Evaluating Application Architecture, Quantitatively

V. Gnanasekaran
Collabera, Bangalore, India

Abstract

This article describes how quantitative treatment can be applied to an application's architecture-evaluation process and shows how a quantitative output with intuitive reports will provide more clarity than a qualitative output on the quality of an application architecture.

"You cannot control what you cannot measure."—BILL HEWITT

Keywords: *quantitative treatment, architecture-evaluation, application architecture*

1. Introduction

Evaluation of an application architecture is an important step in any architecture-definition process. Its level of significance varies from organization to organization, based on a variety of factors (such as application size and business criticality). In some IT organizations, it is a part of a formal process; in others, it is performed only upon special requests that stakeholders might raise. Enterprises sometimes have a dedicated "Architectural Review Board" (or ARB) that is made up of a team of experienced architects who are earmarked for performing periodic architectural evaluations.

Scenarios that drive the architecture-evaluation process include:

- When a business must validate an application architecture to see whether it can support new business models.
- An expansion to new geographies and regions—resulting in the need to check whether an existing application architecture can scale to new levels.
- Impaired application performance and user concerns that lead to an assessment, to see whether it can be reengineered with minimal effort to ensure optimum performance.
- Stakeholders having to ensure that a proposed application architecture will meet all technical and business goals—ensuring that key architectural decisions were made with key use cases/ architectural scenarios in mind and will meet the nonfunctional requirements of the application.

In the context of the new application development, the key objectives of carrying out an architecture-evaluation process are:

- Avoiding costly redevelopment later in the software-development life-cycle (SDLC) process by detecting and correcting architectural flaws earlier.
- Eliminating surprises and last-minute rework that is due to the suboptimal usage of technology options that are provided by platform vendors such as Microsoft.

Architectural reviews are also performed based on only a particular quality-of-service attribute—such as "Performance" or "Security"—for example, how secure the architecture is, whether an architecture has the potential to support a certain number of transactions per second, or whether an architecture will support such a specified time.

The application architectural-evaluation process involves a preliminary review, based on a checklist that is provided by the platform vendor and subsequent presentations, debates, brainstorming sessions, and whiteboard

discussions among the architects. Key aspects of brainstorming sessions also include the outputs of the scenario-based evaluation exercises that are performed by using industry-standard methods such as the Architecture Trade-Off Analysis Method (ATAM), Software Architecture Analysis Method (SAAM), and Architecture Reviews for Intermediate Designs (ARID). There are also different methods that are available in the industry to assess the architectures, based exclusively on factors such as cost, modifiability, and interoperability.

The checklist that is provided by a platform vendor ensures the adoption of the right architectural patterns and appropriate design patterns. With its patterns & practices initiative, Microsoft provides a set of checklists/questionnaires across various crosscutting concerns for the evaluation of application architectures that are built on Microsoft's platform and products. An architecture-evaluation process usually results in an evaluation report that contains qualitative statements such as, "The application has too many layers" or "The application cannot be scaled out, because the layers are tightly coupled."

Instead of having qualitative statements, if the evaluation process ends up providing some metrics—such as a kidney-diagnosis process that ends with a "kidney number" or a lipid-profile analysis that ends with numerical figures for HDL and LDL—it will be easier for stakeholders to get a clear picture of the quality of the architecture.

This article outlines a framework for applying quantitative treatment to the architecture-evaluation process that results in more intuitive and quantitative output. This output will throw more light on areas of the application architecture that need refactoring or reengineering and will be more useful for further discussions and strategic decision making.

2. Background

Evaluation of an application architecture is equal to evaluation of the different architectural decisions that are taken as part of the definition of that application architecture. The objectives of architectural decisions can be viewed from multiple perspectives.

An architectural decision is taken for any of the objectives that are explained in the following list:

- **To adopt a best practice that suits a specific context**—Take, for example, a banking application that has been architected for Internet customers. In that context, to protect the application from hackers and malicious users, it is a best practice to keep the presentation layer in a separate tier in a DMZ, the business-logic layer in a separate tier, and the DB layer in another separate tier. An architectural decision to distribute multiple layers across different tiers is the adoption of this best practice.

- **To achieve a particular business goal**—Say that a publishing company has a business goal of increasing its sales volume by having an online order-acceptance facility, to allow customers worldwide to place an order. In this case, to achieve the business goal, the system should be built to make it highly available through an architectural decision of having a distributed architecture.

- **To achieve a desired level of a particular quality-of-service attribute**—In some scenarios, stakeholders might directly demand "Reliability" for a mission-critical application. In such cases, an architectural decision might be taken to have message queues and asynchronous communications as part of the architecture, so as to achieve a desired level in the "Reliability" quality-of-service attribute.

When an architecture decision is taken either to achieve a business goal or to adopt a best practice, it is implicit that it might have an impact on one or more quality-of-service attributes. In typical scenarios, the key quality-of-service attributes that will be in focus are "Scalability," "Security," "High availability," "Reliability," and "Performance"—also known as SHARP qualities.

Microsoft's patterns & practices resources that are specific to application architecture provide checklists/questions across these quality-of-service attributes and span multiple subcategories. These questions make the evaluation process simpler. Because these questions are the result of the collective experience of various experts from Microsoft, the performance of an architectural review that is based on these questions will definitely ensure that our application architecture is based on proven best practices, as well as architectural and design principles and standards.

While these review checklists/questions make our life easier, architects have to put effort into using them when they perform an application-architecture evaluation. Architects have to take printouts of these checklists/questions and conduct interview sessions with respective application architects, based on these checklists. Then, they have to perform some manual analysis/due-diligence process and arrive at an output.

Like medical reports that have clearly defined metrics that all doctors understand, if we want to have a clear quantitative output for an architecture-evaluation process, this will not be possible unless we have a framework that will help architects apply a quantitative treatment that is based on the checklists and generate outputs that will help architects and stakeholders immediately get a sense of the state of an application architecture.

Given this background, this article will outline a simple framework that can be used to carry out an architecture-evaluation process, based on the perspectives of adopting best practices and achieving a desired level in quality-of-service attributes.

3. Approach

There are two types of quality-of-service attributes: those that result in the runtime behavior of the system (such as "Performance," "Security," and "Scalability"—also known as runtime qualities), and those that can be evaluated only over the life cycle of an application (such as "Maintainability" and "Flexibility"—also known as design qualities). Usually, architectural evaluations focus more on runtime- quality attributes. The significance of the quality-of-service attributes that are considered for the architectural evaluation will vary, based on the context. For example, in line-of-business (LOB) applications, performance and scalability will gain more importance, while interoperability will become more important in heterogeneous environments.

The questions that are available from the Microsoft patterns & practices resources are the key input for this framework. They are elaborate and exhaustive, and they include questions that pertain to crosscutting concerns and platform-specific issues. These questions can be tweaked, so that the resulting repository can be used only for architectural evaluation. In the scenarios in which there is a need to evaluate application architectures in a heterogeneous environment, some platform-specific questions can be selectively dropped or replaced.

In fact, the questions and checklists that are available from the patterns & practices resources also include things that are applicable in technology-agnostic scenarios.
More categories and subcategories of questions can be added to the existing set, based on your experience; the greater the number of quality-of-service attributes that are covered by the repository, the wider the variety of applications on which evaluations can be performed. In the age of rich Internet applications (RIAs) and mashups, "Usability" is also gaining high importance on par with other key quality-of-service attributes. Figure 1 illustrates the quantification framework.

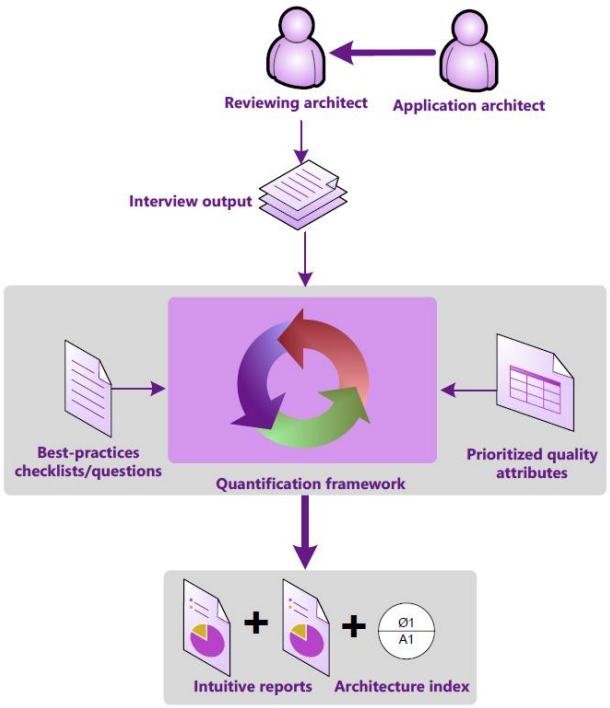

Figure 1. Quantification framework for architecture-evaluation process

The resulting repository will be a set of checklists that are based on the required quality-of-service attributes. These checklists can be used by reviewing architects to question the respective application architects. Also, answers for these checklists/questions can be extracted from documents such as a system-architecture definition and a solution-architecture definition. For every positive answer, a value of 1 can be assigned to each question, and a value of 0 can be assigned to a negative response.

After the completion of this probing process, and based on the number of positive responses, scores will be computed for all the quality-of-service attributes that are considered for evaluation. These scores are the summation of the scores that are available for each subcategory. The scores at the subcategory level are the summation of the ones that are allotted to each item/question in the checklist, as a positive response. Say, for example, that under the "Performance" attribute, we might have subcategories such as caching, data access, state management, resource management, and concurrency. Then, the result will be as shown in Table 1.

Performance	29
Caching	4
Data access	8
State management	5
Resource management	5
Concurrency	7

Table 1. Score for "Performance" quality-of-service attribute

Based on the actual number of questions that are available in the repository in each subcategory under the "Performance" attribute, we can arrive at a percentage that is scored against the "Performance" attribute for the application that is under review.

The same method can also be applied to arrive at percentage scores for other required quality-of-service attributes.

Now, you might think that the average of the scores across the different quality-of-service attributes will give an overall score that indicates the quality of an application architecture. However, that might not be the actual case.

Let us see why.

4. Architectural Trade-Offs

An application cannot score 100 percent across all quality-of-service attributes. Architectural definition is the result of the trade-off decisions that are taken across various quality-of-service attributes. These trade-offs are arrived at, based on the architecturally significant scenarios and nature of the business domain for which the application is developed. Also, one quality-of-service attribute can have either a positive or negative impact on other quality-of-service attributes.

	Availability	Efficiency	Flexibility	Integrity	Interoperability	Maintainability	Portability	Reliability	Reusability	Robustness	Testability	Usability
Availability								+		+		
Efficiency			-		-	-	-	-		-	-	-
Flexibility		-		-		+	+	+		+		
Integrity		-			-				-		-	-
Interoperability		-	+	-			+					
Maintainability	+	-	+					+			+	
Portability		-	+		+	-			+		+	-
Reliability	+	-	+			+				+	+	+
Reusability		-	+	-				-			+	
Robustness	+	-						+				+
Testability	+	-	+			+		+				+
Usability		-								+	-	

Table 2. Mutual impact of quality-of-service attributes

Table 2 provides an idea on the mutual impact that exists across different quality-of-service attributes. Because of an architectural decision to achieve a desired level in a particular quality-of-service attribute, another quality-of-service attribute could be adversely affected.

For example, in a banking application, security is considered to be more important than performance. The "Security" quality-of-service attribute will have a negative impact on the "Performance" quality-of-service attribute. So, any architectural decision to achieve a high degree of security will affect the performance of said application. This is a known trade-off decision that is intentionally taken; hence, the application that is under evaluation will score less under the "Performance" quality-of-service attribute.

To accommodate the trade-off decisions without affecting the final score and resulting in a misguided outcome, we have the concept of the *prioritization* of quality of service attributes. No application can have two mutually exclusive quality-of-service attributes at the same level of priority. For example, an application cannot have both "Performance" and "Security" as equal priorities. If "Performance" is the top priority for an application, "Security" automatically assumes a position in the next-available priority levels. If the evaluation of an application architecture is based on the SHARP quality-of-service attributes, and if the application is architected for a domain in which "Performance" is most critical and other attributes are of lower priority, the reviewing architect might assign priority numbers, as shown in Table 3.

Quality-of-service attribute	Priority number
Performance	5
Security	4
Scalability	2
High availability	1
Reliability	3

Table 3. Prioritization of quality-of-service attributes

Prioritization should be based on the business goals and input from stakeholders. It can also be achieved through the ATAM method. Use of ATAM ensures that business goals and stakeholder interests are taken into consideration. As a rule of thumb, the highest priority number should not exceed the number of quality-of-service attributes that is considered for the architectural evaluation. Also, no two quality-of-service attributes should have the same priority number.

Quality-of-service attribute	Priority number	Threshold (%)
Performance	5	100
Security	4	90
Scalability	2	70
High availability	1	80
Reliability	3	50

Table 4. Threshold numbers for quality-of-service attributes

As shown in Table 4, an architect can also assign threshold numbers against each quality-of-service attribute to indicate whether an application architecture scores below that number; before proceeding to the next stage, it is important to revisit the decisions under that quality-of-service attribute. These threshold numbers are subjective and should be based on a consensus that is agreed upon by a team of architects in the enterprise-architecture group.

If an application scores below the threshold values, it is a clear indication of the level at which the application architecture is below the mark.

This will also be especially helpful in mergers and acquisitions (M&As). Say that when Company A acquires Company B and carries out an assessment process, Company A might retire the applications that score well below the threshold values.

5. Architecture Index

After consideration of the scores for all quality-of-service attributes and prioritization of those attributes, the final quality of the application architecture can be arrived at by using the weighted-average formula, as shown in Table 5.

$$\text{Architecture index} = \frac{\begin{array}{l}\%\text{ score for Performance} \times \text{Performance priority number} \\ +\ \%\text{ score for Security} \times \text{Security priority number} \\ +\ \%\text{ score for Scalability} \times \text{Scalability priority number} \\ +\ \%\text{ score for High availability} \times \text{High-availability priority number} \\ +\ \%\text{ score for Reliability} \times \text{Reliability priority number}\end{array}}{\begin{array}{l}\text{Performance priority number} \\ +\ \text{Security priority number} \\ +\ \text{Scalability priority number} \\ +\ \text{High-availability priority number} \\ +\ \text{Reliability priority number}\end{array}}$$

Table 5. Architecture index through weighted-average formula

This weighted-average formula will result in a single number, which can be called the "Architecture index." Table 6 shows an architecture-index value that is based on the application of the weighted-average formula to the sample scores of different quality-of-service attributes, and their respective priority numbers.

Quality-of-service attribute	Priority number	Percentage gained (%)
Performance	5	86.30
Security	4	82.00
Scalability	2	77.00
High availability	1	59.00
Reliability	3	46.00
Architecture index		**74.03**

Table 6. Scores of quality-of-service attributes & corresponding architecture index

The architecture index will be between 0 and 100. This number gives an immediate sense of where that application architecture stands. Because the resulting number is based on the best practices and guidelines that are provided by platform vendors, it will reflect how best the application can be architected. For instance, an evaluation that is performed based on the checklists/questions that are provided by the Microsoft patterns & practices and results in a lower architecture index will indicate that the application architecture does not adhere to the proven best practices.

Because a positive or negative response to a question directly contributes to a score of a particular quality-of-service attribute, we can easily identify the impact of a particular architectural decision on a particular quality-of-service attribute and, hence, the overall quality of the application architecture.

6. Intuitive Reports

Although a single architecture index gives a clear view of the strength or quality of an application architecture, it must have some intuitive reports that highlight the weak areas of an application architecture, so that they can be used to carry out an effective reengineering or refactoring process.

It makes sense to have a tool or to build small software to automate the entire process and generate reports. Microsoft Office Excel can perform wonders, with few scripts and limited effort. For an application architect to know immediately what went wrong (based on the architecture index) and react immediately, these intuitive reports play a significant role.

Figure 2, and Figures 3 and 4, show screen shots of some of the reports that are generated by the tool and that resulted in our past successful architectural-consulting engagements.

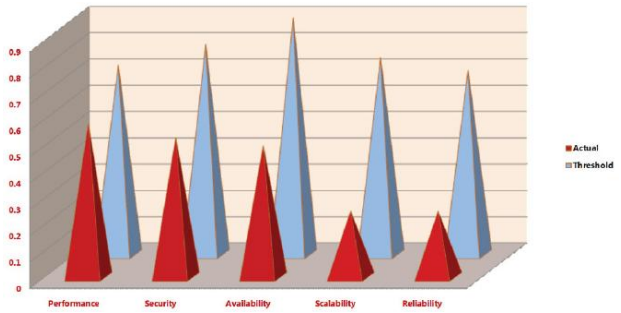

Figure 2. Overall-architecture quality of application

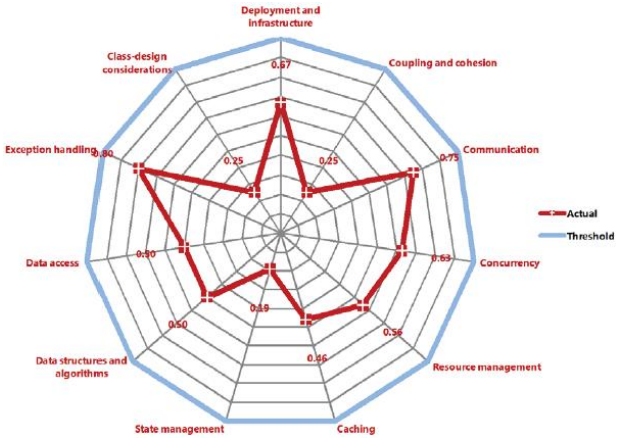

Figure 3. Quality of application architecture from perspective of "Performance"

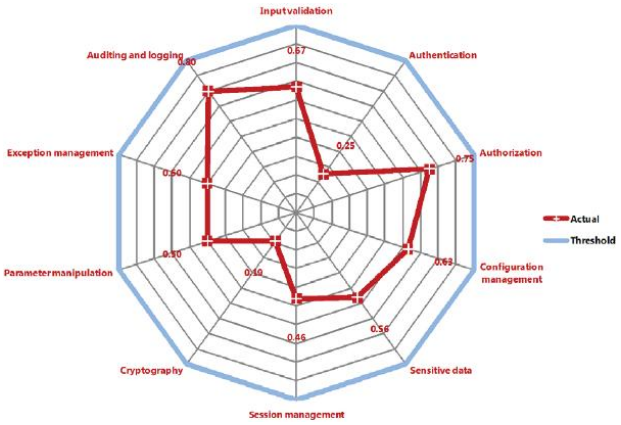

Figure 4. Quality of application architecture from perspective of "Security"

Say, for example, after an evaluation process, that an application architecture scores 49 percent. The application architect can immediately identify under which quality-of-service attribute it is scoring low. If it scored low in "Performance," the architect could go to the performance-analysis report, which will show the scores across different subcategories (such as caching and state management). If it scored less under a particular subcategory —for example, caching— the architect could trace back from that point to see why the architecture scored so many zeros under that subcategory. The architect could also get a handle on how a particular decision might affect a particular quality-of-service attribute and, hence, the overall architecture.

In scenarios in which the existing application architectures are evaluated, application architects can use these reports in meetings with stakeholders to convey why application architecture is considered inferior, as well as to highlight areas that need refocus. This will drive corrective actions that must be taken to revamp respective applications.

7. Conclusions

A quantitative architecture-evaluation process provides a crystal-clear picture of the quality of an application architecture. The output of this process helps in taking concrete, corrective decisions.

While the quantitative evaluation of application architecture is more promising and results in a clearer picture of the state of the architecture of existing applications or the proposed architecture of new applications that are to be built, it cannot replace an application-architecture process that is based on a scenario-based method such as ATAM. ATAM involves a more elaborate exercise that is based on architecturally significant scenarios and could be supplemented by a quantitative evaluation. While the output of a method such as ATAM is qualitative and based on scenario-based analysis, this framework- based evaluation output is quantitative and based on best practices and guidelines.

Let us go back to our inspiration: the "kidney number" or lipid-profile analysis. That is the key driver behind the conceptualization of this idea in applying a quantification treatment to the architectural-evaluation process. They have industry-standard benchmarks and ranges that are used as the basis to classify a particular patient.

Similarly, if platform vendors, service organizations, and enterprise IT teams work together to publish benchmark architectural indexes for applications, based on various

factors—such as business domain, architectural style and pattern, SLA requirements, and various combinations of quality-of-service attributes—they can be leading lights for building well-architected applications.

V. Gnanasekaran is a Senior Architect at Collabera in Bangalore, India. His areas of specialization include SOA/BPM, Integration, and Enterprise Architecture. He spends most of his time on Solution Architecture consulting, R&Ds on the latest technologies, and technology evangelism. Currently, he is focusing more on Cloud Computing and Enterprise Mobility.

Acknowledgements

Special thanks to Bala Variyam, CTO, Chander Damodaran, Senior Architect, and Sohail from Collabera for their reviews.

Resources

- [1] Morgan, Gabriel. " Implementing System-Quality Attributes." Microsoft Developer Network (MSDN) Architecture Center, March 2007.
- [2] Turner, Michael S. V. Microsoft Solutions Framework Essentials: Building Successful Technology Solutions. Redmond, WA: Microsoft Press, 2006.
- [3] Gorton, Ian. Essential Software Architecture. Berlin; New York: Springer, 2006.
- [4] Bass, Len, Paul Clements, and Rick Kazman. Software Architecture in Practice. Second ed. Boston, MA: Addison-Wesley, 2003.
- [5] Malcolm, Graeme, and Lin Joyner. Application Architecture for .NET: Designing Applications and Services. Microsoft patterns & practices Series. Redmond, WA: Microsoft Corp., 2002.
- [6] Meier, J.D., et al. Improving .NET Application Performance and Scalability. Microsoft patterns & practices Series. Redmond, WA: Microsoft Corp., 2004.
- [7] Microsoft patterns & practices Team. Microsoft Application Architecture Guide. Second ed. Microsoft patterns & practices Series. Redmond, WA: Microsoft Press, 2009.
- [8] Esposito, Dino, and Andrea Saltarello. Microsoft .NET: Architecting Applications for the Enterprise. Redmond, WA: Microsoft Press, 2009.