2019 IEEE International Conference on Software Architecture Companion (ICSA-C)

Assessing migration of a 20-year-old system to a micro-service platform using ATAM

Pablo Cruz, Hernán Astudillo
Departamento de Informática
Universidad Técnica Federico Santa María
Avenida España, Valparaíso, Chile
Email: {pcruz, hernan}@inf.utfsm.cl

Rich Hilliard
IEEE Computer Society
Bar Harbor, United States
Email: r.hilliard@computer.org

Miguel Collado
Innovasoft
Padre Alonso de Ovalle 612
Oficina 4 Santiago, Chile
Email: m.collado@winper.cl

***Abstract*—Architecture evaluation is a systematic approach to evaluate target architectures and ATAM (Architectural Trade-off Analysis Method) is one of the available methods. Migration of software systems imply many architectural decisions that should be systematically evaluated to assess concrete trade-offs and risks. This article reports on the ATAM usage at a mid-size Chilean specialty software development company to assess the migration of its 20-year old flagship product to a micro-service platform. Over three days, 10 key architectural decisions, addressing 35 scenarios, were considered. Since almost all requirements were deemed essential for legal reasons, the evaluation scheme used a modified importance dimension distinguishing among business-key (first line deal breakers), legally-mandated (second-line deal breakers), and desirable requirements. Key lessons learned include the very positive client feedback on the introduction of systematic evaluation of architectural choices using business criteria.**

I. Introduction

Over time, the architecture of a software system will be subject to many changes. These changes are motivated by new requirements, changes in market conditions and laws, or the company's intention to succeed in the long term.

WinPer is a payroll and human resources software system developed and maintained by Innovasoft, a mid-size software company based in Santiago, Chile. As WinPer is today almost 20 years old, and considering Innovasoft's intentions to succeed in the long term, an architecture migration was internally discussed and proposed for the software.

One of the key constrains by Innovasoft is motivated by the current functionality implemented in WinPer. Although many architectural and technological aspect might seem a-priori outdated, the functionality is a competitive advantage for the company. Thus, there is a main restriction to the new architecture: the functionality must be maintained.

This paper presents the evaluation of the software architecture of a 20-year-old payroll and human resources system developed and maintained by a Chilean company. This evaluation was made in order to assess the fitness of the current architecture and to find well-supported evidence and guidance for the future evolution of the software and its architecture.

We understand a legacy system as one exhibiting two characteristics: (1) it is critical for the business [1] and (2) significantly resists modification and evolution [2]. Both characteristics are observed in WinPer.

Thus, we decided to perform an architectural evaluation for the following two reasons:

1) First, the company wanted to assess the actual architecture's fitness in the current and future context.
2) Second, the company wanted to avoid *fashionable architectural decisions*. That is, we needed a framework for systematic discussion about many possible architectural approaches for the target architecture.

For the architecture evaluation, we used ATAM, the Architectural Trade-off Analysis Method. Among various methods for evaluation, we finally decided to use ATAM because of our previous experience in this one.

When using ATAM, we faced an interesting issue at the prioritization step: as most of the scenarios were legally mandated, almost all of them were noted as high priority in terms of importance to the success of the system [7]. In practice, this hinders planning the potential migration of the architecture. However, we found some scenarios to be *special* compared to the others: they were derived by the company's strong belief that they must be achieved to gain an important competitive advantage in the market.

We argue in this paper that in particular cases in which many, or all, requirements are directly or indirectly legally mandated, users of ATAM should consider a third dimension for prioritization, besides the classical two-dimension prioritization (importance and difficulty [7]).

The paper is structured as follows. Section II presents the company's intentions and context. Section III briefly summarizes ATAM and its steps. Section IV describes our particular experience using ATAM to evaluate this system. Section V presents the main lessons learned in this experience. In section VI we present related work and, finally, in section VII we discuss our main conclusions.

II. The Migration Context

Innovasoft is a mid-size software development company based in Santiago, Chile. Its main product, WinPer, is a payroll and human resources system. The company started its development about 20 years ago, and the software system has been maintained over time by the company with no big changes to the architecture. Rather, the maintenance has been mostly corrective and evolutive (mostly functional) in accordance to new requirements. The customer portfolio size

978-1-7281-1876-5/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSA-C.2019.00039

is about 200 clients, and in the near future Innovasoft plans to reach 500 clients. Concurrent usage is not a big concern today as almost all of the clients are companies that run the software internally (i.e., on premise deployment), thus concurrent usage is only in the context of a client organization.

WinPer has seven groups of modules:

- Basic modules: common and shared functionality.
- Payroll modules: related to payroll calculations.
- Personnel modules: to support personnel department functions, and related to the functionality of the payroll modules.
- Human resources portal modules: to provide information querying features for the clients.
- Human resources modules (support): to provide Web-based information querying for the clients.
- Workers management modules: functionality for managing workers in an organization.
- Complementary modules: also common and shared functionality, but understood as complementary to the basic modules.

A. WinPer's Architecture

WinPer's current architecture is based on several modules organized into the groups presented at the beginning of this section. Most of them are developed in Team Developer (Gupta). The only exception are the modules related to the Web portal, which are developed in .NET (C#).

The architectural style is client-server with the main database (SQL Server) being the data server for all of the modules. Much business-related functionality is implemented as stored procedures. It was said that this was an architectural decision in the beginning of the development of WinPer to address many performance issues that appeared with the language itself. The new architecture is expected to face this challenge by moving business functionality from stored procedures to a domain-related layer. Moreover, Team Developer-based modules and the Web portal use the database server as the data source. Team Developer modules are desktop applications that run in Windows as independent `.exe` programs, though they are all called by a common main interface which is also a Team Developer executable program. The application state in a particular session is supported by the basic modules (to provide the running instance with temporary storage of session elements). The main database is also used by the modules to share data between them.

B. Innovasoft's Intentions

In the near future, Innovasoft plans to expand its client portfolio to about 500 clients. In this context, it is open to, though not completely convinced of, a cloud-based solution which would eventually allow them to sell WinPer in a software-as-a-service model (today, Innovasoft uses the classical license-per-user model). One interesting constraint here is that many clients are reluctant to have their data stored in Innovasoft-governed clouds. That is, they are slightly open to cloud-based storage, but with the restriction that this cloud must be under the governance of the client company (not Innovasoft).

In very particular cases, ad-hoc modules are developed, but this is not a core service of Innovasoft's business (in fact, Innovasoft wants to avoid this going forward).

We can also remark that Innovasoft is making great efforts in formalizing and modernizing the instrumentation of their development processes.

III. The Architectural Trade-off Analysis Method (ATAM)

ATAM is one of the architectural evaluation methods in use today. A key characteristics of ATAM is that the method not only allows one to determine how well an architecture satisfies particular quality goals, but also to determine how quality goals, and the architectural decisions, trade-off against each other [7]. The method has been used with newly developed systems as well as legacy ones that have been around for a long time [9] (and in general terms, architectural evaluation has been also used for software migration [6]).

The ATAM is composed of nine steps that are grouped in four categories: Presentation, Investigation and Analysis, Testing, and Reporting.

The nine steps are grouped into the categories outlined as follows.

The Presentation group includes the following three steps:

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 require to 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 group includes the following three steps:

4) Identify the architectural approaches: As noted in [7], "architectural approach" is a more general term than architectural style that will allow architects not accustomed with the term "style" to discuss and identify the many sets of decisions that are available.
5) Generate the quality attribute tree: by generating a utility tree, the team can elicit quality attributes specified 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, non-risks, sensitivity points and trade-off points are identified.

The Testing group includes:

7) Brainstorm and prioritize scenarios: in this step, more scenarios are elicited, but this time considering a larger

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

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 group includes only one step:

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

The main outputs in ATAM are approaches, scenarios, attribute-specific questions, utility tree, risks, nonrisks, sensitivity points, and trade-offs [9]. We remark here that as we were evaluating the architecture in the context of a migration, many other outputs such as architecture models and components and their relations were created.

IV. ARCHITECTURE EVALUATION IN INNOVASOFT WITH ATAM

As reported by the company's staff, using ATAM allowed them to systematically approach the elicitation and analysis of trade-offs related to the decisions that emerged as a consequence of the application of patterns. In this section, we describe the concrete experience at Innovasoft.

A. 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.

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

B. ATAM Sessions

In all sessions, the two external consultants participated. One of the consultants mainly played the role of a process enforcer and the other one acted mainly as a scribe. The process enforcer also led the sessions. Findings in each session were recorded on paper (computers were not used, to avoid unnecessary distractions) and we found the use of a blackboard to be very valuable. All important ideas written on the blackboard were photographed and then *digitalized* by the consultants. One of the key difficulties we experienced was the recording of utility trees: it was a bit hard to create the utility tree diagram, both in the blackboard and in paper (especially because of its size).

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

Of all the steps, identifying the architectural approaches and discussing quality attributes in the utility tree generation required the greatest effort. Also, in second place, discussion of business drivers was also effortful. There was also some discussion regarding scenarios that appeared to be traceable to more than one quality attributes.

1) First Day: Presentation of ATAM, Business Drivers and Architecture: In our case we found important to remark that the method does not necessarily require to be enacted linearly (i.e., some steps may appear to overlap with others).

We scheduled and carried out three evaluation sessions (three days) of about 6-7 hours each[1]. Due to time and scheduling constraints 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 for presenting the evaluation results.

On 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 section IV-A), presented the actual architecture. They also presented the rationale for many of the decisions they made 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, microservices being 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).

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

In summary, we present here the main business drivers around quality attributes:

- Usability:
  - A critical aspect is that functionality must be maintained. It is not allowed to change functionality as this is the core that supports the business of Innovasoft.
  - Innovasoft is also interested in development of a mobile application.
  - Desirable to provide to clients, in the near future, some degree of automation in some basic functional aspects.
- Maintainability, Manageability/Supportability, Reusability:
  - Innovasoft wants to depart from Team Developer. Though they recognize many benefits of the language and framework, it is becoming hard to hire people with Team Developer knowledge.
  - The new architecture should have in mind continuous integration and deployment (especially for software corrective updates).
  - Improve decoupling in developers' work.
  - Repeated code must be reduced to a minimum. Innovasoft expects to begin a systematic reuse of components.
  - Company's staff believe they have to start making decisions based upon recorded characteristics of business events (also for auditing purposes).
- Performance:
  - Payroll calculation is currently slow. Innovasoft's biggest interest is to provide a huge improvement in payroll calculation time (we discuss this in the next sections).
  - Clients should feel the agility of the system in the transition between windows or Web pages.
- Availability and Data Integrity:
  - Ensuring availability is critical. A company relies on WinPer to pay wages after payrolls are computed.
  - Internet connections are of good quality in most cases. However, in cases in which Internet connections are not good (some important clients run WinPer in remote places in which connectivity is an issue), availability of previous application consistent state is a must, even if a transaction fails due to connection problems.
- Security:
  - As data is sensitive, security is critical and must be considered in any kind of new architecture. Data must travel in a secure way.
  - Authorization and, especially, authentication is critical and most of the time required by government units.
- Interoperability:
  - WinPer has to interact with the government Taxes Department, Ministry of Labor, pension organizations, some ERPs (SAP, Softland, Oracle), among other units.

2) Second Day: Identification of Architectural Approaches and Quality Attributes Utility Tree Generation: In the 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 a relative ranking (High, Medium, Low) as explained in [7] and 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). See section V for a detailed discussion on this.

As noted in [16], 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 (available at [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 this company. The Microsoft Quality Attributes list was approved by the customer. We also remark that we needed to add another quality attribute: data integrity.

3) Third Day: Finalizing the Utility Tree, Analysis of Architectural Approaches, Brainstorming 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, non-risks, sensitivity points and trade-off points were 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 remained implicit in the team's heads.

In some cases, these technical decisions can be easily understood by developers and architects, but in our case, we as external consultant played an important role in explicating the trade-offs among these alternative technical decisions.

After finishing the quality attribute utility tree, we continued with an analysis of the architectural approaches, now considering not only the approaches themselves, but also analyzing them considering the high-priority scenarios identified in the utility tree generation.

Then we moved to a brainstorming activity. In this activity, the external consultants were also present as key stakeholders for this step. Recalling section IV-A, the external consultants were not 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 iterated 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.

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 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). Four of the ten architectural decisions are presented in the following sections. All of the outputs of ATAM, including the architectural decisions and related scenarios were recorded in a document that was delivered to, and signed off by, the company's staff.

C. Reflecting on the ATAM use

In the practice, it is interesting to see that ATAM use is itself a trade-off point. If we could name the elements to which ATAM is a trade-off point, they would be effort and business value. Thus, when using ATAM, the evaluation team should invest some time to analyze current situation to determine if the value of evaluation is clearly exceeding the effort required.

Despite the fact that at the beginning we faced a challenge in trying to explain the benefits of doing architectural evaluation, the company felt comfortable with the process. They found the structured discussion about scenarios and their relation to quality attributes particularly useful. This has been also reported before [16].

Therefore, we can argue that 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.

D. Architectural Decisions, Risks, Non-risks, 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][9]:

- Architectural decisions: the resolutions, at architectural significance level, made and documented.
- Risks: potentially problematic architectural decisions.
- Non-risks: unlike risks, non-risks 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 to achieve a particular quality attribute level.
- 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. For example, *important* events impact positively maintainability quality attribute, but it also may impact negatively performance.

Due to the intrinsic limitations in terms of space of an article, we cannot list every architectural decision. However, we present four examples we find interesting:

1) Organize the application as microservices using REST.
   - Description: a suite of small components (services), perhaps deployed in different environments. Each service will be developed and run by a particular team.
   - Related quality attribute: Maintainability, Reusability.
   - Related scenarios: 1) When doing software maintenance or when adding new functionality, reuse of code must be enforced in order to avoid duplication. 2) When some functionality stops running, a clearly identified developer or development team members must be identified to be the responsible of re-running the halted functionality.
   - Sensitivity point: RESTful Microservices facilitates decoupling of both modules and objects lifecycle.
   - Trade-off point: 1) Committing to Microservices require more effort by the development team to maintain them (including adding new microservices). 2) Microservices will also imply data isolation. Transactions must be managed by making further architectural decisions. 3) RESTful implies using HTTP as the transport medium. While HTTP es lightweight enough, it is also harder to attend security requirements.
   - Risks: A misunderstanding of resources (vs. services) could lead the development team to design and implement API's in a transaction-fashion rather than a resource-fashion. Also, microservices architectures are not fully trusted until they are *seen* working (this is important as this architecture is new to the Innovasoft teams).
   - Non-risks: N/A.
2) Decision: Use of sagas in Command/Orchestrated Mode to implement transactions.
   - Description: some transactions will necessarily involve more than one microservices and consistency must be ensured. A *saga* is a set of transactions (each one acting on a microservice) with respective compensating actions (compensating actions are available for the case in which the next transaction fails). Atomicity is at the saga level, i.e. the saga is

the long-live-transaction that must be implemented atomically. [4][5].
- 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 (of services), as required.
- Sensitivity point: While Command/Orchestrated Mode is not mandatory, the use of a Saga service component is critical to ensure data integrity and consistency in a distributed context.
- Trade-off point: Development and maintainability effort, especially at the beginning, can be high. In the case of maintainability, it will be a trade-off in the long run.
- Risks: Complex transactions managed by saga can be implemented incorrectly if the team does not make a good interpretation of the Saga pattern. Also, there is a risk for misinterpreting some transactions as not atomic when in the practice should be treated as atomic.
- Non-risks: N/A.

3) Use of a dedicated component to record events.
- Description: a log service is devised to store any event of interest for the company. The service can also answer with events data.
- Related quality attribute: Manageability, Maintainability and Supportability.
- Related scenarios: When critical business transactions are executed, the system shall be able to record related events for, both, audits and maintenance/support purposes.
- Sensitivity point: Recording events is critical for ensuring effective and efficient client support and maintainability when derived by problems reported to the support area.
- Trade-off point: A high rate of events recording may impact negatively performance.
- Risks: N/A.
- Non-risks: N/A.

4) Hypermedia as the Engine of the Application State.
- Description: HTTP is stateless. One way in which state can be maintained (thus allowing users to *navigate* through the system without having to characterize all requests) is to use hyperlinks in the resources to create the relations that allow state to emerge in a session.
- Related quality attribute: Usability.
- Related scenarios: When user *navigates* through the system, state must be maintained so he or she does not need to re-enter the same data to characterize the request.
- Sensitivity point: This decision is a sensitivity point. As we chose to commit to a RESTful microservices architecture, HATEOAS is critical for maintaining state between modules (instead of the client working with single unrelated resources).
- Trade-off point: An interesting trade-off appears here as HATEOAS will make operations to appear to the client as a kind of workflow. A workflow can hinder flexibility for clients, in some cases, but we expect this to be of lesser impact as it was said by the company that stateful navigation is key.
- Risks: N/A.
- Non-risks: This decision is a non-risk point as it is understood as a good architectural decision (well understood and easy to implement).

Figure 1 shows the main components of the proposed architecture using the *API Gateway* [3] and *Saga* [4] [5] microservice related patterns (using UML's package notation). According to the model, the Saga pattern was used in *orchestrated* version that is typically regarded as developable and deployable with modern technology [5]. Also, this architecture overview shows that a new Web application will be the main interface (replacing the old desktop applications). API Gateway is used not only for security reasons, but also to lower coupling between the user interface and the services (this was motivated by the company's intention of developing soon a new mobile interface).

V. Lessons Learned

The most important lessons learned in our experience using ATAM in this context are the following:

- An "ATAM process enforcer" is required. This is not new, as it was recognized previously in [15] and also noted in the Software Architecture Review and Assessment Report (SARA) [14], especially for situations where evaluations seem to get out of control. When the evaluation team begins to ramble in the discussions, the process enforcer's main task is to put again the evaluators in line with the method.
- Almost all scenarios were directly or indirectly legally mandated. Therefore, most of them were rated as *high priority* in terms of importance for the success of the system. This hinders planning for an architecture migration, a critical aspect in our experience considering that we enacted ATAM in the context of an architecture migration. To face this challenge, we added another dimension (besides "importance of the scenario for the success of the system" and "how easy the level of the quality attribute related to the scenario will be to achieve" [7]). This dimension is *critical* for the migration project to be undertaken, i.e. if the scenario cannot be achieved, then the project is canceled or reformulated. We marked critical scenarios as deal-breakers. We exemplify below the three kinds of elicited scenarios:
  - Directly or indirectly *legally mandated*: "Worker can access to salary settlement report 24/7 (related to availability quality attribute)."
  - *Deal-breaker*: "Salary settlement, for 4000 workers, with two liquid assets must be computed in less

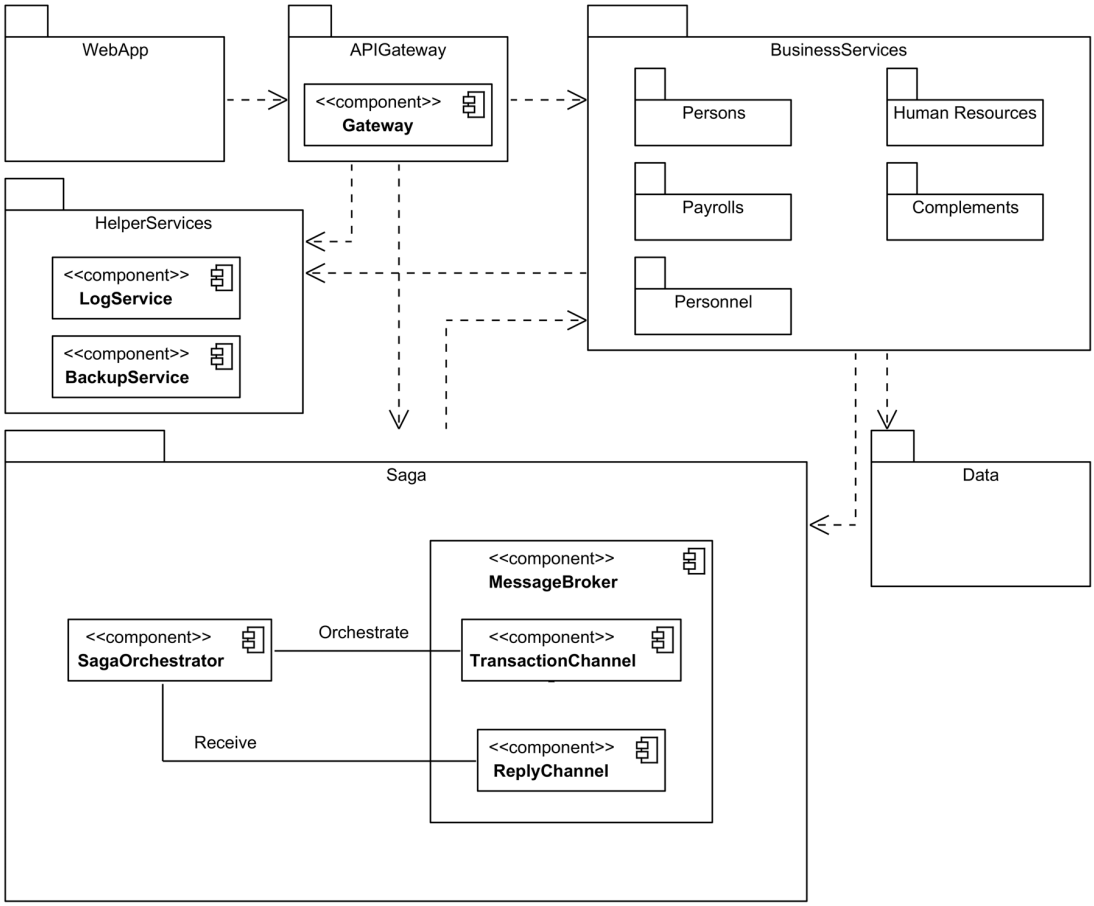

Fig. 1. Overview of the new proposed architecture.

than 20 minutes (related to performance quality attribute)."

- *Desirable*: "When user *navigates* through the system, state must be maintained so he or she does not need to re-enter the same data to characterize the request (related to usability quality attribute)."

A deal-breaking scenario can also be used to guide the migration planning. The components that implement the requirements pointed to by these kind of scenarios should be implemented first. If the implementation shows that the specified level of the quality attribute cannot be achieved, then the project is canceled or reformulated. In fact, it is interesting to note that the third example was not rated as high priority because the company and their clients felt relatively comfortable with a *stateless navigation*, although an architect would have considered this a critical aspect.

- Most, if not all, experiences of evaluation with components are described at a very abstract level. We argue that today's evaluations require considering more concrete components and connectors. For example, a company would appreciate that the evaluation mentions concrete loggers that can be used to log events (e.g., log4j) rather than simple phrases like "a component for logging events".
- Finally, another characteristic ATAM seems to lack is to consider a step for discussion of ways to manage risks. If risks appear, the natural question by stakeholders is how we deal with risks. In particular, when we mentioned the risk related to RESTful APIs in which the development team could end designing and implementing service-oriented ones, our answer was to elaborate and distribute a guide that will codify the way APIs must be designed and versioned. However, this was a last-minute answer, not an answer prepared in a concrete kind of "Analyze Architectural Decisions Risks" step.

VI. Related Work

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, ATAM (Architectural Trade-off Analysis Method) allows to under-

stand the impact of architectural decisions on the quality attribute requirements of a system [7][8].

ATAM is today almost 20 years old [7], and there are some reported uses 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.

In another interesting case [16], the authors used ATAM to evaluate an architecture for decentralized control of a transportation system. They reported 3 key benefits from using ATAM. First, ATAM facilitated the analysis and identification of trade-offs in a structured way, allowing architects and developers to not only gain insights about some previously known trade-offs, but also writing them down. Second, when preparing documentation for the evaluation and when discussing the architectural approaches, ATAM usage 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.

Finally, in [15] the authors present their experience using ATAM in eleven cases. Interestingly, they cite [16] 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. The authors highlight the structured scenario analysis of ATAM as a powerful tool in the method, in agreement with [16]. They also presented the use of templates for the ATAM session records, which proved to be useful for process enactment and enforcement.

VII. Conclusions

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

Among issues uncovered, we found interesting some difficulties in the prioritization step which we argue is due to the fact that many, if not all, of the functional and non-functional aspects of the system are legally mandated. As mentioned before, ATAM was used in a wider context (an architecture migration). Given this context, we prioritized some scenarios as deal-breaking ones. These scenarios are critical for the project to be undertaken or re-evaluated. This prioritization allowed us to provide the company a concrete scheduling for migration. Specifically, we were able to indicate with which component to start the migration (i.e., the ones that implement the functionality for the deal-breaking scenarios).

It is remarkable that ATAM, despite being more than twenty years old, remains an interesting approach to evaluate architectures regardless of the natural technological advancements which imply many new potential components.

The company is currently migrating the current architecture to a microservices-based one, using an iterative and incremental approach. Many of the design decisions presented in this paper are being worked on in an iterative manner; for example, the sagas are being developed using an SQL Server instance as a messaging system because Innovasoft's clients are reluctant to deploy on-premise *new* technologies like *RabbitMQ* (and they are also not yet convinced of using the cloud, at least for the first migration stages).

Acknowledgment

This work has been partially supported by CONICYT through Basal FB0821 CCTVal.

References

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

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

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

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

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

[6] 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).

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

[8] 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).

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

[10] 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.

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

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

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

[14] SARA W.G.: Software Architecture Review and Assessment (SARA) Report, Version 1.0 (2002), https://pkruchten.files.wordpress.com/2011/09/sarav1.pdf, reviewed at: 2018-11-20.

[15] 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).

[16] 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).