Trade-offs evaluation in pattern-based product migration: A Chilean case study

PABLO CRUZ, Universidad Técnica Federico Santa María, Valparaíso, Chile.
HERNÁN ASTUDILLO, Universidad Técnica Federico Santa María, Valparaíso, Chile.

Design patterns are well-known as reusable solutions to recurring design problems, and their use in application migration projects brings about multiple patterns. However, there is little systematic guidance to discover and analyze patterns trade-offs, and specially so when involving case-specific business drivers. This article reports the use of Architectural Trade-offs Analysis Method (ATAM) to identify and assess patterns trade-offs in the migration of a 20 year-old payroll product maintained by a small Chilean company. The company's staff reported that the approach allowed them to systematically approach trade-off analysis and made them confident that their design decisions would actually be aligned to their business drivers. We exemplify the approach with two microservices-related patterns, namely API Gateway and Saga.

Categories and Subject Descriptors: D.2.11 [**Software Engineering**]: Software Architectures—*Patterns*

General Terms: Software architecture

Additional Key Words and Phrases: Software architecture, software architecture evaluation, design decisions, software engineering

ACM Reference Format:

Cruz, P. and Astudillo, H. 2018. Trade-offs evaluation in pattern-based product migration: A Chilean case study. HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. V (January YY), 8 pages.

1. INTRODUCTION

Over time, the architecture of a software system suffers many changes, which are motivated by new requirements, changes in market conditions and laws, or by the simple company's intention to succeed in the long term. A *legacy system* as one exhibiting two characteristics: (1) it is critical for the business [1] and (2) significantly resists modification and evolution [2].

This article presents the design and results of evaluating the software architecture of a 20-year-old payroll and human resources system, developed and maintained by a Chilean company. The evaluation was done to (1) assess the fitness of the current architecture, and (2) find well-supported evidence and guidance for the system future evolution. A specific trigger for the evaluation was the customer interest in migrating the legacy architecture to a microservice-based one, since it seemed well suited for its business.

For the architecture evaluation, we used the well-known Architectural Trade-off Analysis Method (ATAM) [9]. Although there are many methods for evaluating architecture, we used it because it focuses on analyzing and discussing trade-offs between diverse quality attributes. Indeed, the customer staff reported that the approach allowed them to systematically approach the elicitation and analysis of trade-offs related to the decisions that emerged as

This work is supported by DGIP-USM and by ANID under grant ANID PIA/APOYO AFB180002.
Author's address: P. Cruz, Avenida España 1680, Valparaíso, Chile; email: pcruz@inf.utfsm.cl; H. Astudillo Avenida España 1680, Valparaíso, Chile; email: hernan@inf.utfsm.cl

SLPLoP'18, NOVEMBER 20-23, Valparaíso, Chile. Copyright 2018 is held by the author(s). HILLSIDE 978-1-941652-11-4

a consequence of applying patterns. In particular, we present here the results regarding two microservice-related patterns: API Gateway [3] and Saga [4][5].

The reminder of the paper is structured as follows: Section 2 briefly summarizes ATAM and its steps; Section 3 describes our particular experience using ATAM to evaluate this system; Section 4 presents related work; and Section 5 summarizes and concludes.

2. THE ARCHITECTURAL TRADE-OFF ANALYSIS METHOD (ATAM)

Architectural evaluation has been used for software migration [6]. ATAM is a well-known architectural evaluation method; one of its key characteristics is allowing to explore how trade-offs among architectural decisions impact the system quality goals [7]. ATAM has been used with newly developed systems as well as legacy ones [9].

ATAM has nine steps, grouped into four categories: Presentation, Investigation and Analysis, Testing, and Reporting.

The Presentation steps are:

(1) Present the Method: the evaluation leader present ATAM and the steps that will be executed in the complete journey, remarking that the method does not necessarily will be enacted in a strict linear fashion (i.e., at times, some steps seem to be overlapped with others).
(2) Present the business drivers: a business-related person describes what business goals are guiding the development efforts.
(3) Present the architecture: the product's architect describes the architecture, highlighting how this one intends to address the business drivers.

The Investigation and Analysis steps are:

(4) Identify the architectural approaches: As noted in [7], architectural approaches is a more general term than styles that will allow architects not accustomed with the style term to discuss the identify the many set of decisions that are available.
(5) Generate the quality attribute tree: by generating a utility tree, the team can elicit quality attributes specified down to the level of scenarios. The scenarios are also prioritized.
(6) Analyze the architectural approaches: the architectural approaches are now analyzed regarding the scenarios identified in the utility tree. It is in this step in which risks, nonrisks, sensitivity points and trade-off points are identified.

The Testing steps are:

(7) Brainstorm and prioritize scenarios: in this step, more scenarios are elicited, but this time considering a larger group of stakeholders. A voting process is used to prioritize the scenarios.
(8) Analyze the architectural approaches: step 6 appears as a reiteration in this step in the sense that the analysis of architectural approaches is again done, but using the new set of high-priority scenarios.

Finally, the Reporting step is:

(9) Present the results: in this final step, the evaluation team presents the results of the process to the stakeholders.

The main ATAM outputs are *approaches*, *scenarios*, *attribute-specific questions*, *utility tree*, *risks and non-risks*, *sensitivity points*, and *trade-offs* [9]. We remark that, since the evaluation was done in the context of a migration, it also yield many other outputs, like architecture models and components and their relations.

3. THE ARCHITECTURE EVALUATION

In this section, we describe the concrete experience at a Chilean company.

3.1 Evaluation Team

The evaluation team was composed of 8 persons, 6 from the company and 2 external consultants. Table I shows the roles involved in enacting ATAM in the company and table II shows the mapping between the days used for the evaluation and the ATAM's steps.

Table I. Main roles in the ATAM evaluation team.

Role	Number
Chief executive officer	1
Chief architect	1
Architect	1
Lead developer	1
User consultant	2
External consultant	2

As user consultants were the most restricted in terms of available time for evaluation (they spent most of the time attending company's clients), they were not present at every ATAM step. However, they were both present in the scenario brainstorming which is directly related to understanding all stakeholders needs.

3.2 ATAM Sessions

3.2.1 *First Day: Presentation of ATAM, Business Drivers and Architecture.* In our case, and in line with other reports, we remark that the method was not enacted in a strictly linear manner (i.e., some steps may appear to be overlap with others).

We scheduled and carried out three evaluation sessions (three days) of about 6-7 hours each one[1]. Due to time and scheduling constrains of the company, the meetings were not contiguous, but separated by one week. Here, we need to remark that a final session, not counted as the evaluation sessions, was devoted to the evaluation results presentation.

Table II. Mapping between sessions (days) and ATAM steps.

Day	Steps executed
1	1) Present the ATAM. 2) Present the Business Drivers. 3) Present the Architecture.
2	4) Identify the Architectural Approaches. 5) Generate the Quality Attribute Utility Tree.
3	5) Generate the Quality Attribute Utility Tree (continued). 6) Analysis of Architectural Approaches. 7) Brainstorming and Prioritization of Scenarios. 8) Analysis of Architectural Approaches.
4	9)Presentation of results

In the first day, we presented the ATAM method and we also explained the concept of quality attribute, putting special attention in the relationship between software architecture, stimuli and response. This was done because the company has never had an architecture evaluation process before. One challenge we faced here was to convince the company about the benefits of having a planned architecture evaluation. In the first day, we also discussed business drivers. We mention here this as a discussion as this was more a guided elicitation than a simple presentation. Then, the development team, particularly the chief architect and the main architect (see

[1] In Chile, the legal workday spans to 9 hours.

section 3.1), presented the actual architecture. They also presented the rationale for many of the decisions they took on the past, but mainly because many of these decisions were clearly outdated for today. At the final moments of the first session, we discussed at a very high level many architectural approaches, being microservices regarded as one of the most interesting for the company (this was partly motivated by the intention of the company to go from a desktop-licensed business model to a cloud-SaaS model).

3.2.2 *Second Day: Identification of Architectural Approaches and Quality Attributes Utility Tree Generation.* In second and third sessions, we elicited scenarios by using utility trees and scenario brainstorming and then we moved to a prioritization activity. For prioritization, we used one of the ATAM's relative ranking schemes (High, Medium, Low) as explained in [7]. When elaborating the utility tree we prioritized each node (i.e., the scenarios) along two dimensions (one regarding the importance of the node to the success of the system, and the other one regarding the difficulty the team feels about achieving the level of the quality attribute) [7].

As noted in [15], and given our previous experience with utility trees, we used a quality attributes list as a guidance to avoid unnecessary scenario elicitation. Particularly, we used the Microsoft Quality Attributes list [10]. Please note that we are not arguing for or against this specific quality attributes list. Our decision was merely practical: the Microsoft's list is easy and free to access, a valued characteristic by the company. We also needed to add another quality attribute: data integrity.

3.2.3 *Third Day: Finalizing the Utility Tree, Analysis of Architectural Approaches, Brainstroming an Prioritization.* The utility tree generation was demanding in terms of effort. We did not complete the step in the second day, thus we needed to account for a third day. It is in this step in which architectural risks, nonrisks, sensitivity points and trade-off points are discussed and identified. Trade-off identification was harder than expected because previous decisions among alternatives were made always considering one quality attribute at a time, and trade-offs remain implicit in the team's heads. This means that trade-offs between quality attributes and related decisions were not so apparent as one would expect in order to have a good discussion about them. As external consultants we played an important role in expliciting the trade-offs among these alternative technical decisions.

After finishing the quality attribute utility tree, we continued with an analysis of the architectural approaches. This time, we considered the approaches themselves (i.e., the ones elicited in the previous "analyze architectural approaches" step) and we also analyzed them considering the high-priority scenarios identified in the utility tree generation.

Then we moved to a brainstorming session. In this session, the external consultants were also present as key stakeholders for this step. Recalling section 3.1, the external consultants were not be able to be present at every step due to the strict time constraints. Immediately after finishing the brainstorming, we executed another prioritization using a voting mechanism (a spreadsheet was presented to all stakeholders with the brainstormed scenarios and we scrolled down the sheet as the stakeholders voted for them).

With the new information, we again reiterated the analysis of architectural approaches. Some scenarios proved to be in conflict with previously generated ones. Thus, we appreciated the benefit of having this new iteration of the step as a testing mechanism.

3.2.4 *Fourth Day: Presentation of Results.* A final day was scheduled for presenting results. We did not consider this day as a concrete evaluation session as the main purpose of the meeting carried out on this day was merely for presentation purposes (i.e., no more evaluation was done).

As we were evaluating the architecture in the context of a migration, the presentation also included many technological aspects (e.g., frameworks, messaging software, etc.). The presentation lasted about 2 hours and some technical questions arose from the company's development team.

It is also worth noting that some steps were not linearly enacted, specially the last ones, a characteristic that is expected in ATAM and noted in [7].

In summary, we dealt with 10 key architectural decisions addressing 35 scenarios. We remark here that some architectural decisions encompass other decisions (e.g., committing to RESTful microservices).

3.3 Reflecting on the ATAM use

Despite the fact that at the beginning we faced a challenge in trying to explain the benefits of doing architectural evaluation, the company finally felt comfortable with the process. They found particularly useful the structured and systematic discussion about scenarios and their relation to quality attributes and the elicitation and analysis of tradeoffs associated with each decision (decisions come as an effect of a pattern application). Similar benefits have been also reported before [15].

This very evaluation is a trade-off analysis per se, although one at business level and beyond the scope of ATAM (and of software architecture itself). Time was the most scarce resource in the company. We will not mention it as a drawback, but as a cost, in the sense that any team thinking in using ATAM for architecture evaluation should consider that the process is time consuming.

Nevertheless, the company felt comfortable with the method as the time went on mainly because they began to recognize that many of the architectural design decisions that were implied by the use of the aforementioned patterns began to be systematically analyzed considering trade-offs and risks (among other aspects).

3.4 Architectural Decisions, Risks, Nonrisks, trade-offs and Sensitivity Points

An architectural evaluation has outputs that embody the tangible work products in the evaluation process. In particular, ATAM prescribes the following outputs [7]:

—Architectural decisions: the resolutions, at architectural significance level, made and documented.

—Risks: potentially problematic architectural decisions.

—Nonrisks: unlike risks, nonrisks are *good* architectural decisions that rely on assumptions that are frequently held.

—Sensitivity points: in the context of an architectural decision, some components and/or components relationships may exhibit properties that are critical for achieving a particular quality attribute level, and finally.

—Trade-offs points: a property exhibited by some components and/or components relationships that affects more than one attribute and is a sensitivity point for more than one attribute. *important* events impact positively maintainability quality attribute, but it also may impact negatively performance).

While some of these outputs are generic, others are specific to ATAM (e.g., sensitivity points and trade-offs) [9].

Due to the intrinsic limitations in terms of space of an article, we cannot list every architectural decision. However, we present two examples we find interesting (see tables III and IV). Both tables present the main decisions derived from the use of the patterns with their related quality attributes, scenarios, sensitivity and trade-off points, risks and non risks.

We also provide in figure 1 a general overview of the proposed new architecture using the API Gateway and Saga microservices-related patterns. The figure shows the main components (using UML's package notation) of the new architecture. According to the model, the SAGA pattern was used in *orchestrated* version which is typically regarded as developable and deployable with modern technology [5].

4. RELATED WORK REGARDING ATAM

Architecture evaluation is an ongoing topic in software engineering which deals with systematic analysis of software architecture to identify potential risks and verify that quality requirements has been addressed in the design [11][12]. Among the several methods for architecture evaluation, there is ATAM. The Architectural trade-off Analysis Method, appears as a method to understand the consequences of architectural decisions with respect to the quality attribute requirements of a system [7][8].

Table III. API Gateway.

Decision:	Use of an API Gateway as a single point of entry to the services and for load balancing.
Related Quality Attribute:	Flexibility, Performance.
Related Scenarios:	-When requesting a composite result, developers should have a single point of entry for developing mashup-based views. -Salary settlement, for 4000 workers, with two liquid assets must be computed in less than 20 minutes.
Sensitivity Point:	An API Gateway is essential to avoid low cohesion between calls in services.
Trade-off Points:	Development and maintenance effort. Also, if developed at home, a new component must be considered in the planning. If another existing component is used for the gateway, monetary cost and learning effort must be considered.
Risks:	The API Gateway itself becomes another entry point which could endanger client's workers data.
NonRisks	-

Table IV. Saga Pattern used to facilitate data transactions management.

Decision:	Use of Sagas in Command/Orchestrated Mode to manage transactions.
Related Quality Attribute:	Data Integrity
Related Scenarios:	When changing a data record, the system must be responsible for updating all related records in all data stores, as required.
Sensitivity Point:	While Command/Orchestrated Mode is not mandatory, the use of a Saga service component is critical for ensuring correct data integrity in a distributed context.
Trade-off Points:	Development effort, especially at the beginning, might be affected negatively (more effort required).
Risks:	Complex transactions managed by a Saga can be implemented incorrectly if the team does not make a good interpretation of the Saga pattern.
NonRisks	-

Although ATAM is today almost 20 years old [7], there is little practical use evidence reported in the literature. We can mention the use of ATAM to evaluate an agent-based system's architecture [13] where it is interesting to note that the authors used ATAM with a set of quality attributes provided from previous discussions in the Artificial Agents community, making clear a decoupling between the method and the set of quality attributes used. Another interesting issue in this article is the absence of utility trees. They used a distinct approach in which a scenario is first described and then the potential quality attributes that can be affected are discussed.

ATAM has been used to evaluate an architecture for descentralized control of a transportation system [15]. In that work, the authors reported three key benefits from using ATAM. First, ATAM facilitated analyzing and identification of trade-offs in a structured way, allowing the architects and developers to not only gain more insight about some previously known trade-offs, but also noting them down. Second, when preparing documentation for the evaluation and when discussing the architectural approaches, ATAM use revealed non-conformance (i.e., erosion) between the architecture and the implementation itself. The team learned that architecture should constrain the implementation, thus making clear the importance of architecture to be included in development. Third, and finally, the authors mention that ATAM use revealed several architectural improvements opportunities. The authors reported utility trees as the most important artifact in ATAM.

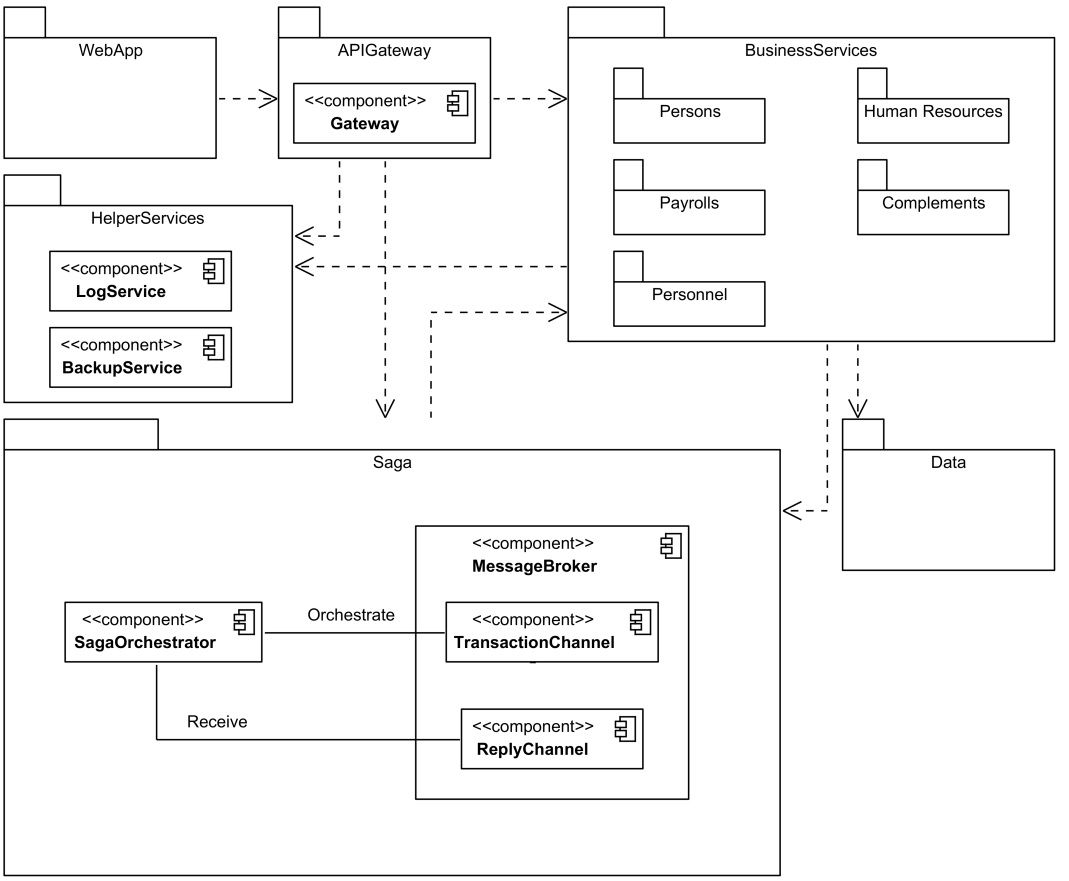

Fig. 1. Overview of the new architecture.

In [14], the authors present their experience using ATAM in eleven cases from which they drove some recommendations for enacting ATAM. Regarding our case, the most important one was the use of a process enforcer [14]. Interestingly, they also cite the work in [15] as one of the few papers that devote *more than a few lines* to industrial application of architecture evaluation. Besides obvious roles, the evaluation teams also considered a scribe, a process enforcer and a proceedings scribe. Also, the authors highlight the structured scenario analysis of ATAM as a powerful tool in the method, in agreement with [15]. They also presented the use of templates for the ATAM session records, which proved to be useful for process enactment and enforcement.

5. CONCLUSIONS

In the context of an architecture migration, we chose ATAM as an evaluation method to provide support for architectural design decisions for the new architecture. We scheduled and carried out three sessions in which we executed the ATAM steps.

In this article, we presented our experience in the use of the method, specially regarding the use and application of two concrete microservices-related patterns: API Gateway and Saga. ATAM allowed us to systematically analyze the implied design decisions related to these patterns considering, among others, two key issues: trade-offs and risks.

Finally, we highlight the very valuable assets that were created in the architecture evaluation. These assets allowed the company to assess the real benefits of migrating this old system, as well as the costs and trade-offs. The new architecture is being developed in an iterative-incremental manner, and many of the design decisions presented in this paper are being worked in this way. For example, the SAGAs began to be developed using an SQL server database as a messaging system. This was motivated by the fact that clients were reluctant to the idea of deploying on-premise *new* technologies such as RabbitMQ (and they are also still not convinced of using the cloud, at least for the first stages of the migration). We also remark that the evaluation is not intended for answering close-ended questions, thus we cannot expect the evaluation ending in a yes or no for the migrations. Instead, the assets created in the evaluation are used as well as other information to make appropriate technical and managerial decisions.

REFERENCES

Bisbal, J., Lawless, D., Wu, B., Grimson J.: Legacy Information Systems: Issues and Directions. IEEE Software, Volume: 16, Issue: 5, (1999).

Stonebraker, M., Brodie, M. L.: Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach, Morgan Kaufmann, (1995).

Richardson, C.: Microservices Patterns: API Gateway, `https://microservices.io/patterns/apigateway.html`, reviewed at: 2018-03-20.

Garcia-Molina, H., Salem, K.: Sagas. In: SIGMOD Rec., Volume: 16, Issue: 3, (1987).

Richardson, C.: Microservices Patterns: Saga, `https://microservices.io/patterns/data/saga.html`, reviewed at: 2018-03-20.

Knodel J., Naab M.: Software Architecture Evaluation in Practice: Retrospective on more than 50 Architecture Evaluations in Industry. IEEE/IFIP Conference on Software Architecture (2014).

Kazman R., Klein M., Clements, P.: ATAM: Method for Architecture Evaluation. Carnegie Mellon University (2000).

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., Carriere, J.: The Architecture Tradeoff Analysis Method. In: IEEE International Conference on Engineering of Complex Computer Systems. IEEE Computer Society, Los Alamitos (1998).

Clements, P., Kazman R., Klein M.: Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley Professional, first edition, (1998).

Microsoft: Chapter 16: Quality Attributes. Microsoft Application Architecture Guide (2nd Edition), `https://msdn.microsoft.com/en-us/library/ee658094.aspx`, reviewed at: 2018-03-20.

Dobrica L., Niemela E.: A Survey on Software Architecture Analysis Methods. IEEE Transactions on Software Engineering, Volume: 28, Issue: 7, (2002).

Li W., Henry S.: Object-Oriented Metrics that Predict Maintainability. J. Systems and Software, Vol. 23, No. 2, (1993).

Woods, S.G., Barbacci, M.: Architectural evaluation of collaborative agent-based systems. Technical Report CMU/SEI-99-TR-025, CMU/SEI (1999)

Reijonen V., Koskinen J., Haikala I.: Experiences from Scenario-Based Architecture Evaluations with ATAM. In: Babar M.A., Gorton I. (eds) Software Architecture. ECSA 2010. Lecture Notes in Computer Science, vol 6285. Springer, Berlin, Heidelberg, (2010).

Boucké N., Weyns D., Schelfthout K., Holvoet T:. Applying the ATAM to an Architecture for Decentralized Control of a Transportation System. In: Hofmeister C., Crnkovic I., Reussner R. (eds) Quality of Software Architectures. QoSA 2006. Lecture Notes in Computer Science, vol 4214. Springer, Berlin, Heidelberg, (2006).