Software Architecture in Practice

Architectural Reflection

Literature



Main

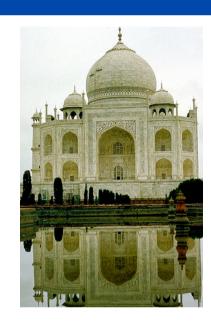
- [Menasce and Kephart, 2007]
 - Menasce, D. and Kephart, J. (2007). Guest Editors' Introduction: Autonomic Computing. *IEEE Internet* Computing, 11(1), pp 18-21
- [Kramer and Magee, 2007]
 - Kramer, J. and Magee, J. (2007). Self-Managed Systems: an Architectural Challenge. In Proceedings of Future of Software Engineering (FOSE '07), pp 259-268
- [Schmerl et al., 2006]
 - Schmerl, B., Aldrich, J., Garlan, D., Kazman, R., and Yan, H. (2006). Discovering Architectures from Running Systems. In IEEE Transactions on Software Engineering, vol 32, no. 7, pages 454-466

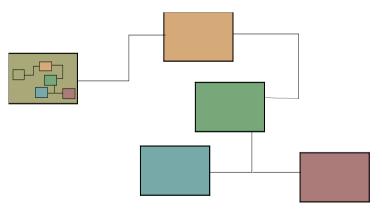
Reflection



Architectural reflection

- The ability for a system to observe and possibly modify its (runtime) architecture
- Is concerned with reification of whole systems rather than e.g objects reachable within a given programmatic scope
- If reflection in e.g. Java or Lisp is reflection-in-the-small, then architectural reflection is reflectionin-the-large.





Styles of Reflection



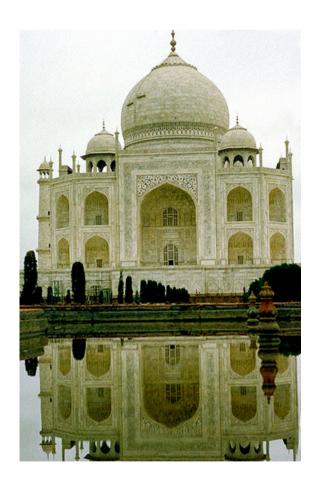
Structural vs behavioral

- Structural reflection is the representation of (static) structure of the system
- Behavioural reflection is the representation of ongoing activity

Procedural vs declarative

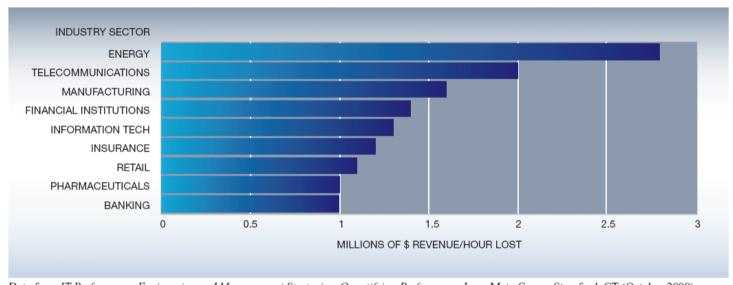
- Procedural reflection involves direct manipulation of the system
- In contrast, declarative reflection involves manipulation of a more abstract representation

[Gordon Blair]



The Complexity Problem





Data from IT Performance Engineering and Measurement Strategies: Quantifying Performance Loss, Meta Group, Stamford, CT (October 2000).

Autonomic Computing



AARHUS UNIVERSITET

- Computer systems that manage themselves
 - Given high-level objectives from administrator
- Initiative spearheaded by IBM

| BASIC LEVEL 1 | MANAGED LEVEL 2 | PREDICTIVE LEVEL 3 | ADAPTIVE LEVEL 4 | AUTONOMIC LEVEL 5 |
|-----------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| MULTIPLE SOURCES OF SYSTEM GENERATED DATA REQUIRES EXTENSIVE, HIGHLY SKILLED IT STAFF | CONSOLIDATION OF DATA THROUGH MANAGEMENT TOOLS IT STAFF ANALYZES AND TAKES ACTIONS | SYSTEM MONITORS, CORRELATES, AND RECOMMENDS ACTIONS IT STAFF APPROVES AND INITIATES ACTIONS | SYSTEM MONITORS, CORRELATES, AND TAKES ACTION IT STAFF MANAGES PERFORMANCE AGAINST SLAS | INTEGRATED COMPONENTS DYNAMICALLY MANAGED BY BUSINESS RULES/POLICIES IT STAFF FOCUSES ON ENABLING BUSINESS NEEDS |
| | GREATER SYSTEM AWARENESS IMPROVED PRODUCTIVITY | REDUCED DEPENDENCY ON DEEP SKILLS FASTER AND BETTER DECISION MAKING | IT AGILITY AND RESILIENCY WITH MINIMAL HUMAN INTERACTION | BUSINESS POLICY DRIVES IT MANAGEMENT BUSINESS AGILITY AND RESILIENCY |

Autonomic Computing – Status



Products

- Parts of, e.g., IBM Tivoli and MS SQL are autonomic

Specifications

 Oasis's Web Services Distributed Management (WSDM)

Research and Development

- Large companies
 - IBM: Autonomic Computing project
 - HP: Adaptive Enterprise project
 - MS: Data Management, Exploration and Mining (DMX) group
- Google Scholar
 - "Autonomic Computing" -> 6.530 results...

Robotics-Inspired Three-Layer Model



Reference model for self-managed/autonomic Create new plans based systems on high-level objectives **Goal Management** Plan Requests Execute pre-computed Change Plans plans **Change Management** Application control Sensors, actuators **Status Change Actions Component Control** 8

[Schmerl et al., 2005]: Dynamically Discovering Architectures with DiscoTect

Discovering Architecture

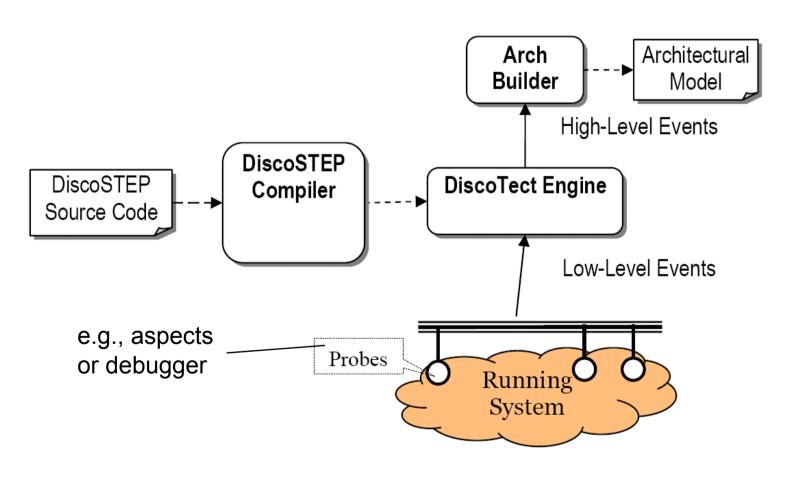
- Mapping of runtime system events to architectural events constructing C&C views
- Online analysis while monitored system is running
- Hooks into AcmeStudio

Challenges

- Runtime system events may have a many-to-many relationship with architectural events
- Architecturally relevant events are interleaved in systems
- Architectural elements may be implemented in differing ways

DiscoTect Overview





(Cf. reference [21])

DiscoSTEP



AARHUS UNIVERSITET

Specification language for mapping between runtime system events and architectural events

```
Input event con-
                          rule CreateServer {
    sumed by rule
                            input { init $e; }
                            output { string $server_id;
Output events
                                      create component $create server; }
produced by rule
                            trigger {?
                              contains($e/@type, "ServerSocket")
                            ?}
  Condition, determining
                            action = \{?
  whether the rule will fire
                              let $server_id := $e/@instance_id;
                              let $create_server :=
 Actions that produce
                                 <create_component name="{$server_id}"</pre>
 output events
                                                      type="ServerT" />;
```

- Triggers and actions use XQuery
- Rules may be composed with other rules via output/input events

DiscoSTEP – Events



AARHUS UNIVERSITET

Input

```
<init
type= "java.net.ServerSocket"
instance_id="0x0f67d9"
/>
```

Output

```
<string value="0x0f67d9"/>
<create_component
name="0x0f67d9" type="ServerT"/>
```

Code

```
ServerSocket ss = new ServerSocket(1111)
```

Event schema, input

DiscoSTEP – Formal Semantics



Using Coloured Petri Nets (CP-Nets)

```
rule CreateServer {
                                                                                                              string
  input { init_$e; }
                                                                           [contains(...)]
  output { string $server_id;
          create component $create_server;
  trigger {?
                                                                                            $server to
    contains($e/@type, "ServerSocket")
                                                         init
                                                                      $e
  action = \{?
    let $server id := $e/@instance id;
    let $create server :=
                                                                                            $create_server
      <create_component name="{$server_id}"</pre>
                         type="ServerT" />;
                                                                                                     create component
```

DiscoTect Example (1)



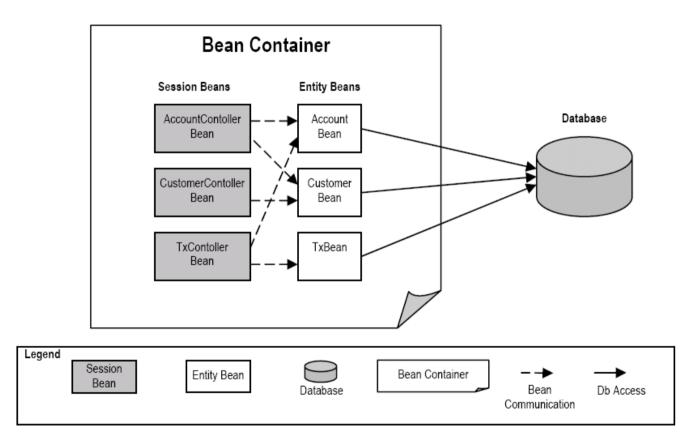


Figure 13. Documented architectural view of Duke's Bank Application

DiscoTect Example (2)



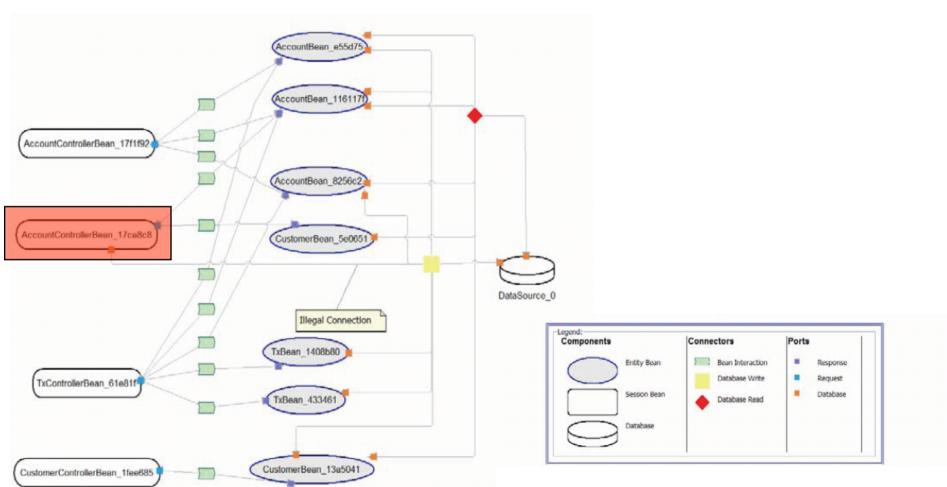


Figure 14. Discovered architecture of Duke's Bank

Summary



Three parts of architecture reflection

- Monitor, model, control
- Three-layer model

Modeling and controlling is hard

Examples

- Autonomic computing
- DiscoTect