QoS Contract Preservation through Dynamic Reconfiguration: A Formal Semantics Approach

Gabriel Tamura, Rubby Casallas, Anthony Cleve, Laurence Duchien
The Quest for Quality-of-Service in Software Services

Software Services:
- In everyday life
- Massively accessed
- Users increasingly depend on them

... but also require increasing quality of service guarantees!

QoS Software Contracts
Quality-of-Service Challenges in SaaS - Cloud Computing
A QoS Contract Example

Controlling the Black Friday Syndrome: QoS contracts must be preserved at run-time (challenge 1)

How critical is to preserve this QoS contract? (i.e., Can the SaaS/IaaS cope with the next Black Friday's unforeseeable load?)
# QoS Contracts Are Out There: The Amazon EC2's SLA* based on availability for SaaS/IaaS

<table>
<thead>
<tr>
<th>Context Condition</th>
<th>QoS Service Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annual Uptime ≥ 99.95% (based on cumulative RUs) along a Year</td>
<td>No guarantees</td>
</tr>
<tr>
<td>Periods of 5 minutes (« Region Unavailable » state) along a Year</td>
<td></td>
</tr>
<tr>
<td>Outside the « Region Unavailable » state</td>
<td>100% Availability</td>
</tr>
</tbody>
</table>

*Drops below 99.95% grant you a 10% refund of your bill*

*Amazon reserves the right to modify this SLA at anytime*
QoS Contract Preservation Must be Reliable

QoS contracts must be preserved autonomously and reliably (challenge 2)

QoS Contracts are a Reality and a Necessity

Software services must satisfy them:
- At run-time (continuously)
- Autonomously
- Reliably
Presentation Outline

1. Overview
2. Background
3. Contributions
4. Validation
5. Conclusions & Future Work
Overview

QoS-CARE: A Comprehensive Solution to Preserve QoS Contracts Reliably at Run-time

- From a formal foundation
- ... to a concrete software architecture and implementation
- and its validation:
  - Theoretically, on the formal model properties to guarantee its reliability
  - Experimentally, on its practical feasibility
However (at the beginning):

- How to exploit formal models in software engineering (to preserve QoS contracts)?
- Is the result going to be feasible in practical terms?
The Story of QoS-CARE: A Theory for Reliable Preservation of QoS Contracts

An example of the use of Formal Models as foundations for Software Engineering for SAS
2. Background

Finding the path to follow...
The main research question:

How to preserve QoS contracts fulfillment in component-based systems through dynamic reconfiguration reliably and autonomously?
Departure Points (2): Secondary Research Questions

Q1: How to specify QoS contracts?
- Languages and models [Becker, 2008; Jureta et al., 2009]

Q2: What is the QoS contract semantics?
- QoS Contract Models [Meyer, 1992; Beugnard et al., 1999; ISO, 2001; Collet et al., 2005; Comuzzi and Pernici, 2009]
- Formal semantics [Braga et al., 2009; Cansado et al., 2010]
### Departure Points (3):
Secondary Research Questions


- **CBSE Foundations** [Szyperski, 1998; Bachmann et al., 2000; Heineman and Councill, 2001; Papazoglou et al., 2007]
- **CBSE models and frameworks** [Bruneton et al., 2006; Beisiegel et al., 2007a; Bures, 2007; OMG, 2009; Seinturier et al., 2009]
- **SAS Foundations** [Shaw, 1994; Oreizy et al., 1999; Kephart and Chess, 2003; Hellerstein et al., 2004; IBM, 2006; Kramer and Magee, 2007; Müller et al., 2008; Cheng et al., 2009a; de Lemos et al., 2011]
- **SAS Approaches** [Garlan et al., 2003; Litoiu et al., 2005; Solomon et al., 2007; Weyns et al., 2010; Parra et al., 2011; Bentancour et al., 2011]
Departure Points (4): Secondary Research Questions

Q4: How to manage context uncertainty?
- Approaches [Murray et al., 2003; Goldsby and Cheng, 2008; Lin et al., 2009]

Q5: How to evaluate self-adaptive software?
- Approaches [Candea et al., 2004; Garlan et al., 2004; Salehie and Tahvildari, 2009; Cardellini et al., 2009; Cheng et al., 2009b; Léger et al., 2010]
3. Contributions
What are formal models useful for?

Finding the appropriate "handles" to grab the problem

"Give me a place to stand, and I shall move the Earth with a lever"

(Archimedes)
General Contributions

1. Adaptation Properties

2. A Formal Model for QoS Contracts-Preserving Reliable Reconfiguration

3. QoS-CARE Architecture, Implementation, and Validation
Adaptation Properties
Specific Contributions

- Evaluation Framework
  - Characterizing dimensions for SAS systems
  - Identification of quality factors

- Catalog of adaptation properties: means for
  - Standard evaluation
  - Verifying and guaranteeing self-adaptation goals
  - Determining the reliability of QoS-CARE

- >50 analyzed papers
- 20 characterized papers
- 6 dimensions
- 37 classification options

- 4 SASO properties
  - Stability
  - Accuracy
  - Settling time
  - Small overshoot

- 9 adaptation properties

- 4 quality attributes
- 12 quality factors
General Contribution 1: Adaptation Properties

How to determine if a solution (e.g., QoS-CARE) is acceptable?

Adaptation properties are fundamental to certify, compare and combine SAS approaches.

<table>
<thead>
<tr>
<th>Adaptation Property</th>
<th>Quality Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stability</td>
<td>Performance</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
</tr>
<tr>
<td></td>
<td>Throughput</td>
</tr>
<tr>
<td></td>
<td>Capacity</td>
</tr>
<tr>
<td></td>
<td>Safety</td>
</tr>
<tr>
<td></td>
<td>Integrity</td>
</tr>
<tr>
<td></td>
<td>Security</td>
</tr>
<tr>
<td></td>
<td>Integrity</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Performance</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
</tr>
<tr>
<td></td>
<td>Throughput</td>
</tr>
<tr>
<td></td>
<td>Capacity</td>
</tr>
<tr>
<td>Settling Time</td>
<td>Performance</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
</tr>
<tr>
<td></td>
<td>Throughput</td>
</tr>
<tr>
<td>Small Overshoot</td>
<td>Performance</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
</tr>
<tr>
<td></td>
<td>Throughput</td>
</tr>
<tr>
<td></td>
<td>Capacity</td>
</tr>
<tr>
<td>Robustness</td>
<td>Dependability</td>
</tr>
<tr>
<td></td>
<td>Availability</td>
</tr>
<tr>
<td></td>
<td>Reliability</td>
</tr>
<tr>
<td></td>
<td>Safety</td>
</tr>
<tr>
<td></td>
<td>Interact. Complex. Coupling Strength</td>
</tr>
<tr>
<td>Termination</td>
<td>Dependability</td>
</tr>
<tr>
<td></td>
<td>Reliability</td>
</tr>
<tr>
<td></td>
<td>Integrity</td>
</tr>
<tr>
<td>Consistency</td>
<td>Dependability</td>
</tr>
<tr>
<td></td>
<td>Maintainability</td>
</tr>
<tr>
<td></td>
<td>Integrity</td>
</tr>
<tr>
<td>Scalability</td>
<td>Performance</td>
</tr>
<tr>
<td>Security</td>
<td>Security</td>
</tr>
<tr>
<td></td>
<td>Confidentiality</td>
</tr>
<tr>
<td></td>
<td>Integrity</td>
</tr>
<tr>
<td></td>
<td>Availability</td>
</tr>
</tbody>
</table>
General Contribution 2:
A Formal Model for QoS Contracts-
Preserving Reliable Reconfiguration
Example Scenario

Reliable Mobile Videoconference System (RVCS):

- VideoConf-Intra
  - Extension Points: LowBandwidth
    - FromIntranet
    - FromExtranet
- VideoConf-Extra
  - Extension Points: LowBandwidth
    - FromExtranet
- VoiceConf
- QoS Reliability Management
- VideoConf Hold

Represented in a QoS Contract:

**Confidentiality: access point**

<table>
<thead>
<tr>
<th>Contextual Condition</th>
<th>Service Level Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC1: Connection from Intranet</td>
<td>Clear Channel</td>
</tr>
<tr>
<td>CC2: Connection from Extranet</td>
<td>Confidential Channel</td>
</tr>
<tr>
<td>CC3: No Network Connection</td>
<td>Call on Hold</td>
</tr>
</tbody>
</table>

**Availability: bandwidth**

<table>
<thead>
<tr>
<th>Contextual Condition</th>
<th>Service Level Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC4: BandWidth \leq 12</td>
<td>Call on Hold</td>
</tr>
<tr>
<td>CC5: 12 &lt; BandWidth \leq 128</td>
<td>Voice Call</td>
</tr>
<tr>
<td>CC6: 128 &lt; BandWidth</td>
<td>Voice and Video Call</td>
</tr>
</tbody>
</table>
The Component-Based Application Subject to a QoS Contract

Context: Corporate Building

Events (e.g.):
- Change user location (access point, bandwidth, ...)
- Season change
- Calendar date (conference start, ...)

QoS Contract Violation:

<table>
<thead>
<tr>
<th>QoS Contract</th>
<th>Component System</th>
<th>Service Level Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC1: Connection on Intranet</td>
<td>Client Application Runtime</td>
<td>Clear Channel</td>
</tr>
<tr>
<td>CC2: Connection from Extranet</td>
<td>Server Application Runtime</td>
<td>Confidential Channel</td>
</tr>
<tr>
<td>CC3: No Network Connection</td>
<td>Conference Server</td>
<td>Call on Hold</td>
</tr>
<tr>
<td>CC4: BandWidth ≤ 12</td>
<td>Network Adapter</td>
<td>Call on Hold</td>
</tr>
<tr>
<td>CC5: 12 &lt; BandWidth ≤ 128</td>
<td>Conference Server</td>
<td>Voice Call</td>
</tr>
<tr>
<td>CC6: 128 &lt; BandWidth</td>
<td>Conference Server</td>
<td>Voice and Video Call</td>
</tr>
</tbody>
</table>
How to Preserve the Contract under the New Context Condition?

Violating event: move to extranet

Context: intranet

Component-Based Application

Dynamic Structure Reconfiguration

EnDe Cipher

The secure-proxy design pattern

Context: extranet

Reconfigured Component-Based Application
QoS-CARE Approach: TAGTTS

Context: Corporate Building

Component-Based Application

Formal Properties

Context: Parking Lot

Reconfigured Component-Based Application

QoS Contract Compliant System
Formal Models
Specific Contributions

1. Formal Models (e-graphs) for:

1. Component-Based Software Structure

2. QoS Contracts Structure

3. Reconfiguration Rules
   Design patterns encoded in LHSs and RHSs

G. Tamura, A. Cleve, R. Casallas and L. Duchien. FACS'2010.
Formal Models
Specific Contributions

II. Reconfiguration System (TAGTS Extension)

Reconfiguration rules:
- Parameterized in the contract
- Provided by the user (software evolution architect)
- design patterns @ runtime
Is TAGTS+ enough?

Dealing with Context Uncertainty
Formal Models: Specific Contributions

III. QoS Contract Semantics (FSM extension)

- QoS contract states of fulfillment: enriched with unfulfillment states

This semantics provides:
- Autonomous QoS Contract Preservation
- Uncertainty Management
General Contribution 3: QoS-CARE Architecture, Implementation, and Validation
How to implement the Formal Model?

Maintaining the Formal Model Properties while

Sticking to SEfSAS Design Principles
Specific Contribution: The Formal Model in the Unified MAPE-K/Feedback Loop

<table>
<thead>
<tr>
<th>QoS-CARE Reconfiguration Loop Element</th>
<th>QoS_FSM Formal Model Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor</td>
<td>$evalInContext : PREDICATE \times \Delta^U \rightarrow {F,U}$</td>
</tr>
</tbody>
</table>
| Analyzer                            | $- (uses)\ match : CBSRR \times CBSAR \rightarrow Boolean$
|                                     | $- (uses)\ evalInContext : PREDICATE \times \Delta^U \rightarrow \{F,U\}$ |
| Planner                             | $-reconfig : CBSAR \times \mathcal{P}(\Gamma) \rightarrow Boolean$
|                                     | $-match : CBSRR \times CBSAR \rightarrow Boolean$ |
| Executor                            | $-getCBSAR : \rightarrow CBSAR$
|                                     | $-execReconfPlan : INSTRUCTION^* \rightarrow Boolean$ |
| Reconfiguration Knowledge Manager    | $-\Psi : STATES \rightarrow PREDICATE$
|                                     | $-\kappa : STATES \rightarrow PREDICATE$
|                                     | $-\eta : STATES \rightarrow \mathcal{P}(\Sigma)$
|                                     | $-\rho : STATES \rightarrow \mathcal{P}(\Gamma)$
|                                     | $-\pi : STATES \rightarrow QoSProperty$ |
Specific Contribution: QoS-CARE SCA Architecture

1. App in Execution

2. CB to EGraph

3. Context event

4. getSLO & rules

5. compute ReconfPlan

6. Egraph transf & plan synthesis

7. execute ReconfPlan

8. translate plan into SCA platform

9. Fscript execution

10. Fscript reconfiguration primitives instrumentation

New QoS Level Fulfilled
Specific Contribution: QoS-CARE in the SCA Stack

An SCA layer for dynamic reconfiguration in FraSCAti

- CB Application
- QoS-CARE
- FraSCAti Middleware
- JVM
4. Validation
Formal Models Re-evaluated

Not A Solution for Everything, yet A Good Decision
Specific Contribution: QoS-CARE Properties V&V

Reconfiguration Independence and Separation of Concerns in QoS-CARE:
- Re-usable
- Comparable
- Maintainable


Validation Scenarios

Determining the practical feasibility of QoS-CARE
QoS-CARE Applied to Preserve QoS Contracts

(Executed in FraSCATi)

1. A mobile videoconference system (RVCS)
   QoS contract: availability and confidentiality

2. A simple Web mashup application (Twitter / Weather)
   QoS contract: service readiness
QoS-CARE Benchmark
(Overhead and Settling-time)

Benchmark Results
Confirm the
QoS-CARE Practical Feasibility

Rule for changing to a "confident channel" from a "clear channel" configuration.
Instrumented over a ~4Mbps, 11 hops Internet connection.
5. Conclusions & Future Work
Conclusions

Self-adaptive software properties and metrics: to standardize SAS assessment (compare, improve, combine approaches)

Formal model for QoS contracts preservation through self-reconfiguration providing verifiable properties:
- Termination
- Atomicity
- SCA structural conformance
- Robustness
- Settling-time

Architecture and implementation: platform-independent SCA layer for dynamic reconfiguration

Reliability reconfiguration properties: validated formally and experimentally

Benchmark results: practical feasibility (settling-time, overhead), re-usability
Future Work

- Leverage design-patterns @ run-time
- Improve robustness to context uncertainty
- Guarantee additional adaptation properties
Thank You!
Publications

International Journals

Book chapters

International Symposia

International Workshops

Thesis