Towards Architecture-based Self-Healing Systems

Eric M. Dashofy, André van der Hoek, and Richard N. Taylor

WOSS’02
November 18, 2002
What is “self-healing?”

Key Question: What is the difference between a fault-tolerant and a self-healing system?

Fault-Tolerant

- Connotes fault-based repair and understanding
- Faults are likely pre-specified
- Repair strategies are also pre-specified

Self-Healing

- Connotes goal-based repair and understanding
- Unexpected faults are expected
- Arbitrary repair strategies constructed at runtime
Overall Vision

Plan Changes

Deploy Change Descriptions

Evaluate & Monitor Observations

Enact Changes & Collect Observations

Architectural Model

Maintain Consistency

Implementation

Choose repair strategy

Model & enact repair strategy
Our Focus

Plan Changes

Deploy Change Descriptions

Evaluate & Monitor Observations

Enact Changes & Collect Observations

Architectural Model

Maintain Consistency

Model & enact repair strategy

Choose repair strategy
Additional Aspects of the Approach

Architectural Styles

- Loosely-coupled, event-based
- Foundation for runtime change
- Foundation for monitoring

Systems described in extensible ADL

- Description accompanies deployed system
- Repair strategies expressed in terms of architecture description
Expressing Repair Strategies Using Architecture Differencing

Architecture 1
(bound to running system)

Watching

Smart Monitoring Agent
Expressing Repair Strategies Using Architecture Differencing

Architecture 1
(bound to running system)

Smart Monitoring Agent

Malfunction Detected!
Expressing Repair Strategies Using Architecture Differencing

Architecture 1
(bound to running system)
watching
Smart Monitoring Agent
creates description of
Architecture 2

Note: Opportunity for architecture analysis here.
Expressing Repair Strategies Using Architecture Differencing

**Architecture 1**
(bound to running system)

Smart Monitoring Agent

**Architecture 2**

Differencing Engine creates “architectural diff” describing differences between architectures
Expressing Repair Strategies Using Architecture Differencing

Architecture 1
(bound to running system)

Smart Monitoring Agent

watching

Differencing Engine creates “architectural diff” describing differences between architectures

Remove
Add

Comp4
Comp6

Diff 1
Effecting Repairs Using Architectural Diffs

<table>
<thead>
<tr>
<th>Remove</th>
<th>Add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp4</td>
<td>Comp6</td>
</tr>
</tbody>
</table>

Repair Plan 1

Architecture 1

Architecture Evolution Manager

Maintains Consistency

Running System
Effecting Repairs Using Architectural Diffs

<table>
<thead>
<tr>
<th>Remove</th>
<th>Add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp4</td>
<td>Comp6</td>
</tr>
</tbody>
</table>

**Repair Plan 1**

**Architecture 1**

Architecture Merging engine merges architectural diffs into architecture descriptions.

Merging Engine

Architecture Evolution Manager

Running System

Merging Engine merges architectural diffs into architecture descriptions.

Maintains Consistency
Effecting Repairs Using Architectural Diffs

<table>
<thead>
<tr>
<th>Remove</th>
<th>Add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp4</td>
<td>Comp6</td>
</tr>
</tbody>
</table>

**Repair Plan 1**

**Note:** A “what-if” merge can also be done against a copy of the architecture description for validation or analysis.

**Architecture 1**

- Comp1
- Comp2
- Comp3
- Comp4
- Comp5

**Merging Engine**

**Architecture Evolution Manager**

**Running System**

**Maintains Consistency**

**Perform merge**
Effecting Repairs Using Architectural Diffs

<table>
<thead>
<tr>
<th>Remove</th>
<th>Add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp4</td>
<td>Comp6</td>
</tr>
</tbody>
</table>

Repair Plan 1

Architecture 1

Merging Engine

Architecture Evolution Manager

Maintains Consistency

Running System
Effecting Repairs Using Architectural Diffs

<table>
<thead>
<tr>
<th>Remove</th>
<th>Add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp4</td>
<td>Comp6</td>
</tr>
</tbody>
</table>

Repair Plan 1

Architecture 1

Merging Engine

Architecture Evolution Manager

Maintains Consistency

Running System
Effecting Repairs Using Architectural Diffs

### Repair Plan 1

<table>
<thead>
<tr>
<th>Remove</th>
<th>Add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp4</td>
<td>Comp6</td>
</tr>
</tbody>
</table>

### Architecture 1

- **Merging Engine**
- **Architecture Evolution Manager**
- **Comp1**
- **Comp2**
- **Comp3**
- **Comp5**
- **Comp6**

**Running System**

**Maintains Consistency**
Applications Targeted

- **Spacecraft/Spacecraft Ground Systems**
  - Architecture modeling formalism, ideas about dynamism already being adopted by MDS project at JPL

- **Other component-based, event-driven systems**
  - Military command and control

- **Multi-agency systems**
  - Coalition warfare among allied partners with independently developed systems
Future Work/Top Ideas

- **Distributed Dynamism**
  - Making repairs in the face of
    - (Partial) link failure,
    - (Partial) node failure
    - Asymmetric connectivity

- **Are diffs sufficient as repair plans?**
  - Ordering of changes
  - Additional information needed to make changes

- **Approaches to quiescence**
  - Inspired by Kramer & Magee