Architectural Evaluation of Data Center System using Architecture Tradeoff Analysis Method (ATAM): A Case Study

I Made Putrama[1], Kadek Teguh Dermawan[2], Gede Rasben Dantes[3], Kadek Yota Ernanda Aryanto[4]
Informatics Engineering Education Department[1, 2], Informatics Management Department[3, 4]
Universitas Pendidikan Ganesha
made.putrama@undiksha.ac.id[1], teguh.dermawan@undiksha.ac.id[2], rasben.dantes@undiksha.ac.id[3],
yota.ernanda@undiksha.ac.id[4]

***Abstract*—a good and robust integration system cannot be separated from how the initial design of the system is selected and implemented. Therefore, integrating existing systems to become a large system that meets the needs of other connected systems requires full assessment and verification so that when the system is implemented, the necessary changes required in the future will be able to be accommodated seamlessly. This paper demonstrates an evaluation of a prototype of a data center system using a scenario-based architectural evaluation method called ATAM (Architecture Tradeoff Analysis Method), by analyzing the behavior exhibited by the developed system. The results of the analysis shows that some improvements are needed especially in terms of security, reliability and compliance in order to meet the relevant requirements of the robustness and the ease of doing system enhancement and maintenance.**

Keywords—Integration, Data Center, Data Warehouse, Architecture, Evaluation, ATAM

I. INTRODUCTION

Universitas Pendidikan Ganesha (Undiksha) has built various information systems especially for the needs of the implementation of academic activities such as teaching management, student academic activities, data management including lecturer, staff, research data and so forth. Specifically there are currently dozens of information systems that have operated within the Undiksha environment where each system is built separately and has a separated database system as well. Initially the systems were created with the aim to meet the need to facilitate processes that were done manually. As such, the systems were made focused on the needs in a field of study and mostly separated from the existence of other systems that could have an interconnected information among them. Moreover, some of the created systems were operated separately and they were useful only during the beginning of the system launching and then neglected due to the various problems of overlapping information found, and the information system needs further evaluation both in terms of its design consolidation and user experience. At the end of 2016, the Undiksha IT team proposes and designs a prospective system called Data Center (DC) to accommodate the needs of institutional accreditation so that information obtained from existing systems will be more organized and synchronized. Although many systems have been built, as observed there is no standard practice in applying an architectural system evaluation performed beforehand either at the design stage or upon the operation of the system.

The proposed centralized DC system needs to be developed well to integrate data from all supporting systems, so that duplication issues as well as problems due to the existence of various data structures can be reduced because the information contained will be the main source of overall academic activity in Undiksha. During the early development of the DC system, the Information Technology team at UPT-TIK Undiksha department has been exploring and designing a data warehouse server for the DC system using Enterprise Application Integration (EAI) technology that specifically implements a service-based framework called Enterprise Service Bus (ESB) [1].

While undertaking the enhancement of this system, a study has been conducted primarily in relation to the overall selection of the applied system architecture. When working on system development, human resources availability is also a critical determinant for completing and meeting the system requirements. This paper focuses more on how the choice of architecture is made on several options by prioritizing the needs of the system quality attributes by using a method called Architecture Tradeoff Analysis Method (ATAM).

ATAM is a well-known scenario-based approach system architecture evaluation method. In doing the evaluation, one has to consider a set of scenarios which focus on either the system non-functionality or the qualities exhibited by the system. ATAM has been a proven technique for systematically evaluating software architecture for fitness of purpose [2], [3]. There have been many similar system architectures evaluated using the method such as done by Gambo et al. in [4], who examined a Health Information System and stated that addressing performance quality attribute among others is the most crucial. With ATAM approach, the authors show in the generation of the scenario based evaluation, performance issues were broken down into quality attributes for analysis. Other cases as in [5], the authors address security requirements evaluation for an SOE-based ecommerce system which optimized the existing SOA architecture solution with an incorporation of caching mechanism. Another case study has used ATAM and successfully evaluated and improved an online application as described in [6].

978-1-5386-3001-3/17/$31.00 ©2017 IEEE

The evaluation result of this case study as obtained reveals an identified alternative system architectural as well as improvement approaches that has been taken.

This paper is structured as follows; Section I is the introduction, Section II provides a brief description about the evaluation method, ATAM, Section III describes an overview about the DC system and the evaluation approaches with the result obtained and Section IV draws conclusions.

II. METHOD

ATAM has a goal to elicit and refine the architecture's quality attributes whether the architectural design decisions have addressed the quality attribute requirements satisfactory [7]. The quality attributes consist of various system responses such as performance, security, availability, flexibility and so forth. As also described in [6], ATAM uses architectural approaches instead of architectural styles. Architectural style at least consists of a set of component types, connector or interaction mechanism, and informal cost and benefit description. The summarized steps involved in ATAM are depicted in Figure 1.

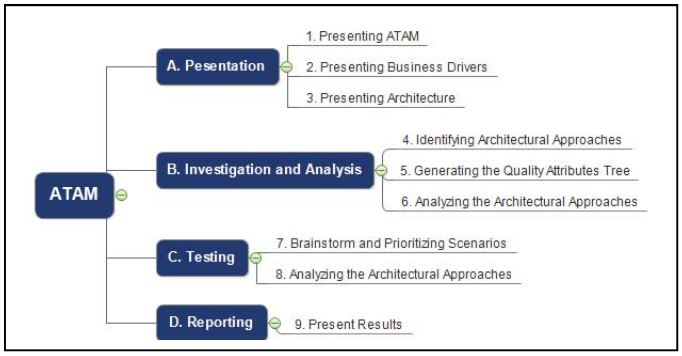

Figure 1 ATAM

ATAM consists of nine steps which can be categorized into four groups such as: **A. Presentation**, which involves activities such as (1) *Presenting ATAM*. During this activity, describing the evaluation method to the system stakeholders is required which typically includes customer representatives, system architectural team, user representatives, administrators, developers, testers, integrators, etc. (2) *Presenting Business Drivers*. In this activity, the project manager needs to explain the business goal which motivates the development effort that will become the main architectural drivers for e.g. high availability, released time, system performance, security, etc. (3) *Presenting Architecture*. In this activity, the system architect needs to elaborate the proposed architecture which focuses on a solution to address the business drivers. **B. Investigation and Analysis**, which involves activities such as (4) *Identifying Architectural Approaches*. During this activity, the system architect develops or identifies the available architectural approaches (5) *Generating the Quality Attributes Utility Tree*. In this activity, the system quality attributes are identified into level of scenarios (6) *Analyzing the Architectural Approaches*. In this activity, the approaches aimed at meeting non-functional requirement are further analyzed and points such as risks, sensitivity, and tradeoff are identified. **C. Testing**, which involves activities such as (7) *Brainstorm and Prioritizing Scenarios*. In this activity, a broader set of scenarios are elicited from the whole group of stakeholders, voted and selected (8) *Analyzing the Architectural Approaches*. In this activity, step 6 is reiterated but including the broader set of scenarios generated at step 7 to point out additional approaches, risks, sensitivity, and tradeoff points which are then documented. **D. Reporting**, which involves (9) *Present Results*. In this activity, the team report the findings along with any proposed mitigation strategies to the stakeholders.

III. RESULTS

To describe the ATAM implementation, authors reviewed steps done for the whole process as in [6] which divided them into two phases, the first phase involved all the steps from 1 to 9 of the ATAM method (as in Figure 1), the second phase repeated the step 4 to 9 until a more matured architecture system was produced. But in this paper, authors prefer to use the term *Iteration* for the implementation of the ATAM and the term *Phase* for each of the ATAM steps as described by Kazman in [8].

3.1 Iteration 1

A. Presentation

Undiksha is an education institution which has several faculties operating and each faculty has several departments such as Engineering and Vocational Faculty, department of Informatics Engineering Education, department of Electrical Engineering Education, and so forth. Previously, almost all of the departments across Undiksha had web-based applications with their own databases configured by the Information Technology department. But, they were not connected each other, so in order to provide a completed information for the institution as well as for the department's accreditation activities, the stakeholders had to extract the required information separately from each of the application and some of activities were done manually, for e.g. using a new database query script which was created on the spot upon required that could lead to data inconsistency during consolidation.

To overcome this problem, a development of DC system was proposed to provide a centralized services which allow all existing as well as new applications across Undiksha to communicate with each other and maintain their exchanged information in sync. Most of the web-based applications are coded in PHP programming language and using an open-source database instances, MySQL. The DC system would act as a backbone for all the applications that need to communicate with each other and would be the primary source whenever tasks such as information mining for academic purposes is required in future.

Management also wanted to ensure that system integration could be done strictly and in a timely manner as it was motivated by institutional interests to participate in the institution accreditation at the national level.

During the initial design, the goal was to integrate the available systems plainly, considering the available resources

such as infrastructures, human resources and technological expertise, and it was expected that future refinement would be needed further. For this purpose, the IT team wanted to use an available framework technologies with some pre-defined criteria such as *low-cost* and *open-source* in order to cut development time and to be easily maintained as well as meeting the stakeholder requirements.

Due to the urgency of the management needs, the DC system was developed under a tight schedule and with a very few resources. So there was not enough time to analyze certain strengths/weaknesses of the chosen system architecture thoroughly.

For the DC system, a prototype has been built in the previous research at Undiksha to design the data warehouse (DWH) as depicted in Figure 2. In the research, the NDS + DDS data flow architecture as described in [9] was used and implemented.

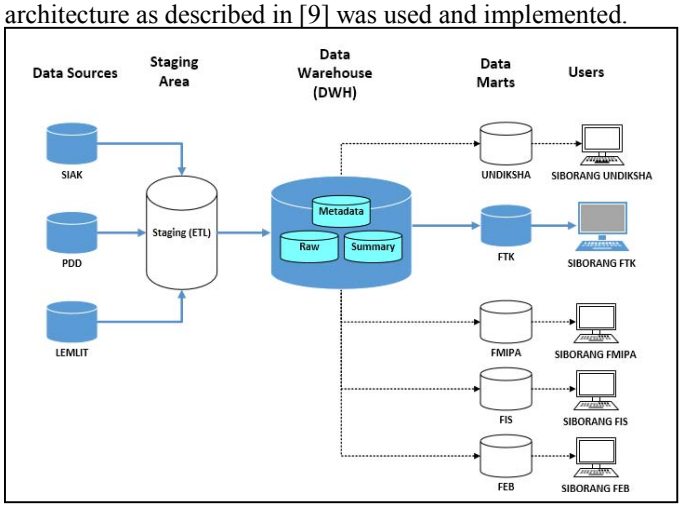

Figure 2 DC system Architecture [1]

In order to integrate the existing systems into the DWH, there were three options of the overall system architecture identified during the IT Team discussion: (1) *A Direct Connection Architecture,* whereby the DWH connects to each of the source systems to exchange data respectively through a direct-wired connection, (2) *A Web Services Based Architecture,* whereby each source system publishes its data through web services to be consumed by the DWH in need, and (3) *An Enterprise Service Bus Based Architecture*, whereby each source system shares their data through an ESB which connects to the DWH in the DC system.

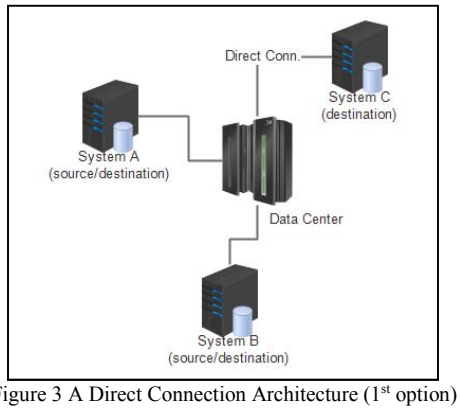

Figure 3 A Direct Connection Architecture (1st option)

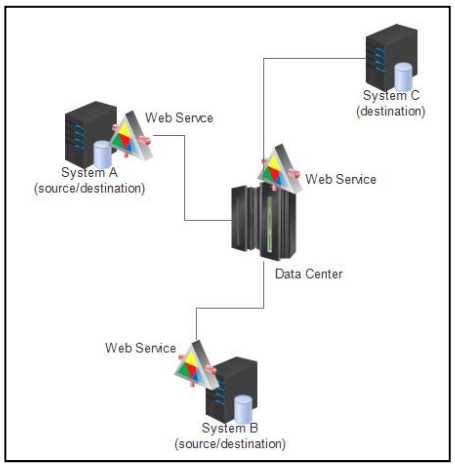

Figure 4 A Web Services Based Architecture (2nd option)

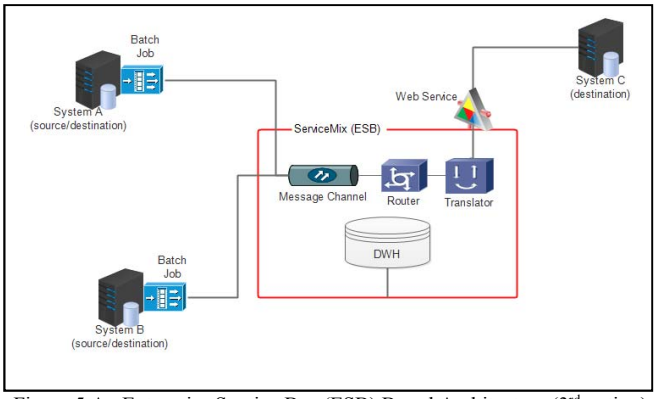

Figure 5 An Enterprise Service Bus (ESB) Based Architecture (3rd option)

B. Investigation and analysis

As stated in [10], the first work for architecture analysis is to precisely elicit specific architectural quality targets that must be assessed. For this purpose, the mechanism used is through the creation of system requirement scenarios such as listed in Table 1.

TABLE 1 REQUIREMENT SCENARIOS

Category	Scenarios
Use Case Scenario	All applications need to be connected to each other and authenticated such that to avoid any data redundancy among them
	Each of the other connected applications must be able to connect to a centralized data-warehouse and pull data upon request as needed
	The DC system must be able to sync any duplicated master data from different applications
	The DC system must be able to verify the identity of the application that connects to it before an access is allowed
	There should be a mechanism to be able to monitor the EOD jobs that were done successful and a way to re-execute the failed EOD jobs
	A finished EOD job must create an audit log information in the system
	The source system should not experience overload when another system is connected to get data
Growth Scenario	Add more data structures into the existing connected application and reflect the changes into the data-warehouse

	The DC system must facilitate any format of data transfer including binary format such as images, audio and video files
	The data added from the source applications into the data-warehouse must be identifiable
	The same information of the master data from the sourced applications must be merged and remain unique in the data-warehouse
Exploratory Scenario	Small to no change should be needed in any of the connected applications in order to implement the solution
	Any common enhancement such as to accommodate changes from the source applications' tables should require small and straight forward modification into the DC system
	The other applications able to connect through secured and non-secured networks without problems
	The system has to be expandable and easy to be customized
	Any connector required for the connected systems such as connection with different protocol has to be easily accommodated and integrated with the DC System
	The cost in term of the required software development has to be minimum and preferably using open-source software
	The community support for any open-source software used are largely available and active

The produced quality goals scenarios are then populated into a Utility Trees attributes which can be prioritized as listed in Figure 6.

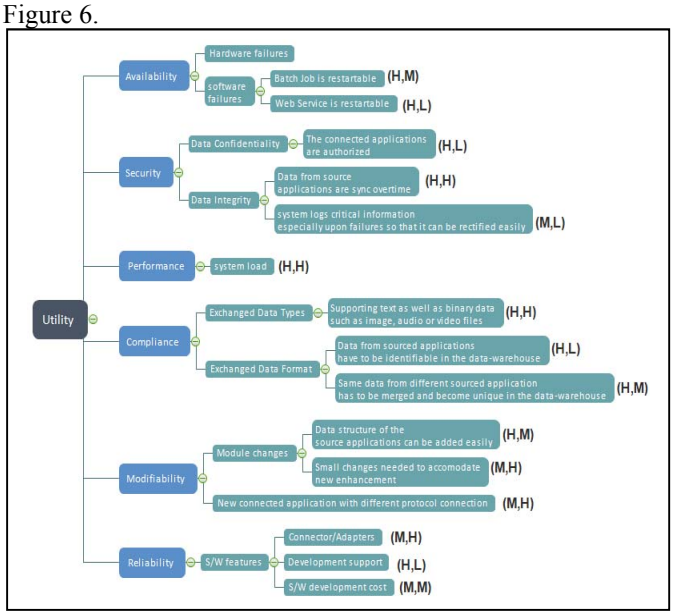

Figure 6 Attribute Utility Trees

From the listed scenarios and the most prioritized quality attributes of the utility trees, the scenario brainstorming was conducted and found out that the most important attributes include *security*, *performance, compliance* and *modifiability.* Although all of the presented utility attributes need to be addressed and analyzed, this paper will discuss only these four attributes to show how an architecture was selected and how the sensitivity as well as trade-off points were analyzed.

Security Analysis

In term of security, the 1st option was deemed to be the most un-secured architecture as each of the other systems has to allow the DC to connect to its database by giving the database credentials to the DC. As it is exposed to the outside it is very vulnerable to malicious attacks. The 2nd and the 3rd options are more secured than the 1st one.

Performance Analysis

In terms of performance, the 1st option was deemed to be the fastest because it uses a direct connection. However it would degrade the security of the system a lot. The 2nd option needs to fire requests to the source systems and publish the updated information to sync the connected destination systems. The source systems may get more requests from the DC on certain condition as such could degrade its performance as well. The 3rd option would have a less impact on performance by the nature of its architecture which uses publish/subscribe messaging mechanism. So the destination systems just need to subscribe to the interested data from the DC without the need to fire additional requests to the DC and vice-versa.

Compliance Analysis

In terms of compliance, the 1st and the 2nd options would need a separated service because most of the existing applications do not save binary data in their database. For e.g. only the location of the image, audio or video files information exist in the database while the physical files are saved on the server's hard drive. The 3rd option is the best option because it has its own built-in FTP as well as blob messaging capability to transfer binary data from DC to other systems.

Modifiability Analysis

In terms of modifiability, the 1st option would suffer the other systems when source systems' data structure changes occurred. Whilst the 2nd and 3rd options are more resistant to the source changes. In terms of changes that are needed when the DC need to retrieve new information from the source systems, in the 1st option, DC would simply write a new query to the source systems' database. However, any further changes to the source systems, would also need to change the code in the DC system which is not desirable. The 2nd option would need a less changes in the DC system, and only need to create a new service in the source system. It is also a better solution to the IT team in UPT-TIK since most of the developers are mostly familiar with web services approach. In the 3rd option, its ESB feature would give more benefit in terms of the available built-in services, however, changes made in the DC system may need some additional changes to the source as well as the destination systems.

C. Testing

From the listed scenarios and the most prioritized quality attributes of the utility trees, the scenario brainstorming was conducted to determine the most suitable option for certain requirements such as listed in Table 2.

TABLE 2 ARCHITECTURAL EVALUATION TESTING

No	Requirement description	Risk (Yes) / Non-Risk (No)
1	The data from the connected applications are sync overtime	No
2	The DC system must support transferring data in text as well as binary formats such as image,	Yes

	audio, video files between the data-warehouse and the connected application	
3	The connected application need to install an adapter in order to connect to the DC system	Yes
4	Batch job to transfer data from the source applications to the DC system is restart-able	No
5	Data from the source applications are merged and remain unique in the data-warehouse	No
6	Data structure of the source applications changes can be accommodated and reflected easily in the data-warehouse	No
7	The cost of the used of S/W is minimum	No
8	The web service implementation has to be accessible overtime and restart-able easily when down	No
9	The connection of the applications to the DC system is authorized before any request	No
10	The data from each of the source applications has to be identifiable	No
11	The community support for the S/W is largely available	No
12	There should be small changes required for an enhancement	Yes
13	The DC system should log critical information such as batch job id, source application id and so on upon the job failure	No

D. Reporting

From Table 2, there are some critical quality attribute responses that were being addressed raised a risk on architecture decision. With the 2nd option, it would suffer on accommodating the scenario 2 and 12 because it would need more changes required to settle an enhancement or a binary data transfer requirement. Whilst with the 3rd option, it would suffer on accommodating the scenario 3 because some of the existing systems are considered legacy, so major changes to the application behavior were not possible. In order to resolve the potential risks, an exploration of new solution to use some other Modifiable Off-The-Shelf (MOTS) software or framework alternatives was conducted.

Sensitivities and Tradeoffs

Given that the 2nd and 3rd options are better in most of the analyzed attributes, they were chosen and considered further. However, the compliance and modifiable attributes of these architectures are sensitive to the accommodation of additional services that are needed. At this point it was discovered an *architectural* tradeoff point for adding new services to the DC system such as FTP service or adding capability to send/retrieve binary files from DC system to other systems, and vice-versa. Because the compliance attribute would be regarded positively whilst the modifiable attribute would be regarded negatively. With this information, a refined architecture was explored.

3.2 Iteration 2

Due to the risks which emerged from the architecture decision were considered critical, together with the system stakeholders, in this case the IT Team, a second iteration of the ATAM evaluation was conducted. The team had identified a new solution on top of the existing architecture by replacing the ESB product with an alternative open-source framework to ***develop services from scratch*** (it includes the data transferring part through batch-job, which pulls data from the source applications to the DC system, and the REST-services accessible by other connected applications).

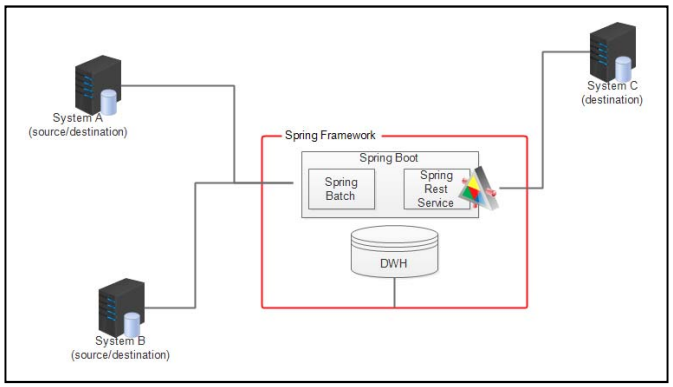

Figure 7 A refined architecture with new MOTS

With this new changes applied to the system architecture, a further step of investigation and analysis was done. By evaluating the same list of scenarios as in Table 2, the new architecture has deemed to resolve the risk decision in the points 2 and 3. But, some additional use case scenarios emerged as in Table 3 which are all considered ***at risk and critical*** to the system.

TABLE 3 ADDITIONAL SCENARIOS IN ITERATION 2

Category	Scenarios
Exploratory Scenario	The DC system should access the source application in secure way and its services has to be securely accessible
	The developed system has to be *developer-friendly* for enhancement purposes

Results

The new architecture decision is considered at risk in the following cases (1) a security module has to be developed and well-tested before it can be applied as the Open-source framework does not come with a built-in security module which can be configured *out-of-the-box* (2) the Open-source framework is considered not so *developer-friendly* as it has a wide learning-curve for new developer to understand. This has been another issue as the existing IT team mostly involves fresh graduated and less experienced developers.

IV. CONCLUSION

In this analysis of the DC system case, the use of the ATAM method has been a valuable method and helps discover strengths/weaknesses of the comparable architecture options. It was difficult to convince the stakeholders to evaluate certain technology that was found popular in use for developing a system architecture when it is new to them. Most of the managements in our case see the software development only important just to have their objective in building system within budget and in timely manner but compromising the quality of the system. As such, a lot of systems built in Undiksha were functioning but not very useful and need to be re-invented

overtime. By following the ATAM steps, we can achieve a better understanding of the architecture system alternatives as its characteristics can be breakdown into some specific quality attributes that become goals and analyzed as being at risk/non-risk to be exposed to the stakeholders for architecture decision. The approach as guided in ATAM steps has been excellent, it can be followed and repeated continuously to achieve the desired architecture result

By conducting an analysis through this ATAM method, it is observed that the initial as well as the current architecture of the DC system are not perfect and need some further exploration and analysis in the future. It is also proven from the taken course that implementing the method can be flexible, it can be applied early in advance even before any development of the system is carried out.

As observed during the research activities, there are many MOTS alternatives in the market to integrate separated systems towards building a centralized Data Center. A future research to evaluate their fitness and compatibility with any given case studies would be of great value.

ACKNOWLEDGMENT

Thanks to the UPT-TIK Undiksha IT Team that help us during the scenario brainstorming as well as facilitate various meetings during the system discussions.

REFERENCES

[1] I. M. Putrama, G. R. Dantes, and D. G. H. Divayana, "Implementation of SOA using Apache ServiceMix in Data Warehouse design," *Proc. Semin. Comput. Sci. Inf. Technol.*, pp. 127–138, 2016.

[2] I. Mistrik, R. Bahsoon, P. Eeles, R. Roshandel, and M. Stal, *Relating System Quality And Software Architecture*, 1st ed. Morgan Kaufmann, 2014.

[3] S. Chatra Raveesh, "Using the Architectural Tradeoff Analysis Method to Evaluate the Software Architecture of a Semantic Search Engine: A Case Study." 2013.

[4] I. GAMBO, A. SORIYAN, and R. IKONO, "Health Information Systems' Architectural Evaluation with Architecture Trade-off Analysis Method (ATAM): a case study in Nigeria," *J. Health Inform. Dev. Ctries.*, vol. 9, no. 1, pp. 14–24, 2015.

[5] H. Reza and W. Helps, "Security Trade-off Analysis of Service-oriented Software Architecture," *World J. Comput. Appl. Technol.*, vol. 1, no. 4, pp. 110–120, 2013.

[6] J. Lee, S. Kang, H. Chun, B. Park, and C. B. Lim, "Analysis of VAN-Core System Architecture - A Case Study of Applying the ATAM," no. January, 2009.

[7] S. Abrahão, E. Insfran, S. Abrahão, and E. Insfran, "Evaluating Software Architecture Evaluation Methods : An Internal Replication," no. May, 2017.

[8] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, "The architecture tradeoff analysis method," *Eng. Complex Comput. Syst. 1998. ICECCS '98. Proceedings. Fourth IEEE Int. Conf.*, pp. 68–78, 1998.

[9] V. Rainardi, *Building a Data Warehouse With Examples in SQL Server*. Apress, 2008.

[10] R. Kazman, M. Klein, and P. Clements, "ATAM : Method for Architecture Evaluation," *Cmusei*, vol. 4, no. August, p. 83, 2000.