Motivation

» What to change in order to improve the software design?
**Problem statement**

» Software performance process

1. Detecting performance flaws:
   How to explore the architectural models to recognize performance problems?

2. Solving performance flaws:
   What are the refactoring actions that lead the architectural model to remove performance problems?
Model Refactoring approach in deep...

Architectural Model “Candidates”

Termination Criteria:
(i) fulfilment criterion
(ii) no-refactorings criterion
(iii) #iterations

Challenges in “Software Performance Model Refactoring”

Software

Uncertainty

Performance

- implementation
- architecture
- specification
- design
- requirement
- Software development
- respond time
- workload
- scheduling policy
- throughput
- operational profile
- bottleneck
- Performance analysis
Software Models Refactoring Uncertainty

What are the refactoring actions that lead uncertainty in the software models?

- Add/Change/Delete software model elements;
- Add/Change/Delete connections between software model elements, i.e. modifying the topology;
- Add/Change/Delete interactions between software model elements, i.e. modifying the behavior;
- Further issues: lack of information (e.g. workload, operational profile, ...).

Performance Models Refactoring Uncertainty

What are the refactoring actions that lead uncertainty in the performance models?

- Add/Change/Delete performance model elements;
- Add/Change/Delete connections between performance model elements, i.e. modifying the topology;
- Add/Change/Delete interactions between performance model elements, i.e. modifying the behavior;
- Further issues: lack of information (e.g. service time distribution, waiting policy, routing strategy, ...).
An example: E-commerce System (1/2)

Software and Performance Modeling

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Required Value</th>
<th>Predicted Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT(browseCatalog)</td>
<td>1.2 sec</td>
<td>1.4 sec</td>
</tr>
<tr>
<td>RT(makePurchase)</td>
<td>2.5 sec</td>
<td>2.69 sec</td>
</tr>
</tbody>
</table>

An example: E-commerce System (2/2)

Software and Performance Refactoring

<table>
<thead>
<tr>
<th>(Annotated) Software Architectural Model</th>
<th>RT(browseCatalog) ≤ 1.2 sec</th>
<th>RT(makePurchase) ≤ 2.5 sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECS</td>
<td>1.4 sec</td>
<td>2.69 sec</td>
</tr>
<tr>
<td>ECS - Refactored - 1</td>
<td>1.4 sec</td>
<td>(2.32, 2.37, 2.42) sec</td>
</tr>
<tr>
<td>ECS - Refactored - 2</td>
<td>(1.4, 1.6, 1.8, 2.0) sec</td>
<td>(2.69, 2.97, 3.22, 3.47) sec</td>
</tr>
<tr>
<td>ECS - Refactored - 3</td>
<td>(1.01, 1.19, 1.39, 1.59) sec</td>
<td>(1.63, 1.67, 1.82, 2.02) sec</td>
</tr>
<tr>
<td>ECS - Refactored - 4</td>
<td>1.4 sec</td>
<td>2.18 sec</td>
</tr>
<tr>
<td>ECS - Refactored - 5</td>
<td>1.4 sec</td>
<td>2.32 sec</td>
</tr>
<tr>
<td>ECS - Refactored - 6</td>
<td>(1.39, 1.4, 1.4) sec</td>
<td>(2.6, 2.65, 2.69) sec</td>
</tr>
<tr>
<td>ECS - Refactored - 7</td>
<td>1.3 sec</td>
<td>2.58 sec</td>
</tr>
</tbody>
</table>

Refactoring at a glance (1/2)

» ECS: detecting performance flaws

Refactoring at a glance (2/2)

» ECS - Refactored: solving performance flaws
Research Agenda

"I want you"... not for U.S. Army ;) !!!

Advertisement

PANDA: “Performance Antipatterns and Feedback in Software Architectures”

- System Modeling Language (i.e. UML + Marte profile)
- Domain Specific Language (i.e. Palladio Component Model)
- Architectural Description Language (i.e. AEmilia)

...any modeling language with “antipatterns-concepts”

http://code.google.com/p/panda-aemilia/
Thank you!

Questions?

catia.trubiani@univaq.it