
pattern LogStmt() : statement = "log.add(\"data1_=\" +  
data1_);".  
pattern LogOnMethodAspect(s:statement_seq):  
statement_seq = "{ \s } \LogStmt(\s)".  
pattern Update(id:identifier): qualified_id = "\id ::  
Update".  
rule log_on_Update(ret:decl_specifier_seq, id:identifier,  
p:parameter_declaration_clause, s: statement_seq):  
function_definition -> function_definition  
= "\ret \Update(\id) (\p) { \s }" -> "\ret  
\Update(\id) (\p) { \LogOnMethodAspect(\s) }"  
if ~[modsList:statement_seq .s matches  
"\:statement_seq \LogOnMethodAspect(\modsList)"].  
rule log_on_Update_cv(ret:decl_specifier_seq,  
id:identifier, p:parameter_declaration_clause, s:  
statement_seq, cv: cv_qualifier_seq): function_definition  
-> function_definition  
= "\ret \Update(\id) (\p) \cv { \s }" -> "\ret  
\Update(\id) (\p) \cv { \LogOnMethodAspect(\s) }" if  
~[modsList:statement_seq .s matches "\:statement_seq  
\LogOnMethodAspect(\modsList)"].  
pattern getData1(id:identifier): qualified_id = "\id ::  
getData1".  
rule log_on_getData1(ret:decl_specifier_seq,  
id:identifier, p:parameter_declaration_clause, s:  
statement_seq): function_definition -> function_definition  
= "\ret \getData1_\(\id) (\p) { \s }" -> "\ret  
\getData1_\(\id) (\p) { \LogOnMethodAspect(\s) }" if  
~[modsList:statement_seq .s matches "\:statement_seq  
\LogOnMethodAspect(\modsList)"].  
rule log_on_getData1_cv(ret:decl_specifier_seq,  
id:identifier, p:parameter_declaration_clause, s:  
statement_seq, cv: cv_qualifier_seq): function_definition  
-> function_definition  
= "\ret \getData1_\(\id) (\p) \cv { \s }" -> "\ret  
\getData1_\(\id) (\p) \cv { \LogOnMethodAspect(\s) }"  
if ~[modsList:statement_seq .s matches  
"\:statement_seq \LogOnMethodAspect(\modsList)"].  
public ruleset applyrules = { log_on_Update,  
log_on_Update_cv, log_on_getData1_,  
log_on_getData1_cv }.
```

Transformed Code fragment

```
1 unsigned int BM_ClosedEDComponentImpl::getData1_() const
2 {
3     Addlog("data1_" + data1_); ← Log on getData1_() method entry
4
5     UM_GUARD_INTERNAL_REGION;
6     BM_ComponentInstrumentation::ReceiveDirectCall(GetId(), "GetData1");
7
8     Addlog("data1_" + data1_); ← Log on reading data1_
9     return data1_;
10 }
11
12 void BM_ClosedEDComponentImpl::Update (const UIEventSet& events)
13 {
14     Addlog("data1_" + data1_); ← Log on Update() method entry
15
16     UM_GUARD_EXTERNAL_REGION(GetExternalPushlock());
17     BM_ComponentInstrumentation::EventConsumer(GetId(), "Update", events);
18     unsigned int tempData1 = GetId().GetGroupId();
19     unsigned int tempData2 = GetId().GetItemId();
20
21     /** REMOVED: code for implementing Real-time Event Channel
22     Addlog("data1_" + data1_); ← Log on writing data1_
23
24     data1_ = tempData1; /** REMOVED: actual variable names (proprietary)
25     data2_ = tempData2;
26 }
```

Two-Level Aspect Weaving



Video Demonstration

Project Web Pages

CoSMIC Modeling Languages and Tools

<http://www.dre.vanderbilt.edu/cosmic>

CoSMIC

C-SAW Aspect Model Weaver

<http://www.gray-area.org/Research/C-SAW/>
Contains papers, downloads, video demos



Department of Computer and Information Sciences

University of Alabama at Birmingham

Backup Slides

Generalization of the control flow for the MDPT process

