Challenges on the Relationship between Architectural Patterns and Quality Attributes

Conference Paper · April 2017

DOI: 10.1109/ICSA.2017.19

Patricia Lago

Vrije Universiteit Amsterdam, Amsterdam, Net…

218 PUBLICATIONS **2,457** CITATIONS

Giuseppe Procaccianti

VU University Amsterdam

35 PUBLICATIONS **159** CITATIONS

Challenges on the Relationship between Architectural Patterns and Quality Attributes

Gianantonio Me*†, Patricia Lago†, Giuseppe Procaccianti†
*Universidad de Castilla-La Mancha, Spain - gianantonio.me@uclm.es
†Vrije Universiteit Amsterdam, The Netherlands - {g.me, p.lago, g.procaccianti}@vu.nl

***Abstract*—Among other knowledge, software architecture design decision-making relies on the relation between architectural patterns and quality attributes (QAs). However, this relation is often implicit, or in the best case informally and partially defined. This leads to sub-optimal understanding of the impact of the architecture design on the desired level of quality.**

In this work, we aim to shed light on the relation patterns-QAs in the context of an important architectural mechanism, architectural tactics. *Tactics* are design decisions that address a specific quality attribute. In turn, the implementation of a tactic has a different impact according to the used pattern.

From a previous systematic literature review, we selected and analyzed 13 primary studies with a clear focus on tactics. From our analysis, we extracted three overarching challenges on the relationship patterns-QAs that are yet unsolved by research.

The essence of these challenges suggests that further research is needed to identify a clear and precise link between the functional nature of architectural elements and non-functional quality properties.

I. Introduction

Software architecture patterns address recurring design problems in specific design contexts and offer a solution to them [1]. A pattern is a *solution schema* where contradictory forces are balanced by a structure (components, their responsibilities and relationships) and run-time behavior [1]. Quality Attributes (QAs) are measurable properties that identify how well the system satisfies stakeholders' needs [2]. Patterns show a clear relation with QAs, as the patterns adopted in a system influence the balance of QAs achieved by the system itself. Unfortunately, this balance is often implicit, or in the best case informally and partially defined. This leads to sub-optimal understanding of the actual impact of the architecture design on the desired level of quality.

This relation between patterns and QAs is the focus of our exploration and analysis. In a previously conducted Systematic Literature Review (SLR [3]) we categorized and labeled the types of relation as reported in literature. Among them, patterns-QAs relations appear addressed by specific *mechanisms*: in this paper, we analyze the studies that focus on the mechanism of architectural tactics, i.e. design decisions that address a specific quality attribute [2]. First, we extract from the studies the concrete design problems where the relation patterns-QAs plays a crucial role. Then, we frame this relation in architectural approaches. Finally, we extract and discuss overarching challenges on the patterns-QAs relationship that we consider relevant for software architecture research.

This article is organized as follows: Section II explains the basic concepts of our model for the relation patterns-QAs and respective classification. We then identify open challenges in Section III and provide further reflection in Section IV. Section V concludes the paper.

II. Concepts

In our previous SLR [3], we identified three main conceptual areas at the basis of pattern-based quality-driven architecting: a) relation between patterns and QAs; b) approaches that characterize the overall process of architecting; and c) related problems. Among the SLR results it emerged that the relation between patterns and QAs ranges from *undetermined* (e.g. not defined, implicit) to tackled in a precise *mechanism*, i.e. a set of rules and associated knowledge that support a decision-making process [4] related to patterns and quality.

In this work, we selected the 13 primary studies from our SLR that focus on how the mechanism of architectural tactics covers the relation between patterns and QAs. Table I lists the studies according to the following template:

- **Problem:** all primary studies address a specific problem. Defining it helps understand the role of the mechanism (i.e. tactics) in the architecting process.
- **Approach:** how the problem is addressed in the study. Approaches are further categorized as follows:
  - *Decision-Making*: with focus on providing knowledge for taking architectural decisions.
  - *Knowledge-Based*: with focus on how to create a body of knowledge for supporting architectural design. Differently from the previous category, the focus here is more on how to extract that knowledge or on the type of knowledge rather than on decisions.
  - *Quantitative-Prediction-Formal Model*: this category includes models where a quantitative measure is given to quality attributes/tactics in order to choose among patterns on the basis of rating, scores, etc.
  - *Specific Technique*: this category is related to studies that use patterns and tactics in the context of specific techniques i.e. architecture recovery.
- **Challenge:** we provide a *thematic analysis* of the primary studies where we elicit the common challenges for the field that emerge among them.

III. Challenges

Within the studies discussing tactics we identified three broad challenges on the relationship between architectural patterns and quality attributes (see Table I). This section introduces these challenges. We also provide a short summary of how each study highlights each challenge.

Ref	Problem	Approach	Challenge
[5]	Facing all stakeholders concerns	Decision-Making	Decision-Making for Pattern Adoption
[6]	Using Tactics for Pattern Adoption	Decision-Making	
[7]	Using Tactics for Pattern Adoption	Decision-Making	
[8]	Concurrency Patterns for multi-core systems	Decision-Making	
[9]	Incorporating Tactics in a Pattern	Decision-Making	Combining Patterns and Tactics
[10]	Architecture as a collection of patterns	Decision-Making	
[11]	How to elicit Security patterns and tactics	Decision-Making	
[12]	Embedding tactics in specific patterns	Decision-Making	
[13]	Tactic-Equipped patterns	Decision-Making	
[14]	Architecture recovery supported by tactics	Specific Technique	
[15]	Empirical validation of patterns	Knowledge-Based	Quantification and Measurability
[16]	Quantitative assessments of tactics	Quantitative-Prediction-Formal	
[17]	Ranking Patterns	Decision-Making	

TABLE I
Overview of primary studies

A. *Decision-making for Pattern Adoption*

This challenge emerges during the architecting process: as architects have to make design decisions related to the adoption of a specific pattern (or collection of patterns), the lack of a direct and explicit relationship between the patterns and the related QAs makes this process difficult and error-prone. Literature barely touched upon this challenge. In particular:

- Elahi & Babamir [5] aim to build a holistic method for architectural decision-making considering all stakeholders' quality concerns. The problem addressed in this paper is how to select appropriate architectural patterns and tactics according to the desired quality attributes to achieve. In the paper, the relation patterns-QAs is made explicit through the concept of "impact size" that represents the appropriateness of the change introduced by the tactic implementation with respect to the patterns. For instance, in the case of the pattern Pipes&Filters, if a stakeholder concern is to increase computational performance, this can be done by improving the algorithm in a Filter as a specific Performance tactic. However, due to this change, a component (Filter) changes in its structure, which may have a negative impact on the architectural pattern.
- Harrison et al. [6] carried out an experiment on the usefulness of fault tolerance tactics in adopting/changing patterns in a pre-existing architecture. In the experiment, two teams of architects were asked to take decisions on pattern adoption: one team had information about the interaction of tactics and architecture patterns, the other did not. While the team using the information had better results than the other, the work focuses on a single QA, namely pattern combinations impacting fault tolerance. Even within this limitation, this result shows that the tactics knowledge (including fault tolerance measures) helps improving architecture design.
- In a follow-up study, Harrison & Avgeriou [7] developed a model for tactics-patterns interaction. The model should support architects in implementing QAs in a software system. The application of the model leads to several benefits, like the ability of assessing alternative tactics for accomplishing the same quality concern. However, implementation of tactics can be a hard or easy task depending on the specific patterns involved: therefore, knowledge about the QA implemented by a tactics is insufficient and should be complemented by other knowledge like the specific pattern elements that realize QA factors.
- Zheng and Harper [8] provide a decision-making approach focused on concurrency patterns for the specific context of multi-core software engineering. In this work the authors mapped the relation patterns-QAs onto specific design problems, creating a problem-oriented guide. Example of concurrency design problems are shared resources and sequence of operations. The main targeted QAs are Performance and Modifiability. Similar to [15], this work considers the relation patterns-QAs indirectly by using relations patterns-tactics and tactics-QAs.

B. *Combining Patterns and Tactics*

When architecting a software system, patterns and tactics often need to be combined. However, as a result, the relationship with the relevant QAs becomes unclear. Representing this relationship when combining tactics and patterns is still an open challenge for the field. In particular:

- Harrison et al. [9] focus on understanding how tactics (again, specifically for fault tolerance) can be implemented in widely known patterns. Tactics show "structural needs" that might be more or less compatible with the structure of the pattern. So, the structure of the pattern can be compatible with a specific tactic; if not, the pattern should be modified in order to implement a given tactic. The implementation of tactics may cause changes in the pattern elements (components and connectors). By adding structural needs to a tactic, the link to its implementing pattern becomes explicit. While still missing, this link can facilitate the analysis of the impact of structural changes on QAs.
- Parvizi-Mosaed et al. [10] aim to automatize the process of pattern selection by building a model composed of two tactics (Ping-Echo and Heartbeat) and two patterns (Microkernel and Pipes&Filters). The authors consider every software architecture as a collection of patterns: a new architecture, hence, is the outcome of a pattern composition method, where the patterns change dynamically due to the tactics implementation. The authors considered it an advantage adopting a pattern-based design method because each subsystem results as based on a pattern developing the associated functionality. In this line of reasoning, systems-of-systems can be seen as a hierarchical organizations of patterns. A hierarchy of patterns

is composed by a root pattern that distributes subsystems. Subsystems are components ran by tactics that assure a desired quality level. The quality level of the whole system (Root pattern) is the sum of each subsystem. So the relation patterns-QAs is the result of a composition of patterns implemented by tactics.

- Ryoo et al. [11] provide a decision-making approach where tactics for security are organized in a 'security tactic taxonomy', a categorization for inferring systems' security characteristics. The role of the tactics taxonomy is to build up a body of knowledge aiding architects' decision-making. The focus is on a specific QA, security. According to the authors, identification of architectural patterns that are resilient or vulnerable to security attacks is possible by combining the results of three steps, which identify (1) insecure patterns (by analyzing bug reports), (2) secure patterns (patterns widely known as secure) and (3) security tactics. Identification of security tactics makes the relation patterns-security more explicit: however, tactics focus on a single QA, whereas patterns usually address multiple QAs. When combining the two, the resulting impact on the target QA (in this case, security) is still undetermined.
- Salama & Bahsoon [12] show how to embody tactics in self-aware patterns, i.e. tactics implementations that are self-implemented by the system itself. In order to realize tactics in self-aware patterns, the authors followed the notion of quality scenario presented in [2]. A self-aware pattern is equipped with a "Monitor" of Quality of Services, i.e. a component able to detect at run-time changes in workload and quality properties. When changes are detected (stimulus) the self-aware pattern (i.e. a pattern component) has the responsibility of selecting, composing and adopting a tactic (selected from a catalogue). Dedicated Monitor components evaluate the response and increase the self-awareness capability. It is interesting to notice that the process of selecting tactics is embedded in the pattern itself, through a mechanism of self-awareness. This mechanism can address multiple tactics according to different self-awareness levels and process flows. That means that the relation patterns-QAs is dynamic, continuously reshaped according to the detected run-time needs of change. This work assumes, however, that the relation patterns-QAs is quantifiable, and specified at the level of individual pattern components.
- Kim et al. [13] propose a decision-making approach where tactics allow a mapping between patterns and QAs: through the specification of tactics and patterns, adopted together, it is possible to build up a body of knowledge for supporting design decisions. Also, the authors introduce the concept of tactics-equipped patterns. The problem is that it is difficult to trace-back which components of a pattern exactly address a specific QA. So in an *emergent pattern*, implementation of tactics make clear the link between a components and its dependent QAs. In other words, tactics should bridge QAs and architectural solutions in a recognizable way. As co-product, Tactics-equipped patterns should also make the system sustainable in terms of accommodating change.
- Solms [14] focuses on architecture recovery supported by tactics. By including QAs, tactics carry the knowledge about which qualities are influenced in the recovery. In this work tactics are analyzed in a large industrial case focused on architecture recovery.
  Architecture recovery is here carried out by analyzing tactics that allow mapping between structural elements and QAs. The main challenge is on how to identify pattern abstractions and tactics in a legacy system. This task can be hardly automated: in the study, the tactics were extracted from the knowledge gathered on components and discussed among the team of architects. Unfortunately, this process does not eliminate uncertainty and the identification of patterns structure and implemented tactics in a legacy system is still a tough task.

C. Quantification and Measurability

As we saw before, making decisions about patterns and QAs requires concrete information about their relationship. A still open related challenge is the quantification of such relationship by means of measurable values that can help architects in assessing impacts and trade-offs throughout the whole lifecycle. In particular:

- Chen et al. [15] assess architecture patterns by performing experiments. Experiments have the goal of validating the shared and established knowledge on patterns. Tactics have an intermediate role in addressing patterns and QAs. The problem is to assess the effectiveness of architecture patterns and tactics by validating recommendation methods through experimental replication. The process of pattern adoption is driven by the knowledge on QAs, shared and communicated among architects by recommendation methods. However, according to the authors, such methods are rarely empirically validated. The proposed solution is to reinforce empirical practice. They designed an experiment where patterns and quality attributes are framed in a matrix. Tactics play an intermediate role, supporting the relation patterns-QAs. This relation works as a double matrix where indirect relations patterns-tactics and tactics-QAs are evaluated. Through those two matrices it is possible to quantify the level of impact patterns-QAs (i.e. the combination of pattern-tactic and tactic-QAs relation). Therefore, the new information extracted increases the value of the recommendations, improving the knowledge base for supporting architectural decisions.
- Kassab et al. [16] consider the relation patterns-QAs as mediated by tactics. The authors propose a method that quantifies the interactions between patterns, QAs and tactics. The authors offer a quantitative measure of *impact size*, with a double matrix patterns-tactics and tactics-QAs in order to determine the relation patterns-QAs.

- Tahmasebipour & Babamir [17] propose a quantitative evaluation of patterns that creates a ranking for prioritizing architecture design decisions. This work adopts the classification of type of change at component level originally presented in [9]. Ranking is determined on the basis of "impact magnitude", i.e. the number of changes of a pattern to implement a tactic. Also here, however, QAs are addressed individually and indirectly via the related tactics.

IV. Reflection

This paper provides an honest and pragmatic discussion on how the quality dilemma has been addressed in the software architecture community, especially in relation to tactics. The three long-standing challenges we identified are not new, yet still unsolved:

- Challenge A points to the fact that architecture decision-making relies on the implicit and unspecified impact of patterns on quality.
- Challenge B is a reality check: modern software systems are combinations of patterns and tactics, but we still do not know how to control the complex resulting impact on software quality.
- Challenge C is the cornerstone of the problem: to quote Lord Kelvin, "*when you can measure what you are speaking about, and express it in numbers, you know something about it*". We still lack quantitative criteria, empirical laws and evidence to support quality-driven decision-making. This strikes as a distinctive flaw of software engineering with respect to other engineering fields.

Several scientific works have tackled these challenges before, on either architectural patterns or tactics. Very few, however, analyzed all elements together. In particular, Harrison & Avgeriou [18] share our perspective that the most significant and often unintended consequences of architectural decisions regard quality properties. To address this problem, they aimed at integrating in patterns the information about their impact on QAs. As a first step they analyzed the impact of a list of widely-known patterns on a set of QAs from the ISO 9126 quality model. As next step towards evaluating the joint impact of pattern compositions, they suggested to study which patterns are often used together. In [19] they also proposed a kind of pattern language capturing QAs among the consequences of adopting a certain pattern. We consider this a promising direction and we suggest to further pursue this line of research.

V. Conclusion

In this paper, we analyzed the relation between architectural patterns and quality attributes when characterized through the well-known mechanism of tactics. Starting from the results of a systematic literature review, we codified for each primary study a design problem and the corresponding approach. We extracted essential knowledge elements that make explicit the relation patterns-QAs with the support of tactics, and we identified three major challenges within this relationship that are still unsolved by research. With this work, we aim at stimulating discussion within the community. The challenges should pinpoint directions for future research in software architecture.

Acknowledgment

This work received funding from the GINSENG project (Ministerio de Economia y Competitividad and Fondo Europeo de Desarrollo Regional FEDER, TIN2015-70259-C2-1-R).

References

[1] F. Buschmann and R. Meunier, *A system of patterns*, 1st ed. Addison-Wesley Publishing Co., 1996.

[2] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley Professional, 2012.

[3] G. Me, C. Calero, and P. Lago, "A long way to quality-driven pattern-based architecting," in *10th European Conference on Software Architecture (ECSA)*. Springer, 2016, pp. 39–54.

[4] J. F. Hoorn, R. Farenhorst, P. Lago, and H. van Vliet, "The lonesome architect," *The Journal of systems and software*, vol. 84, no. 9, pp. 1424–1435, 2011.

[5] A. Elahi and S. M. Babamir, "Evaluating software architectural styles based on quality features through hierarchical analysis and fuzzy integral (FAHP)," in *7th Conference on Information and Knowledge Technology (IKT)*. IEEE, 2015, pp. 1–6.

[6] N. B. Harrison, P. Avgeriou, and U. Zdun, "On the impact of fault tolerance tactics on architecture patterns," in *Proceedings of the 2nd International Workshop on Software Engineering for Resilient Systems*. ACM, 2010, pp. 12–21.

[7] N. B. Harrison and P. Avgeriou, "How do architecture patterns and tactics interact? a model and annotation," *Journal of Systems and Software*, vol. 83, no. 10, pp. 1735–1758, 2010.

[8] J. Zheng and K. E. Harper, "Concurrency design patterns, software quality attributes and their tactics," in *Proceedings of the 3rd International Workshop on Multicore Software Engineering*. ACM, 2010, pp. 40–47.

[9] N. B. Harrison and P. Avgeriou, "Incorporating fault tolerance tactics in software architecture patterns," in *Proceedings of the 2008 RISE/EFTS Joint International Workshop on Software Engineering for Resilient Systems*. ACM, 2008, pp. 9–18.

[10] A. Parvizi-Mosaed, S. Moaven, J. Habibi, and A. Heydarnoori, "Towards a tactic-based evaluation of self-adaptive software architecture availability." in *SEKE*, 2014, pp. 168–173.

[11] J. Ryoo, P. Laplante, and R. Kazman, "In search of architectural patterns for software security," *Computer*, vol. 42, no. 6, pp. 98–103, 2009.

[12] M. Salama and R. Bahsoon, "Quality-driven architectural patterns for self-aware cloud-based software," in *8th International Conference on Cloud Computing*. IEEE, 2015, pp. 844–851.

[13] D.-K. Kim, J. Ryoo, and S. Kim, "Building sustainable software by preemptive architectural design using tactic-equipped patterns," in *Int. Conf. on Availability, Reliability and Security*.

[14] F. Solms, "Experiences with using the systematic method for architecture recovery (symar)," in *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*. ACM, 2013, pp. 170–178.

[15] X. Chen, W. Zhang, P. Liang, and K. He, "A replicated experiment on architecture pattern recommendation based on quality requirements," in *5th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2014, pp. 32–36.

[16] M. Kassab, G. El-Boussaidi, and H. Mili, "A quantitative evaluation of the impact of architectural patterns on quality requirements," in *Software Engineering Research, Management and Applications*. Springer, 2012, pp. 173–184.

[17] S. Tahmasebipour and S. M. Babamir, "Ranking of common architectural styles based on availability, security and performance quality attributes," *Journal of Computing and Security*, vol. 1, pp. 83–93, 2014.

[18] N. B. Harrison and P. Avgeriou, "Leveraging architecture patterns to satisfy quality attributes," in *1st European Conference on Software Architecture, (ECSA)*, vol. 4758, 2007, pp. 263–270.

[19] N. B. Harrison, P. Avgeriou, and U. Zdun, "Using patterns to capture architectural decisions," *IEEE software*, vol. 24, no. 4, pp. 38–45, 2007.