Using the CoCoME community case study for evaluating the SQuAT architecture optimization approach: results and experiences

Sebastian Frank
Santiago Vidal
SQuAT Approach

Motivation

- **Difficulties** to manage and optimize all QAs in a satisfying way, i.e., make everybody happy
- Monolithic analysis is **unnatural**
- Trade-offs between different QA goals & **negotiation among experts** in software architecture
SQuAT Approach

The Bots

Abstract Bot

Scenario (Goal)
Model Analyzer
Model Optimizer

Performance Bot

Workload, Usage Behaviour, CPU Clock Rate
Headless Palladio LQN Solver
Headless PerOpteryx (Evolutionary Algorithm)

Modifiability Bot

Add, Remove, Modify Components & Interfaces
KAMP Extension (Change Impact Analysis)
Henshin Rules
SQuAT Approach

The algorithm

The SQuAT architecture optimization approach consists of the following steps:

1. **Initial Candidate**
   - This step involves the creation of an initial candidate model.

2. **Analyze**
   - This step involves analyzing the initial candidate model.

3. **Optimize**
   - This step involves optimizing the initial candidate model.

4. **Negotiate**
   - This step involves negotiating the optimized candidate model to create proposed candidates.

5. **Proposed Candidates**
   - This step involves proposing candidates for further analysis.

6. **Next Level**
   - This step involves moving to the next level of analysis.

7. **Local search**
   - This step involves conducting local search to refine the proposed candidates.

8. **Global search**
   - This step involves conducting global search to explore more extensive options.

The SQuAT approach uses Palladio Component Model as an initial step to create an initial candidate model. The process then moves through an iterative cycle of analysis, optimization, negotiation, and proposal to create new candidates for further refinement. The approach emphasizes the importance of both local and global search techniques in the optimization process.
Case Study: ST+

- Size: 5 Components, 4 Alternative Components
- Scenarios: 2 Performance Scenarios, 2 Modifiability Scenarios
- Levels: 2
- Quality: Fully Modeled "Clean"

Case Study: CoCoME

- Size: 51 Components, 5 Alternative Components
- Scenarios: 4 Performance Scenarios, 4 Modifiability Scenarios
- Levels: 5
- Quality: Partially Modeled (~25%) "Dirty"

CoCoME

See SSP '17 and/or "Distributed Quality-Attribute Optimization of Software Architectures" (SBCARS)

## Using the CoCoME community case study for evaluating the SQuAT architecture optimization approach: results and experiences

### About

- Business Trip Management
- Trading System (supermarket)

### Size

- 5 Components
- 4 Alternative Components
- 51 Components
- 5 Alternative Components

### Scenarios

- 2 Performance Scenarios
- 2 Modifiability Scenarios
- 4 Performance Scenarios
- 4 Modifiability Scenarios

### Levels

- 2
- 5

### Quality

- Fully Modeled "Clean"
- Partially Modeled (~25%) "Dirty"
Case Study
Scenarios in the CoCoME Case Study

<table>
<thead>
<tr>
<th>Name</th>
<th>Stimulus</th>
<th>Expected Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Overall Usage Increase +10%</td>
<td>&lt; 1.2 seconds</td>
</tr>
<tr>
<td>P2</td>
<td>Overall Usage Increase +50%</td>
<td>&lt; 1.4 seconds</td>
</tr>
<tr>
<td>P3</td>
<td>Server Fault (1 out of 2 in a cluster)</td>
<td>&lt; 1.0 seconds</td>
</tr>
<tr>
<td>P4</td>
<td>Usage Increase of Specific Service</td>
<td>&lt; 2.4 seconds</td>
</tr>
<tr>
<td>M1</td>
<td>Add NFC Payment Method</td>
<td>&lt; 2270 (complexity)</td>
</tr>
<tr>
<td>M2</td>
<td>Add Premium User Service</td>
<td>&lt; 750 (complexity)</td>
</tr>
<tr>
<td>M3</td>
<td>Add Withdraw Money Function</td>
<td>&lt; 170 (complexity)</td>
</tr>
<tr>
<td>M4</td>
<td>Add Logging Function</td>
<td>&lt; 2180 (complexity)</td>
</tr>
</tbody>
</table>
Experiences

Model: Preparations

- Model Allocation, Usage Behaviour, ...
- Understanding CoCoME
- Call Graph Tool
  - Side Effect: Bug Fixing
Experiences
Model: Limitations

- Events
- Composite Components

What are “Events”?

What are “Composite Components”?

Initial Candidate

Replace

Proposed Candidates

SQuAT

Results

Experiences

Case Study

SQuAT
Using the CoCoME community case study for evaluating the SQuAT architecture optimization approach: results and experiences

**Experiences**

Model: Bad Practices

- **Empty Interface**
  - Component without Interface
  - Initial Candidate

- **SQuAT**
  - I can replace this component with an arbitrary one.
  - Error

- **Proposed Candidates**

---

**SQuAT**

---

Using the CoCoME community case study for evaluating the SQuAT architecture optimization approach: results and experiences
11/08/2018
Experiences
Model: Validity

Sometimes my transformations fail
I will filter them out!
Experiences

Scalability

Size: Big

Initial Candidate

Level: 5

Seed Selection

Partial Parallelization

Wait some minutes!

See you in 3-5 days!

Filter population.

Wait some seconds!

See you in 25 days!

Model Caching

Proposed Candidates

Results

Case Study

Future Work

Using the CoCoME community case study for evaluating the SQuAT architecture optimization approach: results and experiences
Experiences
CPU Clock Rate Ripple Effect

Let’s use it!
Again!
Again!

Max. CPU Clock Rate Factor: 2x

I don’t care
I like your candidates!

Proposed Candidates

4 GHz

Initial Candidate

SQuAT
Experiences

CPU Clock Rate Ripple Effect

Model Optimization

1. Step:
Automatically Generate Configuration & Optimize Candidate

Headless PerOpteryx

Using the CoCoME community case study for evaluating the SQuAT architecture optimization approach: results and experiences

11/08/2018
Experiences

CPU Clock Rate Ripple Effect

I have a new parameter

Model Optimization

1. Step: Automatically Generate Configuration

2. Step: Modify Configuration

3. Step: Optimize Candidate

Headless PerOpteryx

Using the CoCoME community case study for evaluating the SQuAT architecture optimization approach: results and experiences
Experiences

Utility Function

My scenarios are satisfied

Worsen performance is more beneficial than improving modifiability

What about me?

Initial Candidate

Proposed Candidates

New Utility Function

SQuAT

Using the CoCoME community case study for evaluating the SQuAT architecture optimization approach: results and experiences
Results
M1 - P1

- Many “good” agreements
- satisfy Performance Scenario 1
- satisfy Modifiability Scenario 1
- Higher levels are useful
Using the CoCoME community case study for evaluating the SQuAT architecture optimization approach: results and experiences

SQuAT ➤ Case Study ➤ Experiences ➤ Results ➤ Future Work

Results
More Results

Initial Candidate

M2 - P2
M3 - P3
M4 - P4

level
0
1
2
3
4
5
Future Work

Lessons Learned

- Model
  - Quality ("Bad Practices")
  - Exception Handling Strategies
    - Filters, ...
- Visualization
  - Understand Changes
  - Compare Models
- Interaction
  - New Insights at Runtime
  - Human Knowledge
Future Work

Visualization & Interaction

![Diagram showing candidate scenarios with modifiability and performance metrics]

Candidate 1 | Candidate 2 | Candidate 3 | Candidate 4
---|---|---|---
Candidate 5 | Candidate 6 | Candidate 7 | Candidate 8

CONTINUE
(Take Selected Candidates as Seed)

FINISH
(Export Selected Candidates)

Modifiability | Performance

- Modifiability Scenario 1: 80%
- Performance Scenario 1: 97%
- Modifiability Scenario 2: 57%
- Performance Scenario 2: 75%
- Modifiability Scenario 3: 41%
- Performance Scenario 3: 82%
- Modifiability Scenario 4: 29%
- Performance Scenario 4: 18%
Future Work

More Ideas

• Personalization of Bots
  • Own Utility Function, ...

• Scalability
  • Microservices
  • Further Parallelization, Caching, ...

• Usability
  • Natural Language
    • Scenarios & Results
  • Machine Learning