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

<table>
<thead>
<tr>
<th>AOP</th>
<th>AOAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leverage expression of cross-cutting concerns</td>
<td>Leverage expression of cross-cutting concerns</td>
</tr>
<tr>
<td>Programming language domain (e.g. Java)</td>
<td>Architectural domain (e.g. UML and other artifacts)</td>
</tr>
<tr>
<td>Solutions architecturally intrusive (completeness)</td>
<td>Architecturally non-intrusive; separable via simulation</td>
</tr>
<tr>
<td>Address dynamic, execution impacts</td>
<td>Address static or dynamic aspects</td>
</tr>
</tbody>
</table>
## Architectural Aspect Types

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

- UML Model Evolution
  - UML Architectural Information
    - Static Analysis Aspects
    - Derivation Aspects
    - Augmentation Aspects
    - Dynamic Analysis Aspects
  - Model Extractor
    - Aspect Translator
    - Model Generator
    - Model Executor
    - Dynamic Assessment
    - Assessment Results
  - Architectural Representation
    - Configuration
    - Results

© 2002 The Aerospace Corporation
REACT UML Extraction
(Sample portion)

- <Classes>
  - <Class class="Class" id="C2" name="CommunicationInterfaceClass">
    <ClassTags />
    <ClassAttributes />
  </Class>
  - <Operations>
    - <Operation id="OP1" name="GetMessage" />
  </Operations>
</Classes>
</Types>
+ <Statemachines>
  - <Sequencediagrams>
    - <Sequencediagram id="SEQD1" name="GetMessage">
      - <Participants>
        - <Participant id="_0.0-2428.6112..148" instanceof="Logical View::Applications::CommunicationInterface::CommunicationInterfaceClass" name="" />
        <ParticipantGeometry height="45" left="397" top="594" width="221" />
      </Participants>
      - <Lifeline lifeline.id="_0.0-2446.6228..148">
        + <LifelineGeometry />
        + <LifelineActivations />
      </Lifeline>
    </Sequencediagram>
    + <Participant id="_0.0-2318.5839..148_XX1" name="AppsMain" />
    + <Participant id="_0.0-2496.6422..148" name="APPS_MGR_INQ" />
  </Sequencediagrams>
+ <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="" />
</Messages>
Aspect Definition Example (Static Analysis)

```xml
<?xml version="1.0" ?>
<RMModelAspects>
  <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>
</RMModelAspects>
```
Results

```xml
<?xml version="1.0" ?>
<RMetamodel>
  <SourceFile name="Demo2_react.xml" />
  <resultFile name="out.xml" />
  <AspectSet>
    <Aspect name="Destructors!">
      <Query name="/LifelineObjectDestructor">
        <QAction filter="Package Sequencediagram Participant" name="printpath">
          <Path>Logical_View:Applications:CommunicationInterface:GetMessage:AppsMain</Path>
          <Path>Logical_View:Applications:CommunicationInterface:GetMessage:APPS_MGR_INQ</Path>
        </QAction>
      </Query>
    </Aspect>
    <Aspect name="Methods!">
      <Query name="/Coperation">
        <QAction filter="Package Class Coperation" name="printpath">
          <Path>Logical_View:Applications:CommunicationInterface:CommunicationInterfaceClass:GetMessage</Path>
        </QAction>
      </Query>
    </Aspect>
  </AspectSet>
</RMetamodel>
```
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);
        }
}
REACT Architecture

Integrated Tool Support:
- UML tools (Describe, ROS, others)
- XML-DB Schema
- UC Berkeley Simulator (Ptolemy)
- Xerces XML Parser

Architecture Artifacts
- UML Model Augmentation
- Aspect Definitions
- Model Parameterization

Support Components
- Assessments
- SW Models
- HW Models
- REACT Develop.
- REACT Products
- Augmentations

REACT's Architectural Representation
- Model Executor
- Model Generator
- Model Extractor
- SW Models
- HW Models
- Parameterization Data
- Assessment Data

Integrated Tools
- UML/XMI
- UML/XMI
- UML/XMI

Contractor Input
- REACT Develop.