Towards automation of architectural tactics application – an example with ArchE

Roger Champagne, Eng., Ph. D. (presenter)
Sébastien Gagné, Eng., M. Eng.

Dept. of Software and IT Engineering
ÉTS (University of Québec)
Montréal, Canada

roger.champagne@etsmtl.ca
sebastien.gagne@gmail.com

Exploring a solution space

Version of the architecture
Architectural decision

... !!!
Outline

• Motivation and context
• Overview of Software Architecture concepts
• Overview of ArchE
• Performance Architectural Tactics
• Results via an example
• Conclusions and future work

Motivation

• Software Architecture (SA) is important
• Many useful SA-related concepts and approaches proposed and used in practice
  – Quality Attribute Scenarios (QAS)
  – Architectural Tactics
  – Attribute-Driven Design (ADD)
  – ...
• We are interested in finding ways to better support architects with tools
  – Many SA-related approaches/tasks performed manually and require high level of expertise
Context [1/]

• The Software Engineering Institute (SEI) developed the Architectural Expert (ArchE)
• ArchE v3
  - Has core architectural concepts implemented in an extensible way
  - Can reason about two quality attributes

<table>
<thead>
<tr>
<th>Quality attribute</th>
<th>Analysis</th>
<th>Tactics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modifiability</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Performance</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>

Context [2/]

• Short term goal
  - Understand what is involved when transforming informally described tactics to a form that can be used by a tool that automates some SA-related tasks
    • Implement tactics support for performance in ArchE

• Long term goal
  - Contribute to more/better tools that support SA elaboration

• Constraint
  - Performance tactics support added to ArchE with minimal modifications to existing performance capabilities
Outline

- Motivation and context
- Overview of Software Architecture concepts
  - Quality Attribute Scenarios
  - Architectural Tactics
  - Responsibilities
  - Reasoning Frameworks
- Overview of ArchE
- Performance Architectural Tactics
- Results via an example
- Conclusions and future work

Quality Attribute Scenarios [1/]

- Used to describe quality attribute (often called non-functional) requirements, such as performance, modifiability, availability, reliability, ...
- Typically expressed in natural language
- Composed of six parts¹
  - Stimulus
    - Stimulus source
    - Artefact
    - Environment
  - Response
  - Response measure

Towards automation of architectural tactics application – an example with ArchE

SATURN 2011, May 16-20 2011

By R. Champagne, S. Gagné (ÉTS, Montréal, Canada)

Page 5

Quality Attribute Scenarios [2/]

• Example performance QAS

<table>
<thead>
<tr>
<th>Scenario part</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stimulus source</td>
<td>A position sensor</td>
</tr>
<tr>
<td>Stimulus</td>
<td>Transmits new position values every 130 ms</td>
</tr>
<tr>
<td>Artefact</td>
<td>Running system</td>
</tr>
<tr>
<td>Environment</td>
<td>Under normal conditions</td>
</tr>
<tr>
<td>Response</td>
<td>Each new value acquired, validated and stored</td>
</tr>
<tr>
<td>Response measure</td>
<td>Within 130 ms</td>
</tr>
</tbody>
</table>

A position sensor transmits new position values every 130 ms when the system is running under normal conditions. Each new sensor value must be acquired, validated and stored in the system before the next sensor update arrives, i.e. within 130 ms.

Architectural Tactics

• Definition
  - A design decision that influences the control of a quality attribute response

• Tactics identified for six Quality Attributes
  - Availability
  - Modifiability
  - **Performance**
  - Security
  - Testability
  - Usability

Responsibilities

• Definition
  - An activity undertaken by the software being designed

• In our work, a responsibility represents both a "function", and the architectural element that realizes the function
  - Equivalent to having
    • Responsibilities on one hand
    • Some architectural elements (e.g. processes) on the other
    • A one-to-one mapping between the two

Reasoning frameworks

• A vehicle for encapsulating the quality attribute knowledge needed to understand a system’s quality behavior
  - Can be used by non-experts
  - Composed of
    • Problem description
    • Analytic theory
    • Analytic constraints
    • Model representation
    • Interpretation
    • Evaluation procedure


Outline

- Motivation and context
- Overview of Software Architecture concepts
- Overview of ArchE
  - General
  - Performance RF
- Performance Architectural Tactics
- Results via an example
- Conclusions and future work

ArchE

- "A design assistant that helps architects explore architectural designs driven by quality attributes. ArchE shows the architect proposals for improving the current architecture and allows the architect to decide on the best alternative" 4
- ArchE is an Eclipse application

ArchE workflow

- Supply functional requirements
  - Responsibilities, their parameters and their relationships
    - Responsibilities are mapped to architectural elements
- Supply QAS and map them to responsibilities
- Launch analysis
  - Verifies if the current architecture satisfies the QAS
- If at least one QAS not satisfied and tactics supported
  - ArchE proposes a set of tactics, from which the user selects one to apply
    - User might be required to supply parameters depending on tactic
- Analysis is run again after every change, and a new set of tactics is proposed (if relevant)

ArchE v3 performance RF [1/]

- Pre-emptive multitasking
  - Fixed-priorities
  - Single CPU
  - Rate-Monotonic scheduling
- Analyzes performance QAS
  - That have periodic stimuli
  - Where response measure is a hard deadline
Towards automation of architectural tactics application – an example with ArchE

ArchE v3 performance RF [2/]

- **S1**: \( P = 130 \) \( D = 130 \)
  - **R2**: Sensor
    - **Pr**: 3, **C**: 4.6

- **S2**: \( P = 450 \) \( D = 450 \)
  - **R9**: Trajectory planning
    - **Pr**: 4, **C**: 90.5

- **S3**: \( P = 140 \) \( D = 150 \)
  - **R8**: Movement planning
    - **Pr**: 16, **C**: 21

- **S4**: \( P = 2000 \) \( D = 100 \)
  - **R5**: Monitor
    - **Pr**: 2, **C**: 0.5

- **R6**: Position Monitor (write)
  - **Pr**: 12, **C**: 15

**Scenario x stimulus**
- **A**: \( P = \) period (ms)
- **B**: \( D = \) deadline (ms)
- **Sx**: \( P = \) priority
- **D**: \( C = \) execution time (ms)

**Responsibility y**
- **Pr**: priority
- **C**: execution time (ms)

**Modified version of the "Robot Controller" example.**


ArchE v3 performance RF [3/]

**ArchE**
- Scenarios
  - Period
  - Deadline
  - Responsibilities
    - Priority
    - Execution time

**Lambda-WBA\(^5\)**
- ICM
  - Component
  - Service
  - Scenario
  - Pin (sink, source)

**MAST\(^6\)**
- **MST**
  - Processor
  - Scheduler
  - Scheduling server
  - Transaction
  - Operation

**Scenarios - Responsibilities mapping**

**Responsibility relationships**

**For each scenario:**
- **Satisfied**
- **Not satisfied**

**Scenario met (Y / N)**

**Operation**
- Best response time
- Worst response time
- Scenario met (Y / N)

**MAST – for each transaction:**
- Best response time
- Worst response time
- Jitter
- Slack time
- Nb. of suspensions

---


Outline

• Motivation and context
• Overview of Software Architecture concepts
• Overview of ArchE
  • Performance Architectural Tactics
    - Current state
    - Mapping to and implementation within ArchE
• Results via an example
• Conclusions and future work

Performance tactics\(^1\)

Mapping tactics to ArchE's performance RF

- Constrained by existing performance RF capabilities
  - Our goal: add tactics support to the RF, not redesign it

<table>
<thead>
<tr>
<th>Original tactic</th>
<th>Our implementation</th>
<th>Our description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling policy</td>
<td>Increase responsibility priority</td>
<td>Identify responsibilities whose low priority has a negative impact on the overall latency of one or more scenarios which are not satisfied</td>
</tr>
<tr>
<td>Manage event rate</td>
<td>Increase scenario period</td>
<td>Scenario period is the inverse of event rate. This in effect relaxes the requirements, it does not change the &quot;architecture&quot; (responsibilities).</td>
</tr>
<tr>
<td>Increase computation efficiency</td>
<td>Reduce responsibility execution time</td>
<td>A responsibility is changed to be more efficient, reducing its execution time. (e.g., a more efficient algorithm)</td>
</tr>
<tr>
<td>Increase available resources</td>
<td>Reduce responsibility execution time</td>
<td>User supplies a single value (percentage by which resources are increased), which translates to a proportional reduction in the execution time of all responsibilities</td>
</tr>
</tbody>
</table>

Automatic suggestion of tactics [1/]

- Goal: only suggest a given tactic if
  - At least one performance QAS is not met
  - Tactic application has a potential to improve the current solution

- Potential improvement to architecture from tactic application: established with rules
  - For two of the tactics we implemented, the rules are based on Rate-Monotonic Analysis (RMA7)
  - The two other tactics are always proposed as soon as at least one performance QAS is not satisfied
    - Increase available resources
    - Reduce responsibility execution time

Automatic suggestion of tactics [2/]

- **Increase scenario period**
  - **Goals**
    - Identify QAS whose period are important contributors to latency
    - Suggest new (reasonably larger) period values for these
  - RMA identifies two groups of candidate scenarios

\[
a_{n+1} = S + \sum_{i=1}^{a_n} C_i + \sum_{k \in H(1)} a_n T_k C_k
\]

1. The scenario under study (e.g. the one for which we are currently computing Worst-Case Latency - WCL)
2. All scenarios in group "H" - scenarios for which all responsibilities have higher priorities than the responsibilities associated to the scenario under study

Automatic suggestion of tactics [3/]

- **Increase responsibility priority**
  - **Goal:** identify responsibilities whose low priority are important contributors to latency
  - **Two important priority values**
    - \( p_{\text{min}} \) - the lowest priority of all responsibilities mapped to the scenario of interest (one that is not satisfied)
    - \( p_{\text{sup}} \) - the lowest priority of a responsibility that is just above \( p_{\text{min}} \) and is associated to a responsibility that is not part of the scenario of interest
  - **Rule:** suggest changing the priority of all responsibilities that are part of the scenario under study and have a priority value \( P \) such that \( p_{\text{min}} \leq P \leq p_{\text{sup}} \) to a value that is just above \( p_{\text{sup}} \)
Towards automation of architectural tactics application – an example with ArchE

SATURN 2011 - Burlingame, California, May 16-20 2011

By R. Champagne, S. Gagné (ÉTS, Montréal, Canada)

Page 13

Outline

• Motivation and context
• Overview of Software Architecture concepts
• Overview of ArchE
• Performance Architectural Tactics
  – Current state
  – Mapping to and implementation within ArchE
• Results via an example
• Conclusions and future work

Example – Robot Controller [1/]

<table>
<thead>
<tr>
<th>Scen. ID</th>
<th>Scen. period (ms)</th>
<th>Scen. ddln.</th>
<th>Resp. ID</th>
<th>Resp. exec. time (ms)</th>
<th>Resp. prio.</th>
<th>Scen. WCL (ms)</th>
<th>Scen. ddln. met</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>130</td>
<td>130</td>
<td>R2</td>
<td>5.6</td>
<td>3</td>
<td>500</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R6</td>
<td>15</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>450</td>
<td>450</td>
<td>R9</td>
<td>90.5</td>
<td>4</td>
<td>333</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R7</td>
<td>15</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R1</td>
<td>22</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>140</td>
<td>150</td>
<td>R8</td>
<td>21</td>
<td>16</td>
<td>115</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R3</td>
<td>13.5</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R6</td>
<td>15</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R4</td>
<td>13.5</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>2000</td>
<td>100</td>
<td>R5</td>
<td>0.5</td>
<td>2</td>
<td>10776</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R6</td>
<td>15</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Column heading acronyms: Scen. ID = Scenario ID, Scen. period = period of scenario stimulus, Scen. ddln. = scenario deadline, Resp. ID = responsibility ID, Resp. exec. time = responsibility execution time, Resp. prio. = responsibility priority (high value = high priority), Scen. WCL = scenario worst-case latency, Scen. ddln. met = scenario deadline met.
Example – Robot Controller [2/]

- ArchE initially proposes 11 tactic instances, user applies the "increase available resources" tactic, with an increase of 10%
  - All responsibility execution times are reduced
  - WCL decreases for all scenarios, S1 and S4 still not satisfied
- ArchE suggests increasing the period for S1 and S3. After consideration, the user chooses to increase the period of S3 from 140 ms to 160 ms.
  - WCL for S4 is reduced, but S1 and S4 are still not satisfied
- ArchE suggests increasing computation efficiency for R6 and R7. The user reduces the execution times of these two responsibilities to 10.5 ms and 7 ms, respectively.
  - WCL decreases for all scenarios, but S1 and S4 are still not satisfied

Example – Robot Controller [3/]

- ArchE suggests increasing the priority of R2 from 3 to 5, user applies this tactic
  - S1 is now satisfied, S4 is still not satisfied
- ArchE suggests increasing priority of R5 from 2 to 5, user applies this tactic
  - S4 is still not satisfied
- ArchE suggests increasing priority of R5 again, this time from 5 to 6, user applies this tactic
  - S4 is now satisfied
## Example - Robot Controller [4/]

<table>
<thead>
<tr>
<th>Scen. ID</th>
<th>Scen. period (ms)</th>
<th>Scen. ddln. (ms)</th>
<th>Resp. ID</th>
<th>Resp. exec. time (ms)</th>
<th>Resp. prio.</th>
<th>Scen. WCL (ms)</th>
<th>Scen. ddln. met</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>130</td>
<td>130</td>
<td>R2</td>
<td>5.0</td>
<td>5</td>
<td>107.6</td>
<td>Yes</td>
</tr>
<tr>
<td>S2</td>
<td>450</td>
<td>450</td>
<td>R6</td>
<td>10.5</td>
<td>12</td>
<td>301.5</td>
<td>Yes</td>
</tr>
<tr>
<td>S3</td>
<td>160</td>
<td>150</td>
<td>R9</td>
<td>82.2</td>
<td>4</td>
<td>91.6</td>
<td>Yes</td>
</tr>
<tr>
<td>S4</td>
<td>2000</td>
<td>100</td>
<td>R5</td>
<td>0.4</td>
<td>6</td>
<td>92.0</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Values that changed with respect to the initial problem are **bold underlined**.

| Column heading acronyms: | Scen. ID = Scenario ID, Scen. period = period of scenario stimulus, Scen. ddln. = scenario deadline, Resp. ID = responsibility ID, Resp. exec. time = responsibility execution time, Resp. prio. = responsibility priority (high value = high priority), Scen. WCL = scenario worst-case latency, Scen. ddln. met = scenario deadline met. |

---

## Outline

- Motivation and context
- Overview of Software Architecture concepts
- Overview of ArchE
- Performance Architectural Tactics
- Results via an example
- Conclusions and future work
Conclusions [1/]

- **Goal:** Implement performance tactics support in ArchE
  - To understand what is involved when transforming informally described tactics to a form that can be used by a tool that automates some SA-related tasks

- **Main lessons**
  - This type of tool is useful to explore a solution space more efficiently
  - The analysis tool constrains tactics possibilities
    - In this case, performance analysis and tactics support were developed separately, many years apart
  - Adding tactics support may require new parameters
    - User-accessible priority added in responsibilities in our case

Conclusions [2/]

- **Main lessons (continued)**
  - Key step: establishing rules to determine when each tactic might apply
    - Constrained by the information available

- **Main contributions**
  - Implementation of some performance tactics in ArchE
  - Elaboration of rules to determine when and where a tactic might apply
  - Improved design for tactics support in ArchE's RFs
    - Less coupling between tactics and the rest of the RF than modifiability RF
Future work

• ArchE's performance RF
  - Use MAST directly, instead of via Lambda-WBA
    • In order to gain some flexibility/control for performance tactics support
  - Study more examples, determine if the approach scales
• ArchE in general
  - New reasoning frameworks
    • Cover more quality attributes
  - How to deal with tradeoffs
    • When two or more RFs manipulate the same architectural parameters

Thank you!