Towards the Identification of "Guilty" Performance Antipatterns

Vittorio Cortellessa, Anne Martens, Ralf Reussner, Catia Trubiani

The problem of interpreting the results of software performance analysis is very critical. Software developers expect feedback in terms of architectural design alternatives (e.g., re-deploy a component), whereas the results of performance analysis are pure numbers. Support to the interpretation of such results that helps to fill the gap between numbers and software alternatives is still lacking. Performance antipatterns can play a key role in the search of performance problems and in the formulation of their solutions.

In this poster, we introduce a process to elaborate the analysis results and to score performance requirements, model entities and guilty performance antipatterns. We illustrate the process with exemplary values.

**Annotated Software System Model**

**Input: Requirements:**
- R1: RT(report) < 2.5 sec
- R2: util(Proc1) < 80%
- R3: RT(maintain) < 3 sec

**Input: Software System Model**

**Violated Requirements with Involved Entities:**
- R1: Webserver, Proc1, Scheduler, GraphicReporting, Proc2
- R2: Webserver, Proc1

**Filtered Antipatterns List**

- PA1: Concurrent Processing Systems
- PA2: Extensive Processing
- PA3: Blob
- PA4: Empty Semi Trucks

**Ranked Antipatterns List:**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Antipattern</th>
<th>Score 1</th>
<th>Score 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA1</td>
<td></td>
<td>0.3</td>
<td>0.15</td>
</tr>
<tr>
<td>PA2</td>
<td></td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>PA3</td>
<td></td>
<td>0.05</td>
<td>0.03</td>
</tr>
</tbody>
</table>

**Annotate the system model**

**Complete Antipattern List with Involved Entities**

- PA1: Concurrent Processing Systems
- PA2: Extensive Processing
- PA3: Blob
- PA4: Empty Semi Trucks

**New Software System Model**

**Input: Antipattern Rules:**
- IF util(Proc1) > 3-util(Proc2) then Concurrent Processing Systems

**Rule engine matching**

**Keywords:** Performance Prediction, Software Architecture, Antipatterns

**Related Work**

Xu et al. [4] present a semi-automated approach to find configuration and design improvement on the performance model level based on LQN models.

Parsons et al. [3] present a framework for detecting performance antipatterns in Java EE architectures. The method requires an implementation of a component-based system, which can be monitored for performance properties.

Diaz Pace et al. [2] have developed the ArchE framework. ArchE assists the software architect during the design to create architectures that meet quality requirements. It helps to create architectural models and suggests improvements.

A more detailed presentation of this work can be found in [1].

**References**


