Innovations in Systems and Software Engineering (2018) 14:263–271
https://doi.org/10.1007/s11334-018-0319-4

ORIGINAL PAPER

Software architectural patterns in practice: an empirical study

Mohamad Kassab[1] · **Manuel Mazzara**[2] · **JooYoung Lee**[2] · **Giancarlo Succi**[2]

Received: 21 November 2017 / Accepted: 6 December 2018 / Published online: 12 December 2018

Abstract
Software architecture involves a series of decisions based on many factors in a wide range of software development. Architects face recurring issues in different software architecture design, and to reduce huge cost and risks, software architecture decisions can rely on a set of idiomatic patterns commonly named architectural styles or patterns. Architectural pattern determines the vocabulary of components and connectors that are used in instances of the pattern together with a set of constraints to combine the two. Little contemporary data exists to document actual practices used by software professionals when selecting and incorporating architectural patterns for their projects in industry. Therefore, a comprehensive survey of software professionals was conducted to attempt to discover these practices. This exploratory survey and its quantitative results offer opportunities for further interpretation and comparison. Data from this survey are presented in this paper and include characteristics of projects, practices, organizations, and practitioners related to the usage of architectural patterns. Some of the notable findings include that architectural patterns are widely used in software projects with the Model–View–Controller being the most common. Despite reported difficulties in incorporating architectural patterns, the majority of the software professionals revealed that patterns were the most essential for completing the projects. The most difficult pattern to implement and the most expensive to adopt was the peer-to-peer, while the easiest was the client–server.

Keywords Software architecture · Architectural patterns · Quality attributes · Common practices · Software professionals · Architectural tactics

1 Introduction

The amount and complexity of software in systems is on the rise [1]. Also, software architecture is a rising subject of software engineering to help stakeholders to describe the high-level structure (or structures) of a system [2]. A software architecture is significant in paving the way for software success for many reasons. It is a structure that facilitates the satisfaction to systemic qualities (such as performance, security, availability, modifiability, etc.). It enhances the traceability between the requirements and the technical solutions which reduce risks associated with building the technical solution [3]. It also serves as a vehicle for communication to the stakeholders of the system under consideration and can serve as a basis for large-scale reuse [4].

Defining Software Architecture involves a series of decisions based on many factors in a wide range of software development. These decisions are analogous to load-bearing walls of a building. Once put in place, altering them is extremely difficult and expensive. Therefore, each of these decisions can have considerable impact on satisfying a project's requirements under given constraints [3].

Architects might face recurring issues in different software architecture design. For saving of huge cost and the reduction of risks, software architecture decisions can rely on a set of idiomatic patterns commonly named architectural styles or patterns [5]. A software architectural pattern defines a family of systems in terms of a pattern of structural organization and behavior [6]. More specifically, an architectural pattern determines the vocabulary of components and connectors that can be used in instances of that pattern, together with a set of constraints on how they can be

✉ Mohamad Kassab
muk36@psu.edu

Manuel Mazzara
m.mazzara@innopolis.ru

JooYoung Lee
j.lee@innopolis.ru

Giancarlo Succi
g.succi@innopolis.ru

[1] Pennsylvania State University, Malvern, PA, USA

[2] Innopolis University, Innopolis, Russia

combined. Common architectural patterns include Layers, Pipes-Filters, Model View Controller (MVC), Broker and Client–Server. Architectural patterns have been described using different frameworks (see e.g., [6–11]). A common framework to describe patterns includes the name, the problem, the structure and the dynamics of the solution, and the consequences of using a pattern expressed in terms of its benefits and liabilities. Hence, the selection of an architectural pattern is qualitatively driven by the properties it exhibits [11] and the architect's knowledge and experience with patterns.

Unfortunately, little contemporary data exists to document the actual practices used by software professionals when selecting and incorporating architectural patterns for their projects in industry. Therefore, a comprehensive survey of software professionals was conducted to attempt to discover these practices.

Surveys of software industry professionals are an effective way to determine the current trends in software engineering processes. Survey responses can also help others to understand the relationship between areas such as software quality and using architectural patterns. A carefully constructed survey has the potential to: (1) remedy the deficiency of lack of data and (2) identify the state of practice of using architectural patterns in industry, which can then be disseminated. Based on these two objectives, we designed a survey study on the current software architectural patterns state of practice. The data from this survey are exhibited herein.

The rest of the paper organized as follows: Sect. 2 outlines the existing research in the field. Section 3 describes the survey design and conduct. In Sect. 4, we present statistics regarding the survey participants, their organizations and the surveyed projects. In Sects. 5 and 6, we report on our findings on the criteria for selecting the architectural patterns for a project in practice and the consequences/challenges of incorporating the patterns. In Sect. 7, the validity and the limitations of our findings are discussed. Finally, the conclusions and implications are presented in Sect. 8.

2 Background

Common approaches to architectural design (e.g., [4,12]) propose processes to guide the architect in making the proper architectural decisions such as selecting architectural tactics and patterns that satisfy the most important quality attributes of a software system. Whereas a given tactic focuses on a single quality attribute, a pattern can address multiple quality attributes at the same time.

Many architectural patterns have been proposed and described in the literature (e.g., [6–11]). Architects may rely on techniques to evaluate their architectural decisions to ensure their validity to the intended use before moving on to subsequent design and development stages. One of the popular architecture evaluation techniques is the Architecture Trade-off and Analysis Method (ATAM) [13]. It is a scenario-based questioning technique developed at the Software Engineering Institute at Carnegie Mellon University to assess the consequences of architectural decisions (e.g., selecting architectural tactics and patterns) in light of the business goals and quality attribute requirements.

While ATAM supports a qualitative evaluation of architectural decisions, Bode and Riebisch presented in [11] a quantitative evaluation of a set of selected architectural patterns regarding their support for the evolvability quality attribute. They refined the evolvability attribute into subcharacteristics and related them to some properties that support good design. The selected patterns are used in a case study and the resulting design is assessed to determine the impact of these patterns on these properties.

Another quantitative framework was discussed in [14] and aimed at relating tactics to patterns. The proposed framework concentrated on quantifying the interaction between tactics and patterns, especially how tactic implementations affect patterns. In [15], the authors proposed a complementary quantitative approach as they considered both the impact of a tactic on a pattern and the impact of the tactic on other quality attributes.

A survey study examining the software architecture in practice was conducted in 1995 and presented in [16]. The paper reported the pragmatic and concrete issues at that time related to the role of architecture in the design and development of systems representing a variety of application domains and range in size from small (fewer than 100 KLOC) to very large (more than 1 MLOC). In [17], nonconclusive and localized results are discussed on the usage of software architecture in software industry.

Empirical studies on architectural prototyping [18] as an instrument to investigate stakeholders' concerns with respect to a system under development also exist [19], in particular with emphasis on industrial practices [20].

In spite of the above-mentioned work, there is still a substantial lack of comprehensive descriptions of what are the consequences of the adoption and use of patterns. Most of the existing work in this area is primarily qualitative and not properly replicable [14], therefore, it is very difficult to draw general conclusions based on it.

To our best knowledge, no other up-to-date work was found on analyzing the current state of practice when using software architectural patterns.

3 Survey design and conduct

A web-based survey instrument was created using the web-based QuestionPro survey tool. The survey was designed and conducted according to the guidelines defined in [21],

namely survey design guidelines, subject privacy guidelines, sampling and subject selection guidelines and survey piloting guidelines.

Respondents were asked to base their responses on only one software project that they were either currently involved with or had taken part in during the past five years. Overall, the survey consisted of 19 questions arranged into five sections related to: project information, criteria and challenges in selecting architectural patterns, impact of the selected pattern on the product quality and project success, organization's information and participant's professional information. We posted the details regarding each question and the raw data from this survey through the link: http://goo.gl/0blFMB.

To address the problem of representativeness and to increase the external validity of our findings, following the recommended approaches in the literature [22–24], we drew our survey participants from multiple sources but primarily from members of the following Linked-In professional groups, to which one or more of the authors belonged: "Software Engineering Productivity: Software Architecture", "Techpost Media", "ISMG: Software Architecture", and "ISMG Architecture World". An invitation on these groups was posted under the subject "Software architecture in practice". All responses were treated anonymously and only aggregate data were used—not the individual responses. Survey data were collected from May 2015 through March 2016.

The survey drew 809 participants from 39 countries. Of these survey takers; 126 completed the survey to the end. The completion rate was 15.6% and the average time taken to complete the survey was 10 min. In our analysis for this paper, we also included the results of the partially completed responses. For example, when the text says something like "25% of the participants have chosen pattern A"; this refers to the 25% of the participants whom answered the question; even though some of them may have not completed the survey to the end.

4 Demographics regarding the survey participants and project characteristics

We determined that it was essential to attract as many experienced participants with wide range of projects as possible. This decision was made to reduce an external threat to validity related to interaction of sample selection and treatment [25]. This is the effect of having a subject population not representative of the population (software professionals) that we want to generalize to. The survey participants reflected indeed a diverse range of positions for the surveyed projects with 45% of the participants describing themselves as programmers/developers, 27% as Architects, 18% as Software/System Engineers and 10% as product/project/Scrum managers.

Cross-tabulation analysis linked the responses on participants' positions to responses on which architectural patterns a participant is familiar with at the time of taking the survey. The average number of patterns a participant is familiar with is six patterns. The number goes higher for architects (on average eight patterns), and lower for programmers/developers (on average five patterns). This variation is expected as architects are the people who take responsibility to develop the architecture design and their most important job is to map software requirements to architecture design and guarantee that both functional requirements and quality attributes are met [26]. They need to take architectural decisions including the selection of architectural patterns.

As far as the size of the participating companies is concerned, a representative sample can be determined. Almost 44% of the participants work in small companies (with 1–100 full time employees) and 26% worked in medium size (with 100–1000 full time employees). But also very large companies (with more than 1000 full time employees) are well represented at 30%.

Several questions also asked respondents to classify and categorize the relevant projects. The responses revealed that projects were distributed across a broad range of application domains with small bias toward sales and retail applications (see Fig. 1). We further linked the reported domains to the responses from the question: "Which quality requirements were relevant to the project?" (see Table 1). Surprisingly, "Security" appears not as relevant even where we would expect (e.g., medical systems, aerospace, defense). The results evidence that "Performance" quality requirement is of a concern in several sectors, not only in embedded systems (e.g., Aerospace or Defense) or in areas requiring high interactivity (e.g., Finance and Gaming) as we might have expected, but also in Government, HR, and Marketing, where probably there are "high profile" users of the systems who expect very fast replies. Areas requiring high interactivity (Finance and Gaming) reported fairly high values of "Availability".

When asked which Software Development Life Cycle process best describe the ones used in the project; the majority of the respondents were aware of the SDLCs used. Agile development methodologies (e.g., SCRUM, Extreme Programming, Feature Driven Development, Lean) were selected 48% of the time making them the most popular.

Altogether, qualitatively the respondents appeared distributed in a way similar to the general world population, as sampled on surveys available online, like [27,28].

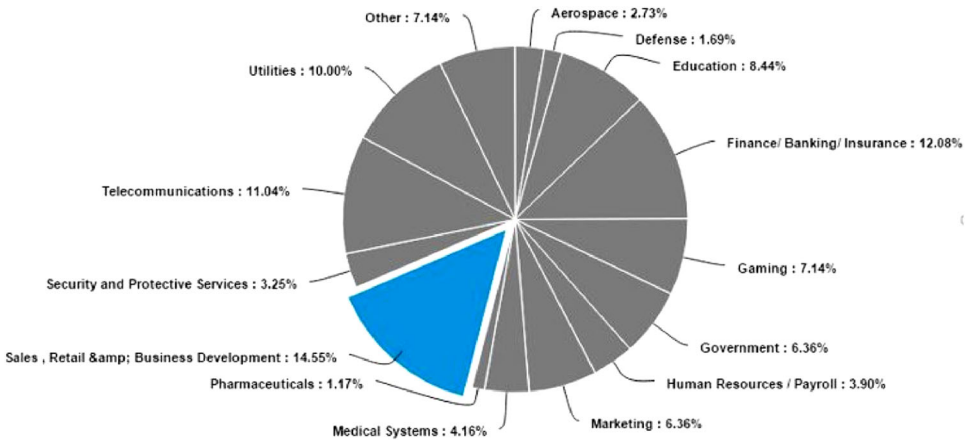

Fig. 1 Which of the following application domains does/did this project apply to? (Check all that apply)

Table 1 Relevant quality requirements per project domains: a value in the table presents% of projects from corresponding domain while concerned about corresponding quality as relevant

	Aerospace (%)	Defense (%)	Education (%)	Finance (%)	Gaming (%)	Government (%)	HR (%)	Marketing (%)	Medical systems (%)	Sales (%)	Utilities (%)
Quality not considered	0	0	22	11	19	14	8	6	3	11	9
Availability	63	25	39	56	43	50	42	38	13	47	59
Interoperability	25	25	17	19	33	32	33	19	40	15	18
Modifiability	100	100	52	53	62	55	83	63	60	56	59
Performance	50	50	39	69	67	64	75	56	40	49	55
Security	50	50	30	44	29	23	50	44	20	29	23
Testability	63	25	22	39	19	27	42	25	20	25	32
Usability	50	50	52	53	43	45	67	56	67	38	36
Other	0	0	0	2	0	0	0	0	13	0	4

5 Criteria for architectural patterns selection

The architecture of a software system is almost never limited to a single architectural pattern [3]. Complex systems exhibit multiple patterns at once. A web-based system might employ a three-tier client–server pattern, but within this pattern it might also use MVC, layering and so forth. This is consistent with our finding from this survey. We asked participants to identify from an extensive list of known architectural patterns, the set of patterns that was selected for their projects. The list of the patterns that was presented to the participants was derived from the representative catalog of patterns which are overviewed in [4]. On average, a respondent indicated that 3 architectural patterns were selected to build the structure of a project. The data (see Fig. 2) revealed that "MVC" was the most common pattern selected (69% of participants reported using this pattern). Client–Server was second at 58% and both layered pattern and Service Oriented Architecture (SOA) hold the third place (30% of participants reported using each of these patterns). Only 3 out of the total 262 whom responded to the question reported that no pattern was actually used for the project.

The literature on software architecture suggests that one of the primary purposes of the architecture of a system is to create a system design to satisfy the quality attributes [29]. Quality is the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs [30]. Software quality is an essential and distinguishing attribute of the final product. Typically, systems have multiple important quality attributes (e.g., Performance, Modifiability, Security, Testability, etc.) and decisions made to satisfy a particular quality may help or hinder the achievement of another quality attribute. The architecture is the first place in software creation in which quality requirements should be addressed. In fact, if functionality was the only thing that mattered, we would not have to divide the system into pieces at all, and we would not need to have an architecture [4]. The motivation for the selection of the appropriate patterns should depend mostly on a set of architectural drivers (mostly quality requirements) which possess the highest business and technical priority to the system. But what are the actual criteria in practice?

Surprisingly, the results from the survey (see Fig. 3) reflected that; in practice, the quality requirements are not the

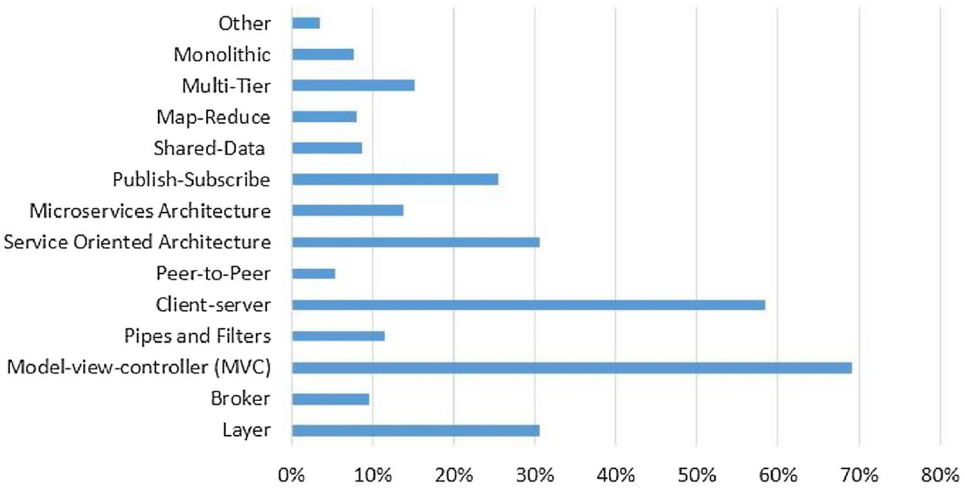

Fig. 2 Which architectural styles (patterns) were chosen for the selected project? Check all that apply

dominant factor when deciding on which patterns to select. In fact, it is the functionality that is taking the front seat. When asked to select which criteria were the most important in choosing the selected architectural patterns for their projects; participants chose functionality 72% of the time, then technology constraints 57% of the time, then quality requirements 49% of the time. On average, the decision on selecting a pattern was based on three criteria at once.

Wrong architectural choices can cost significant time and effort in later development, or may cause the system to fail to meet its quality attribute goals [29]. Out of those who selected quality as a criteria for their patterns selection, 62% reported an agreement (either agree or strongly agree) with the statement that the chosen architectural patterns were sufficient to achieve project goals. The agreement among the group whom did not select the quality as a criteria was only at 43%. This indicates a need to improve the current short-sighted practice when selecting architectural patterns to consider qualities as criteria of selection more often.

The group who selected quality requirements as a criteria was further asked to identify all quality requirements which relevant to the project and the patterns' selection. We presented a list of predominant qualities [4] which a participant may select from, but the option of "Others" was also provided. On average, three quality requirements were considered for a project—whenever qualities were considered. The results (see Fig. 4) show that "Performance" was the most considered quality overall (selected 56% of the time).

When the participants were asked to report on the main challenges in selecting the architectural patterns for their projects in an open text question, around 50% of the participants reported that the main challenge is the continuous changes of requirements/environment.

6 Incorporating architectural patterns in practice

During architectural design, an architect may select one or more architecture patterns in order to produce the initial system structure. The architect selects patterns based on their ability to support the majority of the requirements of the system. Patterns embody the high-level structure of the system. Nevertheless, selected patterns (in their standard structures) may not support the entire requirements list. In order to address the remaining quality requirements that are not prompted by the selected patterns and in order to handle trade-off analysis, architectural tactics can then be incorporated into the higher structure to address particular qualities locally.

Tactics are a special type of operationalization that serves as the meeting point between the quality attributes and the software architecture. In [15], the authors define an architectural tactic as an architectural transformation that affects the parameters of an underlying quality attribute model. "The structure and behavior of tactics is more local and low level than the architectural pattern and therefore must fit into the larger structure and behavior of patterns applied to the same system" [5]. Implementing a certain tactic within a pattern may affect the pattern by modifying some of its components, adding some components and connectors, or replicating components and connectors [5]. This may yield to a final architectural structure that differs from the initial chosen pattern structure.

In our survey, we asked the participants to identify to what degree does/did the final design of the project adhere to the chosen architectural styles (patterns). Even though, 59% reported that a level of agreement (either agree or strongly agree) that the chosen architectural patterns (styles) were easy to implement and 66% reported a level of agreement that the chosen architectural patterns (styles) were sufficient

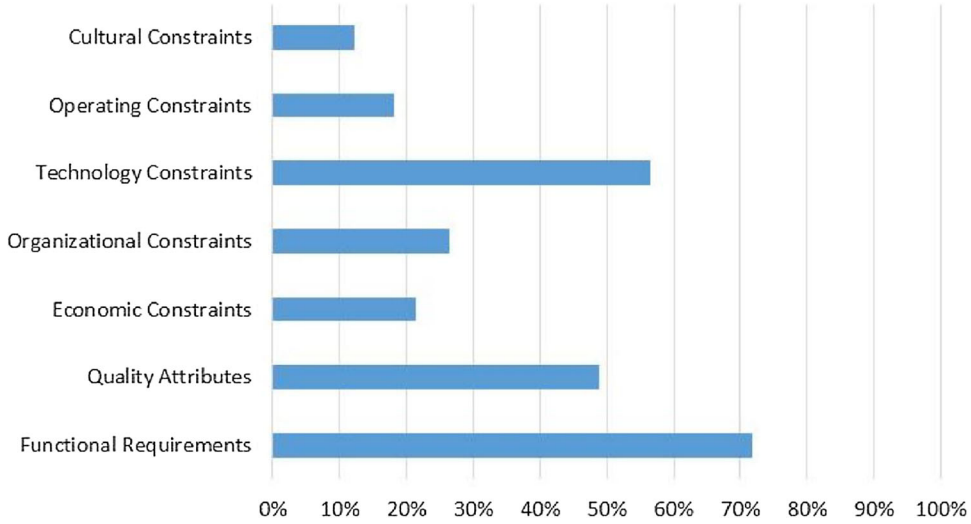

Fig. 3 Which criteria were the most important in choosing the selected architectural styles (patterns) for the project?

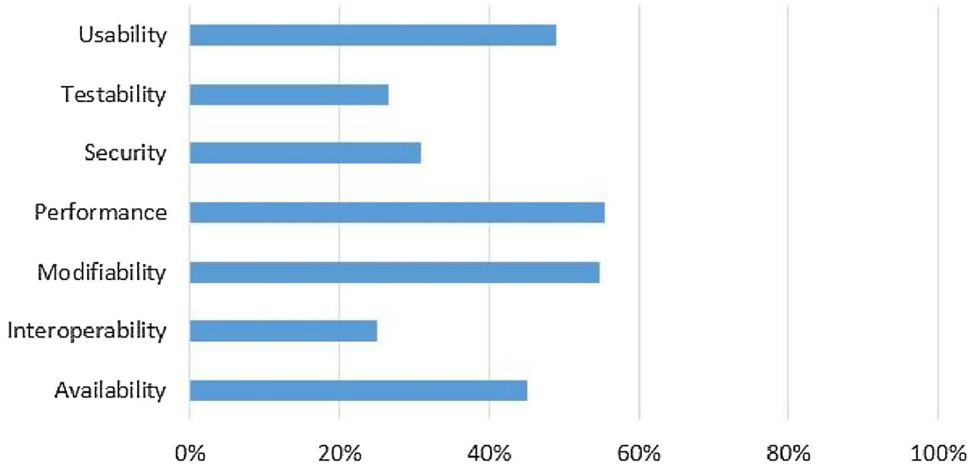

Fig. 4 If quality requirements are a criteria considered for choosing an architectural style, which of these quality requirements are more relevant to the project?

to achieve project goals, the majority (63%) reported that the final design implements the patterns with few changes. Twenty-two percent reported that that the final design implements the patterns with major changes. Only 9% reported that the final design implements the patterns with no changes.

Only 24% reported disagreement with the statement that "It was not possible to complete the project without relying on the architectural patterns (styles)". The most selected aspect related to architectural patterns used for the project, participants were not satisfied with was that "it takes a lot of time to implement" (see Fig. 5). We further linked these aspects to the individual patterns. Table 2 reports on our findings analyzing this link.

7 Validity of the results

We carefully examined our study for the possible types threats to validity described in [25,31,32]. One possible internal threat to validity that we identified is related to the instrumentation. This is the effect caused by the artifacts (e.g., survey questions) if these are badly designed. We sent out the link to the survey to a number of researchers to collect their feedback before the data collection phase started. We addressed the feedback toward improving the quality of the questions. We implemented the suggestions when applicable. For example, in the question on what architectural patterns selected for a project, a clear definition was provided next to each of the possible answers a participant may select from.

Another possible internal threat is related to the morality. This is the effect due to the different kinds of persons who drop out from the survey. We carefully examined the sample whom dropped out in regard to three participants characteristics: job role, education level and years of experiences. We observed that the drop out sample was representative to the total sample. We could not relate the dropout to a particular participants characteristic. In addition, we examined the results from the perceptions of managerial job roles and non-managerial roles. No significant differences on the results came from these two samples.

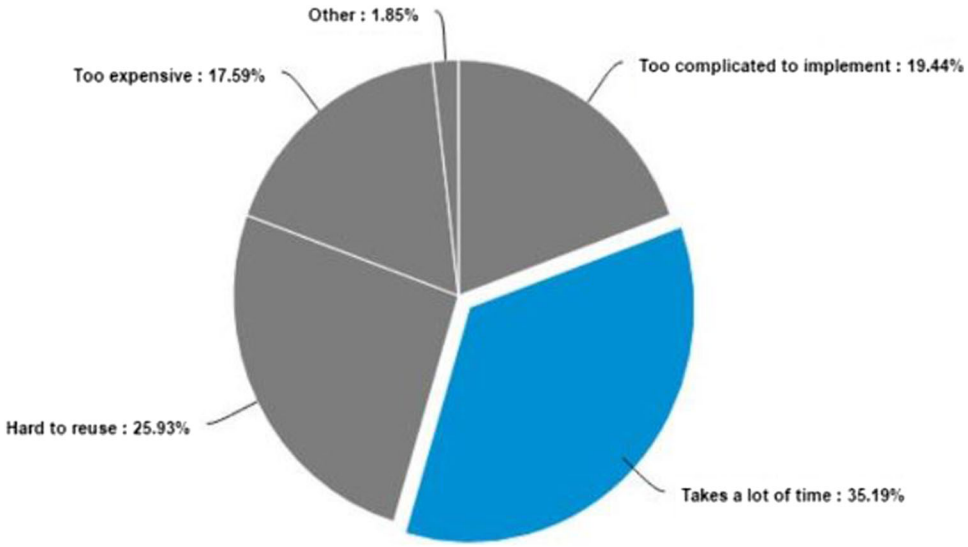

Fig. 5 Problems in using architectural patterns

Table 2 Most and least reported problems for architectural patters

	Most reported	Least reported
Too complicated to implement	Peer-to-peer	Client–server
Takes a lot of time	Multi-tier	Peer-to-peer/shared data
Hard to reuse	Broker/monolithic	Multi-tier
Too expensive	Peer-to-peer	Multi-tier

A third aspect of internal validity we examined was related to history. This is related to different results may be obtained by the same sample at different times. To address this, we run our data collection phase into two rounds. The first round lasted from May 2015 to October 2015. The second round lasted from November 2015 to March 2016. We followed the same recruiting strategy in each round. If we break the results into two samples to correspond to the two rounds, we also observe no significant differences on the findings between the two samples.

On the external validity threats, we examined the possible threat related to interaction of selection and treatment. This is an effect of having a subject population, not representative of the population we want to generalize to. Judging from the participants characteristics we presented in Sect. 4, this was not the case. A second possible external validity threat may be related to the interaction of history and treatment. For example, if the survey is taken a few days after a big software-related crash, people tend to answer differently than few days before. To address this threat, the two data collection phases were spanned over relatively a long period (9 months).

Lastly, it is worth saying that all the studies like this require replications and confirmatory studies, especially in software engineering. To facilitate the replication of our study, we posted the survey questions from this survey through the link http://goo.gl/0blFMB. Researchers and practitioners are welcome to execute further analysis on the data.

The authors are available to offer the original text of the questionnaire to any scientist interested in replication.

8 Implications and conclusions

There is a lack of information on the state of practice in the usage of software architectural patterns. In this study, we collected 809 partial or complete responses from developers from 39 different countries using an online questionnaire that has been designed according to the best practices and the most rigorous criteria defined for empirical investigations.

The results from this study reported that the most impelling reason for selecting patterns is the functionality. This is in contrast with what the literature on software architecture suggests that the primary criteria for pattern's selection should be related to the satisfaction of the quality attributes. In related finding from this study, a higher satisfaction was reported with how the chosen patterns were sufficient in achieving project goals in the sample that considered quality attributes as a criteria for patterns' selection in comparison with the sample that did not consider the qualities. The implication from these findings is that there is a need to improve the current practice when selecting software architectural patterns to include quality attributes as a selection criteria.

Other notable findings included:

- Architectural patterns are widely used in software projects. The majority of the software professionals also thought that without patterns it would have not been possible to complete the projects they were involved. Nevertheless, most of the patterns were applied with changes (by incorporating tactics). Both patterns and tactics are used hand-in-hand when constructing an architectural structure to achieve a comprehensive satisfaction of the architectural drivers.
- Even though the most impelling reason for selecting patterns is the functionality, software professionals concerned for quality nearly always incorporate patterns. Performance was the first quality requirement selected by software professionals concerned in quality, the second being modifiability, and the third usability. It was surprising not to see security selected when it is expected (e.g., medical systems, aerospace, defense). This finding deserves a further investigation.
- The most significant difficulty in adopting patterns in practice is the continuous changes of user requirements or of the environment. This finding suggests a better harmonization of the requirements management activities and software architecture process.

We hope that this survey and corresponding results stimulate research into prevailing software practices. Moreover, we intend these results to highlight the areas of software architecture, software quality management, and software project management that need the attention of both the research community and the industry professional. We plan also in the future to perform a replication of the study to increase the solidity of our findings.

References

1. Charette RN (2009) This car runs on code. IEEE Spectr 46(3):3
2. Aad G, Abbott B, Abdallah J, Abdelalim A, Abdesselam A, Abdinov O, Abi B, Abolins M, Abramowicz H, Abreu H et al (2010) The ATLAS simulation infrastructure. Eur Phys J C 70(3):823–874
3. Microsoft Patterns & Practices Team (2009) Microsoft application architecture guide, 2nd edn. Microsoft Press
4. Bass RKL, Clements P (2013) Software architecture in practice, 3rd edn. Addison-Wesley, Reading
5. Kassab M, El-Boussaidi G, Mili H (2012) A quantitative evaluation of the impact of architectural patterns on quality requirements. In: Lee R (ed) Software engineering research, management and applications. Springer, Berlin, pp 173–487
6. Garlan D, Shaw M (1993) An introduction to software architecture. In: Advances in software engineering and knowledge engineering, vol I. World Scientic Publishing Company
7. Shaw M, Garlan D (1996) Software architecture: perspectives on an emerging discipline, 1st edn. Prentice Hall, Englewood Cliffs
8. Buschmann F, Henney K, Schmidt DC (2007) Pattern-oriented software architecture, on patterns and pattern languages, 5th edn. Wiley, New York
9. Avgeriou P, Zdun U (2005) Modeling architectural patterns using architectural primitives. In: ACM SIGPLAN Notices, vol 40, No 10. ACM, pp 133–146
10. Gamma E (1995) Design patterns: elements of reusable object-oriented software. Pearson Education India, Delhi
11. Bode S, Riebisch M (2010) Impact evaluation for quality-oriented architectural decisions regarding evolvability. In: European conference on software architecture. Springer, Berlin, pp 182–197
12. Wood WG (2007) A practical example of applying attribute-driven design (add), version 2.0. Technical report, DTIC Document
13. Paul C, Kazman R, Klein M (2002) Evaluating software architectures: methods and case studies. Addison-Wesley Professional
14. Harrison NB, Avgeriou P, Zdun U (2010) Incorporating fault tolerance tactics in software architecture patterns. In: Proceedings of the 2nd international workshop on software engineering for resilient systems (ACM), pp 12–21
15. Kassab M, El-Boussaidi G (2013) Towards quantifying quality, tactics and architectural patterns interactions. In: The 25th international conference on software engineering and knowledge engineering, Boston, 510 MA, USA, June 27–29, pp 441–446
16. Soni D, Nord RL, Hofmeister C (1995) Software architecture in industrial applications. In: Proceedings of the 17th international conference on software engineering, ICSE '95. ACM, New York, pp 196–207
17. Gardazi SU, Shahid AA (2009) Survey of software architecture description and usage in software industry of Pakistan. In: 2009 international conference on emerging technologies, pp 395–402
18. Bardram JE, Christensen HB, Hansen KM (2004) Architectural prototyping: an approach for grounding architectural design and learning. In: Proceedings. Fourth working IEEE/IFIP conference on software architecture (WICSA 2004), pp 15–24
19. Christensen HB, Hansen KM (2010) An empirical investigation of architectural prototyping. J Syst Softw 83(1):133–142
20. Christensen HB, Hansen KM (2008) Architectural prototyping in industrial practice. Springer, Berlin, pp 196–209
21. Andrews D, Nonnecke B, Preece J (2007) Conducting research on the internet: online survey design, development and implementation guidelines. https://auspace.athabascau.ca/handle/2149/1336
22. Szolnoki G, Hoffmann D (2013) Online, face-to-face and telephone surveys–comparing different sampling methods in wine consumer research. Wine Econ Policy 2(2):57–66
23. Khazaal Y, van Singer M, Chatton A, Achab S, Zullino D, Rothen S, Khan R, Billieux J, Thorens G (2014) Does self-selection affect samples' representativeness in online surveys? An investigation in online video game research. J Med Internet Res 16(7):e164
24. Maalej W, Tiarks R, Roehm T, Koschke R (2014) On the comprehension of program comprehension. ACM Trans Softw Eng Methodol 23(4):31:1–31:37
25. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer, Berlin
26. Qian K (2010) Software architecture and design illuminated. Jones & Bartlett Learning, Burlington
27. Developer Survey Results—2016. Technical report, StackOverflow (2016). http://stackoverflow.com/research/developer-survey-2016. Visited 26 Feb 2017
28. Wilcox M, Schuermans S, Voskoglou C (2016) Developer economics, state of the developer nation, Q3 2016. Technical report, VisionMobile Ltd
29. Harrison NB, Avgeriou P (2007) Leveraging architecture patterns to satisfy quality attributes. In: European conference on software architecture. Springer, Berlin, pp 263–270
30. ISO I (2001) Software engineering—Product quality—Part 1

31. Campbell DT, Stanley JC (2015) Experimental and quasi-experimental designs for research. Ravenio Books, San Francisco
32. Hyman R (1982) Quasi-experimentation: design and analysis issues for field settings (book). J Personal Assess 46(1):96–97