

Evaluation of Aspects in UML Models

Phillip Schmidt, Ph.D.
The Aerospace Corporation
Phillip.P.Schmidt@aero.org

Robert Duvall, Ph.D.
The Aerospace Corporation
Robert.E.Duvall@aero.org

Jeffrey Lankford
The Aerospace Corporation
Jeffrey.P.Lankford@aero.org

Greg Mulert
The Aerospace Corporation
Gregory.L.Mulert.@aero.org

Agenda

- Introduction
- REACT
- Aspect-Oriented Architectural Analysis
- Architectural Aspect Types
- REACT's AOAA Approach
- Example
- AOAA Benefits
- Future Plans

Introduction

- Deployment schedules and costs drive need for **evolvable software designs**.
 - ◆ New environments
 - ◆ New services
 - ◆ New contexts
- Systemic **concerns** often cut across OO decomposition boundaries.
- **Aspect-oriented programming** techniques attempt to identify and manage crosscutting concerns.
 - ◆ Helpful for fixed, known concerns
 - ◆ Limited when concerns vary or conflict over time
- Early **architectural development** exhibits similar cross-cutting concerns

Early Architectural Risks

- Weak architectural tools
- Unconventional UML usage
- Incomplete, inconsistent UML expression
- Ambiguous interpretation of design intent
- Ascertaining derived architectural information
- Little or no performance assessment insight
- Managing evolving architectural designs

Real-time Embedded Architecture-Centric Testbed (REACT)

- An **architecture-centric**, “**early discovery**” testbed capable of analyzing and modeling architectural designs prior to code development.
- Receives contractor-provided architecture artifacts principally expressed in Unified Modeling Language (UML)
- Automatically extracts architectural information
- Analyzes architectural representations for consistency/completeness
- Collaborate closely with contractor on constructive recommendations
- Creates executable models
- Conducts architectural assessments to
 - ◆ Understand logical execution behavior of architecture
 - ◆ Focus on critical execution paths
 - ◆ Address cross-cutting architectural concerns
- Work closely with UML vendors to address tool deficiencies
- Reverse engineers “as-built” architectures

Aspect-Oriented Architectural Analysis

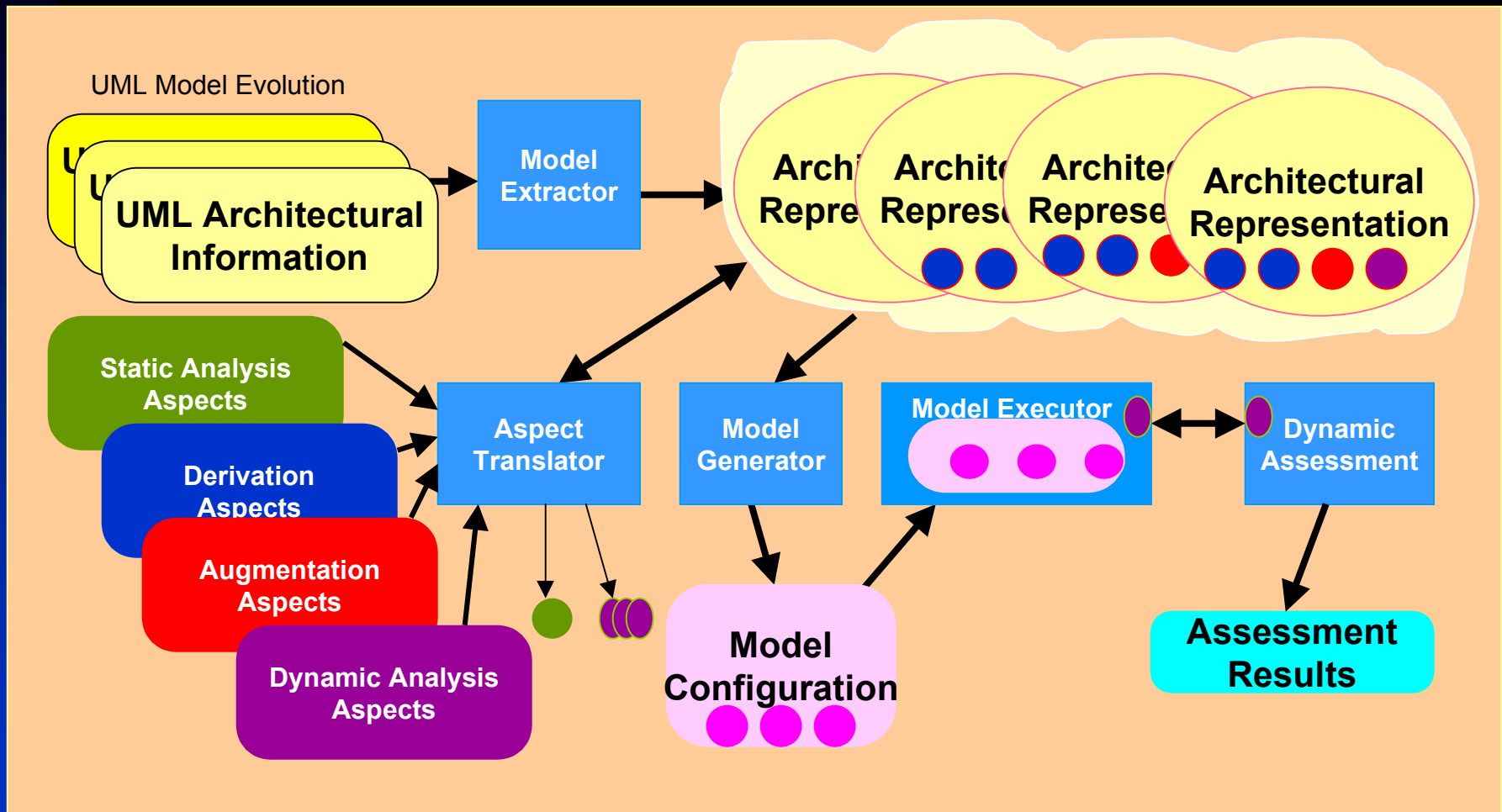
- Idea: **Apply aspects over UML architectural domain**

AOP	AOAA
Leverage expression of cross-cutting concerns	Leverage expression of cross-cutting concerns
Programming language domain (e.g. Java)	Architectural domain (e.g. UML and other artifacts)
Solutions architecturally intrusive (completeness)	Architecturally non-intrusive; separable via simulation
Address dynamic, execution impacts	Address static or dynamic aspects

Architectural Aspect Types

Aspect Type	Description	Example
Static Analysis Aspects	Perform integrity, consistency checks over UML space	Find all examples of destroy object usages
Derivation Aspects	Derive new or customized architectural information from UML space	Collect all event related information
Augmentation Aspects	Add new architectural informational detail	Supply model information based on ICDs, other analysis
Dynamic Assessment Aspects	Define cross-cutting concerns that need to be monitored	Log all raised exceptions

REACT's Aspect-Oriented Architectural Assessment



REACT UML Extraction

(Sample portion)

XML Output

```
- <Classes>
- <Class class.isAugmentation="false" id="C2" name="CommunicationInterfaceClass">
  <ClassTags />
  <Classattrs />
- <Operations>
  <Operation id="OP1" name="GetMessage" />
</Operations>
</Class>
</Classes>
</Types>
+ <Statemachines>
- <Sequencediagrams>
- <Sequencediagram id="SEQD1" name="GetMessage">
- <Participants>
- <Participant id="_0.0-2428.6112..148_"
  instanceofname="Logical_View::Applications::CommunicationInterface::CommunicationInterfaceClass"
  name="">
  <ParticipantGeometry height="45" left="397" top="594" width="221" />
- <Lifeline lifeline.id="_0.0-2448.6228..148_">
  + <LifelineGeometry>
  + <LifelineActivations>
  + <LifelineBranchPoints>
  </Lifeline>
</Participant>
+ <Participant id="_0.0-2318.5839..148_XX1" name="AppsMain">
+ <Participant id="_0.0-2496.6422..148_" name="APPS_MGR_INQ">
</Participants>
- <Messages>
+ <Message comm="synchronous" desid="_0.0-2428.6112..148_" guard="" id="_0.0-2600.6935..148_" name="AddToInputQ" srcid="_0.0-2428.6112..148_" stereotype="">
+ <Message comm="synchronous" desid="_0.0-2428.6112..148_" guard="while(Received Message)" id="_0.0-2563.6782..148_" name="" srcid="_0.0-2428.6112..148_" stereotype="">
+ <Message comm="return" desid="_0.0-2318.5839..148_" guard="Wait State = Wait" id="_0.0-2622.7025..148_" name="" srcid="_0.0-2428.6112..148_" stereotype="">
+ <Message comm="return" desid="_0.0-2318.5839..148_" guard="Normal Priority Message" id="_0.0-2755.7681..148_" name="" srcid="_0.0-2428.6112..148_" stereotype="">
```

Classes,
Statemachine,
Sequence
Diagram details

Aspect Definition Example (Static Analysis)

Architecture to Inspect

```
<?xml version="1.0" ?>
- <RModelAspects>
  <SourceFile name="Demo2_react.xml" />
  <ResultFile name="out.xml" />
  - <AspectSet>
    - <Aspect name="Destructors!">
      - <Query name="//LifelineObjectDestructor">
        <QAction name="printpath" filter="Package Sequencediagram Participant" />
      </Query>
    </Aspect>
    - <Aspect name="Methods!">
      - <Query name="//Coperation">
        <QAction name="printpath" filter="Package Class Coperation" />
      </Query>
    </Aspect>
  </AspectSet>
</RModelAspects>
```

Query the UML space for object destructor usage

Find methods

Collect outputs of interest

Results

```
<?xml version="1.0" ?>
- <RModelAspects>
  <SourceFile name="Demo2_react.xml" />
  <ResultFile name="out.xml" />
- <AspectSet>
  - <Aspect name="Destructors!">
    - <Query name="//LifelineObjectDestructor">
      - <QAction filter="Package Sequencediagram Participant" name="printpath">
        - <Result>
          <Path>Logical_View:Applications:CommunicationInterface:GetMessage:AppsMain</Path>
          <Path>Logical_View:Applications:CommunicationInterface:GetMessage:APPS_MGR_INQ</Path>
        </Result>
      </QAction>
    </Query>
  </Aspect>
  - <Aspect name="Methods!">
    - <Query name="//Coperation">
      - <QAction filter="Package Class Coperation" name="printpath">
        - <Result>
          <Path>Logical_View:Applications:CommunicationInterface:CommunicationInterfaceClass:GetMessage</Path>
        </Result>
      </QAction>
    </Query>
  </Aspect>
</AspectSet>
</RModelAspects>
```

Benefits of AOAA in REACT

- Enables **early discovery** of design concerns
- Performs **automated extraction** of UML architectural information
- Moves **aspect-oriented analysis** into architectural design space
- Applies **auto-derivation and augmentation** as contractor designs evolve
- Permits **customized contractor-driven UML usage**
- Supports **dynamic assessment of concerns**
- Supports **model inspection** of reverse-engineered designs

Future Plans

- Translate manual findings into static aspects
- Develop derivation aspect collections
- Improve augmentation aspects
- Improve aspect primitives for dynamic assessment
- Investigate architectural refactoring techniques

Backup Slides

Aspect Definition (Programming Approach)

```
aspect MyAspect {  
    Log log = new Log();  
    pointcut mypoint (): calls(public  
        *.*(..));  
    after () throwing (Error e):  
        publicInterface () {  
        log.write(e);  
        }  
    }  
}
```

Aspects can
define/create data

Pay attention to
calls of any public
methods

If an error
occurred on the
call log it!

