Information and Software Technology 52 (2010) 1331–1345

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

# Quantification of interacting runtime qualities in software architectures: Insights from transaction processing in client–server architectures

Anakreon Mentis, Panagiotis Katsaros *, Lefteris Angelis, George Kakarontzas

*Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece*

ARTICLE INFO

*Article history:*
Received 22 January 2009
Received in revised form 27 July 2010
Accepted 27 July 2010
Available online 1 August 2010

*Keywords:*
Software architecture validation
Software quality
Design tradeoffs
Transaction processing
Simulation

ABSTRACT

*Context:* Architecture is fundamental for fulfilling requirements related to the non-functional behavior of a software system such as the quality requirement that response time does not degrade to a point where it is noticeable. Approaches like the Architecture Tradeoff Analysis Method (ATAM) combine qualitative analysis heuristics (e.g. scenarios) for one or more quality metrics with quantitative analyses. A quantitative analysis evaluates a single metric such as response time. However, since quality metrics interact with each other, a change in the architecture can affect unpredictably multiple quality metrics.
*Objective:* This paper introduces a quantitative method that determines the impact of a design change on multiple metrics, thus reducing the risks in architecture design. As a proof of concept, the method is applied on a simulation model of transaction processing in client server architecture.
*Method:* Factor analysis is used to unveil latent (i.e. not directly measurable) quality features represented by new variables that reflect architecture-specific correlations between metrics. Separate Analyses of Variance (ANOVA) are then applied to these variables, for interpreting the tradeoffs detected by factor analysis in terms of the quantified metrics.
*Results:* The results for the examined transaction processing architecture show three latent quality features, the corresponding groups of strongly correlated quality metrics and the impact of architecture characteristics on the latent quality features.
*Conclusion:* The proposed method is a systematic way for relating the variability of quality metrics and the implied tradeoffs to specific architecture characteristics.



## 1. Introduction

Software architecture is defined as: "*the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them*" [8]. Architecture sets the boundaries for runtime quality (performance, fault handling, shared resource usage, security etc.) although it cannot be a basis for precise predictions [17], since runtime behavior also depends on implementation details. Nonetheless, software architecture can support analyses that provide confidence for the effects of a design decision (e.g. replication) on quality metrics such as reliability.

We have seen the emergence of methods for analysing quality in software architectures like for example ATAM [14], SBAR [10], SAAM [26] and HoPLAA [42]. All methods combine qualitative analysis heuristics, such as questioning techniques and use cases, with quantitative analyses specific to particular metrics. However, a single change in the software architecture may affect multiple metrics, due to interactions between them [29].

We propose a quantitative method for discovering architecture-specific metric correlations that are affected by the combined effect of architecture characteristics (e.g. level of concurrency, degree of process distribution etc.). These correlations are justified by *latent quality features*, which cannot be measured directly, but they can assist in managing potentially complex tradeoffs.

As a proof of concept, the method is applied on a simulation-based evaluation of a transactional architecture that complies with the Process Coordinator pattern [21]. The pattern is commonly used to implement business processes that issue requests to several server components. Issues like decomposition of the software functionality, allocation of shared resources and communication among the architecture's components form a complex and interesting design problem. Furthermore, experience reports [8] indicate large variations in server availability, performance and scalability of distributed transaction management. The quality metrics investigated in this work are also important for other architecture patterns like the Broker and the Publish-Subscribe pattern [21].

* Corresponding author. Tel.: +30 2310 998532; fax: +30 2310 998419.
*E-mail addresses:* anakreon@csd.auth.gr (A. Mentis), katsaros@csd.auth.gr (P. Katsaros), lef@csd.auth.gr (L. Angelis), gkakaron@teilar.gr (G. Kakarontzas).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.07.006

In the first stage of the proposed method, *factor analysis* is performed. Factor analysis reveals groupings of highly correlated metrics and estimates new and fewer uncorrelated variables that represent the latent quality features. Members of these groupings are positively or negatively correlated. For the transactional architecture at hand, factor analysis is applied on data obtained by a series of experimental runs of the ACID Sim Tools [1,36] simulator, developed by the authors.

In the second stage, separate Analyses of Variance (ANOVA) are performed for each latent quality feature.

The proposed method is a systematic way for relating the variability of quality metrics and the implied tradeoffs to specific architecture characteristics. The analysis uses data that can be obtained from an appropriate simulation environment [4,5] or a benchmarking prototype [7,9,55]. It is based on minimal assumptions for the data distributions, irrespective of the analyzed quality metrics and the architecture characteristics that are taken into account. We believe that the method is applicable to all architecture patterns where the investigated metrics matter and that it is effective in any architecture with interacting quality metrics. However, the latter remains to be confirmed in future research work.

Section 2 describes the concerns of architecture tradeoff analysis and the published related work. Section 3 provides the process of the proposed method. Section 4 introduces the metrics of interest in transactional client–server architecture, as well as a synthetic transaction processing workload that utilizes the servers to a considerable extent. Section 5 shows how the first stage of the method is performed. Section 6 shows the application of ANOVA for detecting the architecture characteristics that are critical for the investigated quality metrics. The paper concludes with a brief discussion on the benefits of the method and the future research prospects.

## 2. Architecture tradeoff analysis and related work

In a software project, the architecture is determined in the early design phase [23,33,48], where the cost to fix an error is orders of magnitude less than in later development phases [54]. An *architecture validation process* [21] evaluates design decisions with respect to possibly prioritized and conflicting quality requirements. Fulfillment of a quality requirement has positive or negative effects on other metrics, due to interactions between them. Design decisions that violate a quality requirement are identified as architectural *risks*, while those that improve quality metrics without violating a requirement are identified as "good" decisions [22]. A decision is considered as an architectural risk or an improvement, based on assumptions for the impact on the system's behavior. For example, in a Process Coordinator architecture, the designer assumes deterioration of throughput when increasing the degree of process distribution.

In related bibliography, there are attempts (e.g. [11,19]) to tabulate the effects of the different metrics on each other. Most of these tables are developed by logical reasoning and experience, rather than existing evidence on the metrics interaction in some specific architecture. In [45], the authors argue that metric dependencies are a property of the architecture. Reliability and performance, for example, can be in conflict in a design that supports a checksums and retry pattern, but they are simultaneously optimized in a replication-based architecture.

Quantitative techniques should address the fact that metric dependencies are architecture-specific and should provide *evidence* for the existing tradeoffs. They should be general enough in order to be *applicable in diverse architectures*, without being constrained by the specific quality metrics that are analyzed simultaneously. Finally, it is desirable to attribute the variability of the metrics to specific architecture characteristics. This allows identifying what in ATAM [14] is called a sensitivity point or a tradeoff point. A *sensitivity point* represents a key architectural decision, which can be a property of one or more components (and/or component relationships [15]), that is *critical* for fulfilling a quality requirement. A *tradeoff point* is a property that affects multiple quality metrics and in fact represents one of the most critical decisions in an architecture design.

Predictable Assembly from Certifiable Components (PACC) [39] is a framework for the evaluation of the ability of a system to meet quality requirements. It supports quantitative evaluation for different metrics, provided that an appropriate analytic theory is available in a form called reasoning framework. Our proposal aims to identify sensitivity and tradeoff points rather than building a

**Table 1**
Characteristics of quantitative techniques for architecture tradeoff analysis.

| Authors, year, reference | Litoiu et al. (2000) [32] | Paul et al. (2003) [44] | Katsaros et al. (2007) [25] |
|---|---|---|---|
| Scope of quantitative analysis | Distributed application systems | Checkpointing and recovery protocols | Distributed systems with independent checkpointing activities |
| Tradeoff points | Process replication or threading levels and process activation policies | Checkpoint intervals | Checkpoint intervals for the independent checkpointing activities |
| Quality metrics | Utilization of processes and devices | Overhead due to checkpointing and recovery activities and quality of recovery | Response times for the fault-affected and the fault-unaffected computations |
| Aim of quantitative analysis | To determine process replication or threading levels, in order to avoid unnecessarily queuing delays for clients and unnecessarily high consumption of memory | To assess the protocols' performance in different execution environments | To determine checkpoint intervals fulfilling the response-time goals at the lowest possible fault-tolerance cost. Trades the gains of a potential improvement in the quality of recovery against the overhead caused in normal processing |
| Analysis approach | Hybrid mathematical programming and analytic evaluation. | Simulation-based measurement. | Simulation-based statistical analysis. |
| Effects of architecture characteristics on the quality metrics | It only takes into account the architecture characteristics included in the model. Estimations are produced only for the tradeoff points | Implicitly takes into account all simulated architecture characteristics | Implicitly takes into account all simulated architecture characteristics. Detailed estimations are produced only for the tradeoff points |