Analysis of VAN-Core System Architecture - A Case Study of Applying the ATAM

*School of Engineering
Information and Communications University
{puduli, kangsw}@icu.ac.kr
*Crossent, Masters Tower 1501, 533 Dowhadong, Mapo-gu, Seoul, Korea
{hcchun, bnpark, cbnlim}@crossent.co.kr

Abstract

The Architecture Tradeoff Analysis Method (ATAM) is an architecture evaluation technique for analyzing risks, tradeoffs, and sensitivity points of architectures. In this paper we describe an experience of evaluating and improving the architecture of VAN-Core system which requires operating for 24 hours a day, 365 days a year, for online credit card transaction services. The goal of the architecture evaluation for VAN-Core system includes establishing risk mitigation strategies for identified risks and suggesting architecture improvements in addition to architecture evaluation.

1. Introduction

Architecture analysis includes operation, controlling, and measurement of various kinds of architectural elements, environmental factors, and architectural constraints. The main task of software architect is to design architecture that meets system’s functional and non-functional requirements. For example, performance requirement may be described as response time and throughput. However, these attributes may depend on the architectural structure as well as the architectural elements related to resource allocation and strategies for allocating processes to a processor. Therefore, a software architect should be able to analyze given architecture and determine whether it satisfies its functional and non-functional requirements.

The Architecture Tradeoff Analysis Method (ATAM) is an architecture evaluation technique for analyzing risks, tradeoffs, and sensitivity points of architectures. The ATAM has proven benefits such as clarified quality attribute requirements, improved architecture documentation, documented basis for architectural decisions, and increased communication among stakeholders [1].

In this paper we describe an architectural evaluation experience of applying ATAM to VAN-Core system that is an online credit card transaction system. In this case study as the results of evaluation we obtained architectural alternatives and short-term, mid-term, and long-term architecture improvements besides identifying potential problems of the current system architecture.

This paper is structured as follows; Section 2 gives a brief description about ATAM and Section 3 describes an overview of VAN-Core system and our approaches. In Section 4, we explain application results and the paper concludes in Section 5 with conclusions.

2. The Architecture Trade-off Analysis Method (ATAM)

ATAM [1, 2] provides a framework for informing technical tradeoffs and risks that we face when software architects decide designs. ATAM identifies major architectural decisions related to quality attribute scenarios extracted from stakeholders. These decisions are turned out detail quality attribute responses such as availability, performance, security, usability, and modifiability. ATAM helps identify tradeoffs those qualities attributes and make communications among stakeholders easy from each attribute point of view. Moreover, ATAM provides a framework that system design and analysis are progressed at the same time as well as clarifying and refining requirements. ATAM uses architectural approaches instead of architectural styles [1]. ATAM describes that each architectural style consists of at least as follows; a set of component types, a set of connector types/interaction mechanisms, a topological layout of these components, and an informal description of the cost and benefits of the style [1].

The summarized results of ATAM steps are as Figure 1. ATAM steps do not include implementations. Each iteration is repeated and thereby the results are renewed and elaborated. At last, it makes designs including much more information [1].
Step 1: Present the ATAM. The evaluation leader describes the evaluation method to the assembled participants.

Step 2: Present the business drivers. A project manager or system customer describes what business goals are motivating the development effort and hence what will be the primary architectural drivers.

Step 3: Present the architecture. The architect describes the architecture, focusing on how it addresses the business drivers.

Step 4: Identify the architectural approaches. Architectural approaches are identified by the architect.

Step 5: Generate the quality attribute utility tree. The quality attributes that comprise system “utility” are elicited, specified down to the level of scenarios, annotated with stimuli and responses, and prioritized.

Step 6: Analyze the architectural approaches. Based on the high-priority scenarios identified in Step 5, the architectural approaches that address those scenarios are analyzed. During this step the risks, non-risks, trade-offs, and sensitivity points of the architectural approaches for addressing scenarios are identified.

Step 7: Brainstorm and prioritize scenarios. A larger set of scenarios is elicited and prioritized from the entire group of stakeholders.

Step 8: Analyze the architectural approaches. This step reiterates the activities of Step 6 but uses the high-priority scenarios from Step 7. If Step 7 didn’t produce any high-priority scenarios that were not already covered by previous analysis, Step 8 is a testing activity. Furthermore, if we do uncover new information then this was a failing of our utility tree exercise and the architectural approaches that it led us to investigate. At this points, we would need to go back to Step 4 and work through it, as well as Steps 5 and 6 until no new information is uncovered [3].

Step 9: Present the results. The ATAM team presents the findings collected during the evaluation to the stakeholders.

3. Evaluation approach for the VAN-Core system

3.1 Overview of the VAN-Core system

VAN-Core system, an online credit-card transaction system, developed by N-Com, receives credit information from a credit-card terminal and then delivers the transaction requests to each credit-card company. And it also receives responses from a credit-card company and transmits them to a credit-card terminal again.

Purpose of Architecture Analysis

N-Com has planned to evaluate whether VAN-Core system architectures meet non-functional requirements and to evaluate the system. Stakeholders who participated in VAN-Core system architecture evaluation project had much knowledge in an online credit card transaction, but they did not have enough knowledge and experience in technologies related to the system. Moreover, stakeholders requested to establish risk management strategies not limited to architecture evaluation and to suggest and evaluate new architectural alternatives.

3.2 Our Approach for architecture evaluation

In our approach we applied tailored ATAM process and thereby stakeholders who do not have enough knowledge for the system architecture can attend and proceed for evaluating architecture and suggesting architectural alternatives. Fig. 2 shows processes used to evaluate VAN-Core system architecture (Phase 1: Step 1 to Step 8, Phase 2: Step 5 to Step 8). As Fig. 2 illustrates, our approach identifies and analyzes non-functional requirements and architectural approaches and then develops strategies for facing analyzed risks unlike iterations of Phase 1 and Phase 2 of ATAM.
Step 1
Present the ATAM

Step 2
Present the business drivers

Step 3
Present the architecture

Step 4
Identify and prioritize non-functional requirements

Step 5
Identify architectural approaches
Identify architectural alternatives

Step 6
Analyze architectural approaches
Analyze architectural alternatives

Step 7
Derive risk management strategies

Step 8
Present the results

Fig. 2. Architecture evaluation process for the VAN-Core system

Moreover, the purpose of Phase 2 is to develop and evaluate architectural alternatives for improving system by reflecting the risk management strategies of Phase 1. Phase 1 of our approach is mapped to Phase 2 of ATAM and the focus of Phase 2 is on recommending better architecture designs and evaluating them. As for differences in detail steps, firstly, our approach derives and prioritizes non-functional requirements in Step 4 and then maps the results to quality attributes unlike ATAM that generates quality attribute utility tree. It is difficult to generate a quality attribute utility tree through brainstorming or workshops under the situation that non-functional requirements are not defined and the stakeholders’ understandability of the system is not enough. Therefore, in this case study, we derived non-functional requirements by presenting and confirming non-functional requirements, cumulated from architecture design and analysis experiences in the financial sector, with stakeholders. Architectural approaches also derived from financial system’s architectural style pool that is a knowledge base of our team. Secondly, our approach has a step (i.e. Step 8) for establishing risk management strategies based on the evaluation results of architectural approaches. In case that the derived risk management strategies are decided enough, iteration is ended. Otherwise, Phase 2 is iterated. Thirdly, in case of iteration in Phase 2 or after that, our approach establishes action plans for improving architecture and derives (Step 5 in Phase 2), analyzes, and evaluates (Step 6 in Phase 2) architectural alternatives.

4. Architecture evaluation for the VAN-Core system

4.1 Phase 1

Step 1 ~ Step 3) Stakeholders of VAN-Core system tried to improve system architecture without changing the relevant business processes.

Step 4) Total 49 numbers of non-functional requirements were derived and high-prioritized requirements were related to performance, operability, security and safety, and usability. Fig. 3 is the results of deriving non-functional requirement of VAN-Core system.

Fig. 3. Non-functional requirements for the VAN-Core system
Step 5) According to the classification of Information System Architecture (ISA) [6], we distinguished VAN-Core system architecture into three viewpoints, application, data, and technological architecture, for deriving architectural approaches. Application architecture is a viewpoint of decomposing system’s application units and of defining interfaces and components among application units. Online and Batch sub-system were identified from an Application architecture viewpoint and 24 architectural approaches were derived. Data architecture, a viewpoint for defining data subject area of an enterprise and for identifying data models consisting with an organization’s business models and process models, identifies information structure of an organization. Master, Work Performance, and Business Code subject areas were identified from a Data architecture viewpoint. 17 architectural approaches were identified. Technological architecture is a viewpoint for identifying technical areas of an enterprise and for defining deployment diagram for an organization’s software, hardware, networks, and etc. Execution, Infra, and Deployment subject areas for the VAN-Core system were identified from a Technical architecture viewpoint and 25 architectural approaches were derived. Fig. 4 is a part of architectural approaches from an Application architecture viewpoint for Online subsystem.

Step 6) Identified architectural approaches were analyzed risk, non-risk, and trade-off for each non-functional requirement. In this case study, because the purpose of this project was to identify potential major risk factors we did not analyze sensitivity points.

AA-OL-01 Defining commonalities for the whole system and applications.
- Minimizing efforts necessary for developing newly, maintenance, and education because of reducing LOC by raising reusability level (achieving high reusability)

AA-OL-02 Defining redundantly similar transaction flow and business logic for each service.
- Developing and deploying independently for a specific work while minimizing impacts to other works

AA-OL-03 transforming messages and inputs/outputs by hard cording
- Achieving high speed through transforming messages and inputs/outputs by hard cording

Fig. 4. Sample architectural approaches for online subsystem

Step 7) The collected information from the ATAM is summarized and presented to the stakeholders. Stakeholders decided to delay the launching system in order to establish the strategies for managing derived risks.

Step 8) We decided and established risk management strategies (avoidance, transference, mitigation, acceptance) and concrete counter-measures after analyzing risk and its risk attribute (i.e. probability, risk level, risk management time). Fig. 5 is an example of risks and their risk management strategies for architectural approaches of application architecture. As described in Fig. 5, probability of risk R002 was H (Very Likely; probability of risk is more than 70%) and risk level was M (Critical; in case of occurring risk, system is impacted by the risk, but, it is possible to recover and it does not exceed the budget by 50%). Management time was analyzed as M (Mid-term; it means that risk is counter-measured within 4 months from as of present) and risk strategy was chosen A (Avoidance).

Together with the stakeholders we decided to conduct Phase 2 because architecture improvement for managing risks was brought up.
Considering risks with trade-offs

Analyzing characteristics of risks

Deciding risk management strategies and establishing detail measures

### Risk Management Strategy

<table>
<thead>
<tr>
<th>ID</th>
<th>Analyzed risk</th>
<th>ID</th>
<th>Analyzed trade-off</th>
<th>Probab.</th>
<th>Risk Level</th>
<th>Manage Time</th>
<th>Classification</th>
<th>ID</th>
<th>Identified strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>R001</td>
<td>In case of changing common components, this causes system deactivation and re-operation</td>
<td>T001</td>
<td>High reusability is risk as for reliability and availability, but is non-risk as for maintainability.</td>
<td>H</td>
<td>M</td>
<td>N</td>
<td>M</td>
<td>S001</td>
<td>1. Improving TANDEM receiving interface using database instead of hard-coding for minimizing deployment. 2. Distinguishing COMPANY BIN into service units. 3. Preserving current structure in case of credit-card merchant information, credit-card terminal information, and etc because of low changeability. 4. Dividing DII table and class into common parts (e.g. number and value of credit-card sales of bonus, membership, and gift card) and specific parts for each service.</td>
</tr>
<tr>
<td>R002</td>
<td>In case of changing message of credit-card company, this cause system deactivation and re-operations for re-deploying relevant component</td>
<td>T002</td>
<td>Hard-coded message transformation is risk factor as for reliability, availability, and maintainability, but is non-risk as for processing time and delay.</td>
<td>H</td>
<td>M</td>
<td>M</td>
<td>A</td>
<td>S002</td>
<td>1. Improving TANDEM receiving interface using database instead of hard-coding for minimizing deployment. 2. For resolving performance degradation problem, maximizing database IO performance through data caching and tuning for improving parameter loading performance</td>
</tr>
</tbody>
</table>

---

### Table 1. Risk, non-risk, and trade-off analysis results for architectural approaches

<table>
<thead>
<tr>
<th>Non-functional Requirements</th>
<th>System should be operational 24 hours a day and 365 days a year</th>
</tr>
</thead>
<tbody>
<tr>
<td>First-level QA</td>
<td>Performance</td>
</tr>
<tr>
<td>Architectural Approach</td>
<td>Analysis of Architectural Approach</td>
</tr>
<tr>
<td>ID</td>
<td>Approach</td>
</tr>
<tr>
<td>AA-OL-01</td>
<td>Defining commonalities for all system and applications</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>AA-OL-03</td>
<td>Transforming messages and inputs/outputs by hard-coding.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>


---

### 4.2 Phase 2

In Phase 2 we establish short-term, mid-term, and long-term action plan for improving system architecture based on risk management strategies and identifying architectural alternatives from them.

**Step 5** In this step we defined architectural alternatives from risk management strategies of step 8 in Phase 1. For example, approach AA-OL-03 was analyzed as Table 1, and according to risk management strategies of Fig. 5, alternative AA-OL-03-Alt-01 was suggested.

Parameter based message transformation using database is no need to change the relevant components whenever a message is changed or new one is added. We only add new values into database.

**Step 6** We analyzed risk, non-risk, and trade-off of architectural alternatives based on non-functional requirements. As a result of analysis “improving message transformation by hard-coding into parameter based transformation” alternative has performance degradation risk as illustrating in Table 2.
### Table 2. Risk, non-risk, and trade-off analysis results for architectural alternative

<table>
<thead>
<tr>
<th>Non-functional Requirement</th>
<th>System should be operational 24 hours a day and 365 days a year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural Alternative</td>
<td>Analysis Results for Architectural Alternatives</td>
</tr>
<tr>
<td>ID</td>
<td>Classifi-</td>
</tr>
<tr>
<td>Alternative</td>
<td>cation*</td>
</tr>
<tr>
<td>AA-OL-03-Alt-01</td>
<td>R</td>
</tr>
<tr>
<td>Using parameter based message transformation</td>
<td>T</td>
</tr>
<tr>
<td>N</td>
<td>Alt-N001</td>
</tr>
</tbody>
</table>

**Step 7** Analysis results for the suggested architectural alternatives were presented.

**Step 8** We established risk management strategies analyzed risks for each architectural alternative. For example, in the alternative AA-OL-03-Alt-01, what was suggested as a risk mitigation strategy was ‘Maximizing database IO performance through data caching and tuning for improving parameter loading performance’. Stakeholders decided to adapt the alternative. In this project, we needed to iterate Phase 2 only once.

### 5. Conclusions

Analytic Principles and Tools for the Improvement of Architectures (APTIA) [5], an agile method for combining architecture analysis and design method such as Architecture analysis and design methods such as ATAM and Cost Benefit Analysis Method (CBAM) [4], intends to improve system architecture by integrating progress of architecture evaluation, improvement suggestion, cost and benefit analysis, and architectural decision. For similar motivation, in this paper we modified ATAM process so that the modified process can suggest design alternatives and risk management strategies.

By applying our modified ATAM process we were able to evaluate and improve VAN-Core system architecture. In this project, it was difficult to derive quality utility tree or architectural approaches through interviews and workshops like ATAM because there were no pre-defined non-functional requirements and experienced customers in VAN-Core system. For resolving these situations, we used non-functional requirements tree and architectural style pool cumulated through projects in financial sector. Especially, because customers requested to establish risk management strategies for identified risks we changed ATAM so as to include these processes and produce relevant outcomes. In our approach we added a phase and steps for establishing risk management strategies and action plans and for identifying architectural alternatives while simplifying iteration of ATAM. Decisions for architectural alternatives are incorporated into the new system, which is operating successfully in an online credit-card transaction company.

### References


