Scientific Research Publishing

Journal of Software Engineering and Applications, 2017, 10, 483-499
http://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Toward Quality Attribute Driven Approach to Software Architectural Design

Payel Bajpayee, Hassan Reza

Department of Computer Science, University of North Dakota, Grand Forks, USA
Email: payel.bajpayee@my.und.edu, reza@aero.und.edu

How to cite this paper: Bajpayee, P. and Reza, H. (2017) Toward Quality Attribute Driven Approach to Software Architectural Design. *Journal of Software Engineering and Applications*, **10**, 483-499.
https://doi.org/10.4236/jsea.2017.106027

Received: November 25, 2016
Accepted: June 13, 2017
Published: June 16, 2017

Abstract

It has been well-documented that the software architecture of any system plays a critical role in success or failure of software intensive systems. In this paper, a method has been proposed to evaluate the software architecture's fitness with respect to key quality attributes for a web-based system. To the end, a comparative analysis based on quality attributes scenarios and tactics is carried out to select an optimal software architecture that meets the system level requirements of a web-based system, namely, Student and Course Evaluation System (SCES). The comparative study was driven by study of quality attributes and tactics with the selected architectures to select the optimal one.

Keywords

Architectural Tactics, Quality Attributes, Software Architecture, Architectural Styles

1. Introduction

A useful system can achieve the functional requirements with the analysis and coding structure but the non-functional qualities such as availability, performance, modifiability, security can be accomplished with the proper architecture. However, design will remain the focus of software engineering. Herb Simon, in his classic, The Sciences of the Artificial [1], includes a discussion of design in the context of "artificial" fields, such as software development, saying: design will remain the center of any development process.

Put in software engineering parlance, the outer environment is the world of requirements, goals, and wants; the inner environment is the set of software languages, components, and tools that have been used for building systems. Software Engineering pushes the barriers for creation of new echelons of infrastructure on which new developments may be constructed. In Simon's terms, the

"inner environment" or the "means" is constantly evolving. A design method not only provides a systematic approach to organize and structure action, but also to cater towards making choices based on criteria.

Among the myriads of methods available for determining the existence of a specific quality, they can be determined individually. In reality the attributes of a system are interactive and dependent on each other. Availability impacts security, which in turn affects performance, which bears upon modifiability. While still on the planning stage, the designer needs to keep in mind that all the attributes are interlinked and prioritizing may have an adverse effect as testing the quality attributes is difficult once the system is built.

To start with the architectural trade-off for web based systems, this paper has taken Student Grading and Evaluation System (SGES) as a simple example. Based on the requirement gathering and analysis for the mentioned system, the suitable candidate architectural design will be analyzed with respect to its availability, security, and performance attributes.

Student and Course Evaluation System (SCES) is a unified system that provides a single point of access to all secure registered students in Higher education sector. The system includes student registration and enrolment, student data, course work, exam information, program information, student growth rate, grade history and course evaluation. The objective is to build a transaction processing system that serves at the operational level of course evaluation in the colleges and universities. Though the campus or course information system traditionally being mainframe applications, they have been evolving since the late 90s and are widely adopted through the presence of a web-based medium as a hassle-free arrangement for viewing relevant information.

As cited following, construction of standalone software is challenging due to several constraints. There are considerable amount of issues with these systems based on design principles and tactics like security, maintainability, performance and most importantly usability. These issues include, but are not limited to,

- User specific accessibility of a generic web system.
- Degradation of the performance of the system while handling of high volume transactions.
- Expensive and complex upgrades for continued support, which maybe improbable for the required improvement of the system.
- Inflexibility partly due to monolithic design, requiring someone to be involved with each change in the system requirements.
- Involvement of large logical comprehensions to support key business activities.
- Non-compliance with other software applications used by the same client or user.

Final and foremost is the usability of the system. The interactive nature of the system will determine the ease at which the user will be able to accomplish the desired tasks and function.

In order to address the above issues, this paper will evaluate different archi-

tectural models by analyzing the quality attributes, the tactics and the comparative study on the tactics to implement the quality attributes to achieve the most suitable architecture for the software. It will identify the pros and cons of all universally accepted and suitable architectural styles and patterns and propose the most suitable one. To perform trade-off analysis of architectures for Student Grading and Evaluation web based System, this paper will analyze the usability and security attributes of the system and come to conclude on a suitable architectural style.

The design will be such that, by subsequent modification it will be limited as far as possible to least cost effect components and would not result in chain reaction of compensating modification. This will make it easier to add more functionality for future requirements. This paper describes a prototype implementation and software architecture. It furthermore discusses the key properties of the system and compares it with other available systems and finally outlines the plans for future research.

Background

Building an enterprise-scale software system is a complex undertaking. Despite wide scale technological advancements, the demands of present day information systems often stretch the limits of a company's ability to design, construct, and evolve its mission-critical software solutions. Noteworthy is the fact that a few systems in particular, are designed from the ground up. Rather, a software architect's task is that of extending the life of an existing solution by describing new business logic that manipulates an existing repository of data, presenting existing data and transactions through new channels such as an Internet browser or handheld devices, integrating previously disconnected systems supporting overlapping business activities [2].

The web based grade evaluation system provides the instructor with a response to the level of understanding for the students and also the standard of course distribution. This will provide a standard for the entire education system and make it process accurate. Thus, the usability and security of the system plays a key role in this case. The success of the system entirely depends upon how effortlessly the user can handle the software and make the most use of it.

Related work

Before making a comparative analysis of the system architecture, an elaborate study on the diverse architectural methods and various web based educational software is essential.

Kazman *et al.* developed the architectural trade off analysis method (ATAM) which evaluates quality attribute requirements by examining the consequences of architectural decisions [3]. Bengtsson *et al.* introduced the architectural level modifiability analysis (ALMA), a scenario based software architecture analysis that focuses on the modifiability quality attribute [4]. Knodel [1] proposed Rapid Architecture Evaluation (RATE) that works on architectural scenarios that are used to identify risk, severity points, strengths, weakness, and tradeoff.

Design decisions are often made for non-technical reasons, viz. strategic

business concerns, and meeting the required cost and scheduling, optimal use of manpower, and so on. Having a structured method helps to ensure questioning in the initial requirement and design stages when discovered problems can be solved in an inexpensive manner. It guides users about the method—the stakeholders—to look for conflicts in the requirements and for resolutions of these conflicts in the software architecture. In realizing the method, assumption is that attribute-specific analyses are interdependent, and that each quality attribute has connections with other attributes, through specific architectural elements. An architectural element is a component, a property of the component, or a property of the relationship between components that affects some quality attribute. For example, the priority of a process is an architectural element that could affect performance. The tradeoff analysis, often regarded as trade off points, helps to identify these interdependencies [3].

Among the myriad of Student Management Systems available, most are based on client server architecture and a few on cloud computing for easier storage and manipulation of data. In addition to understanding the demand of the educational system, the analytical study also provides a better perspective of the requirement, which is one of the main criteria for the tradeoff analysis suggested by Kazman [5]. Among the widely used and various grade evaluation systems, a few can be shortlisted:

- **Moodle**: It is a free and open-source software learning management system written in PHP and distributed under the GNU General Public License [Source: Wikipedia.org]. Developed on educational principles, Moodle is widely used for blended learning, distance education, and other e-learning projects in various institutions.
- **Renweb**: It is complete web based software providing integrated solutions to automate school administration and classroom management. This web-based solution also offers data conversion and setup process, comprehensive training, as well as live Customer Support.
- **ACTIVE Educate**: ACTIVE Network provides web-based registration and management solutions. It enhances efficiency with online registration, process tuition and fees including credit card processing and manages private lessons, streamline administration and staff management.
- **Blackboard**: It is independent school management software, which provides its customer to share student records across admissions and registrars. It also helps in organizing the admission process, store detailed student information, create schedules, produce demographic/analytical reports, and print report cards and transcripts. Likewise, it shares student grades, attendance, and exchange information securely via the web.

Based on the above mentioned study and the entire collected scenario, the main features provided by almost all the above mentioned software and other software(s) available widely are:

Academic Reporting, Admission Management, Attendance Tracking, Event Calendar, Grade Management and Display, Scheduling and Maintaining Record.

This paper tries to provide a manner that will help in improvement of the course, making the coursework easier for the students' understandability and make the instructor aware of the students' capabilities and most importantly in-capabilities. Based on the principal approach and essential requirements of a web based grade evaluation system as per the identified quality attributes, the following section will not only identify the candidate architectural designs but also will trade off them based on the identified quality attributes.

2. Research Approach

To build an efficient and useful web based system, the following section will analyze existing architectural design approaches. The section provides a comparative study of the various systems such as Blackboard architectural, Model-View-Controller, Client/Server architectural styles and their variations.

2.1. Architectural Designs

Blackboard Design:

The blackboard architectural pattern provides a computational framework for the design and implementation of systems that must integrate large and diverse specialized modules, and implement complex, non-deterministic control strategies. Using this style, several specialized subsystems known as agents are assembled to build a possibly partial or approximate solution [2].

As shown in Figure 1, the blackboard model consists of three main components as follow:

- The blackboard is a structured global memory containing objects from the solution space. These objects, also called blackboard nodes (or hypotheses), are hierarchically organized into several levels of analysis and can be linked with each other.
- Knowledge sources can be seen as highly specialized modules with their own representation. They are characterized by a set of triggering conditions and an executable code that retrieves data from the blackboard, which in turn contributes to the problem solving process.
- The control component selects, configures and executes knowledge sources. Determination of executable knowledge sources is based on the state of the problem solving process as expressed in the blackboard.

The blackboard model originally emerged in the domain of artificial intelli-

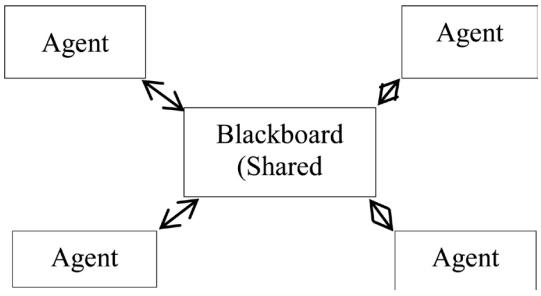


Figure 1. Blackboard architecture.

gence [6]. It is based on a metaphor where a group of specialists gathers around a blackboard and solves a particular problem in cooperation. The blackboard is a shared repository containing the data from various problem solving states [7]. The domain knowledge is partitioned into knowledge sources which read data from and write data to a blackboard. Knowledge sources are independent because they don't communicate with each other directly or are unaware of the presence of other sources [1].

Model-View-Controller Design:

The structure of the MVC pattern consists of three components namely the Model, View, and Controller. The Model provides functional core of an application and notifies views about the data change. Views retrieve information from the Model and display it to the user. Controllers translate events into requests to perform operations on View and Model elements [8].

The following are the three stereotypes from the existing set of design elements:

- Model: A stereotype that extends the Component metaclass of UML and attaches ports for interaction with the Controller and View components.
- Controller: A stereotype that extends the Component metaclass of UML and attaches ports for interaction with the Model and View components.
- View: A stereotype that extends the Component metaclass of UML and attaches ports for interaction with the Model and Controller components.

As shown in **Figure 2**, the Controller receives input and translates it into requests to the associated model using the Control primitive, while the Model calls back View when a specific data change occurs.

Client Server Design:

The Client/Server computing is an environment that satisfies the business need by appropriately allocating the application processing between the client and the server processors. The protocol is that the client requests the services from the server; the server processes the request and returns the result to the client:

- Client: A client is a single-user workstation that provides presentation services and the appropriate computing, connectivity and the database services and the interfaces relevant to the business need.
- Server: A server is one or more multi-user processors with shared memory

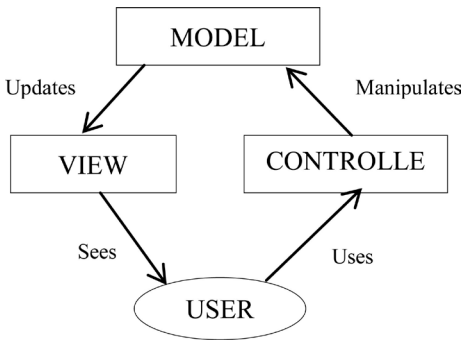


Figure 2. Display the three tier architecture of MVC pattern.

providing computing, connectivity, database services and the interfaces relevant to the business need.

The client/server computing is fundamentally platform independent. The user of an application wants the functionality (business) it provides, while the computing platform provides access to this business functionality. There are no benefits but poses a considerable amount of risk of exposure of the platform to the users. It is also easily demonstrable, *i.e.* the developers become aware of the target platform and development will be bound to that platform. They will use special tricks and features found in that specific platform.

Further discussion is on architectural designs like Cloud Computing, Web design and Service oriented architectures. These architectures have been built on the concept of Client-Server Design but are either different or advanced from Client server basic design or one another. What follows is brief discussion on these architectural designs and later the differences.

1) Cloud Computing Design:

Cloud computing system can be divided into two sections [9]: The front end and the back end. Both sections relate to each other through a network, usually the internet. Front end is what the client (user) sees whereas the back end is the cloud of the system. Front end has the client's computer and the application required to access the cloud and the back has the cloud computing services like various computers, servers and data storage.

Monitoring of traffic, administering the system and client demands are administered by a central server. It follows certain rules and protocols and uses special software called the middleware [9]. Middleware allows networked computers to communicate with each other. A cloud client consists of computer hardware and/or computer software which relies on cloud computing for application delivery, or specifically designed for required cloud services [5].

A cloud application delivers "Software as a Service (SaaS)" over the internet, thus eliminating the need to install and run the application on the users' system [6]. Important characteristics of this are: [6] Network-based access and management of commercially available software that are managed from centralized locations and enable customers to access these applications remotely through the internet. Examples of the key providers are SalesForce.com (SFDC), NetSuite, Oracle, IBM and Microsoft [10]. Google Apps is the most widely used SaaS.

Platform services or "Platform as a Service (PaaS)" provide a computing platform using the cloud infrastructure. It has all the applications deployed on it which are typically required by the client. Thus the client need not go through the hassles of buying and installing the software and hardware required for it. Through this service developers can get a hold of all the systems and environments required for the life cycle of the software, be it developing, testing, deploying or hosting of web applications. Key examples are GAE, Microsoft's Azure [10].

Infrastructure services or "Infrastructure as a Service (IaaS)" provides the required infrastructure as a service. There is no need to purchase the required

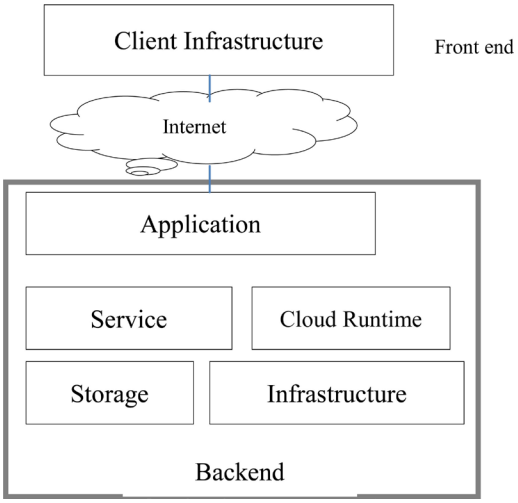


Figure 3. Display of the architecture of Cloud Computing.

servers, data center or the network resources by the client. Also, the key advantage of this application is that, customers need to pay only for the time duration of using the service. As a result they can achieve a faster service delivery with reduced cost. Examples are GoGrid, Flexiscale, Layered Technologies, Joyent and Mosso/Rackspace. Find the cloud computing architecture in **Figure 3** [10].

Difference with Client Server: It is a type of Client Server architecture where the server is in cloud and can be accessed by various clients. This characteristic makes the integration most effective and geographically independent. Hence for the course evaluation system, cloud computing architecture separately from CS architecture has higher consideration.

2) Web based Design:

A web-based application is any application that uses a website as the interface (*i.e.*, front-end). Users access the application from any computer connected to the Internet using a standard browser, instead of using an application that has been installed on their local computer [4].

Almost any desktop software can be developed as a web-based application. To build a bridge between Web and enterprise database, a number of alternative technologies and architectures have been available. These technologies include:

- CGI (Common Gateway Interface), a Web standard for accessing external programs to integrate databases with Web servers. The CGI dynamically generates HTML documents from back-end databases;
- Web server APIs, such as Microsoft's Information Server API (ISAPI), Netscape API (NSAPI), are invoked by third party software to access remote databases;
- Web-ODBC (Open Database Connectivity) gateways rely on an open API (Application Programming Interface) to access database systems;
- Vendor-specific Web browser/data warehousing interfaces are in response to the inherent advantages of the two technologies;
- JDBC (Java Database Connectivity) is used in its Java programming language to program Java applets to access back-end databases [1].

Difference with Client Server: Again this architecture is an advancement of CS architecture where the web application works as client from where user can access data. The server side applications get installed on the local computer which access the database and provide response to the web base users. Web base applications are simple, user-friendly and widely used for small, cost effective applications. This is the main reason for considering this architecture as an important option for the application.

Service-Oriented Architectural (SOA) Design:

A way of designing a software system is to provide services to either end-user applications or other services through published and discoverable interfaces. In many cases, services provide a better way to expose discrete business functions and therefore an excellent way to develop applications that support business processes. Service-oriented architecture is not a new notion; it is important now because of the emerging Web services technology.

For the purposes of this document, consider the following definition of a service: "A service is generally implemented as a course-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled (often asynchronous), message-based communication model." *In* many ways, the terminology for services is much the same as the terminology used to describe component-based development; however, there are terms used to define elements within Web services, as shown in Figure 4 [2].

2.2. Approach to Select the Architecture for Student Grade System

Based on widely used architectural designs and features, the following section

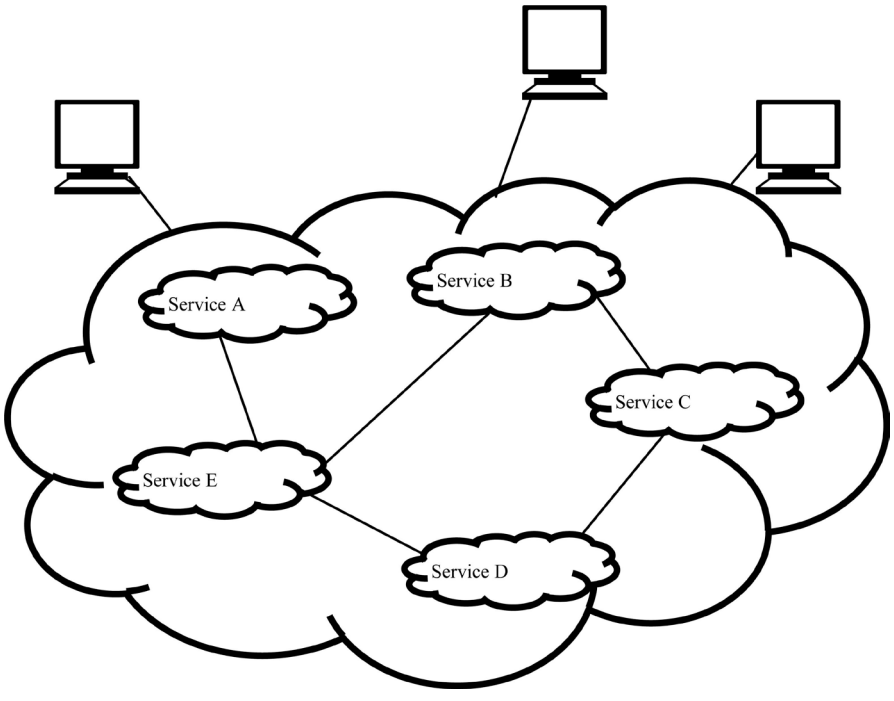


Figure 4. The service oriented architectural pattern.

will elaborately discuss on the candidate architectural systems (like, Black Board Architecture, Client Server Architecture, and the Service-oriented Architecture) with respect to usability and security quality attributes and their tactics. The next section will do a comparative analysis on the above described candidate architectural systems to come to a conclusion for the most suited architecture of a specific web based system by following the steps below:

- Requirement analysis and prioritization with help of Utility tree inspired by Bass *et al.*
- QA-Tactics table to study particular quality attribute based problems and the how Tactics can be used to identify probable solutions.
- QA vs. QA analysis to identify the issues may occur to achieve multiple quality attributes all together.
- QA vs Tactics table to identify the compatibilities for the tactics with identified QAs for the Student grading and evaluation system.
- Tactics vs Tactics table to identify the compatibility and difficulties to achieve several tactics at same point of time.
- Finally the Software Architecture vs Tactics table to identify the difficulty level to achieve the identified tactics for the candidate architectures, based on each SA's design and limitations.

3. Comparative Study and Results

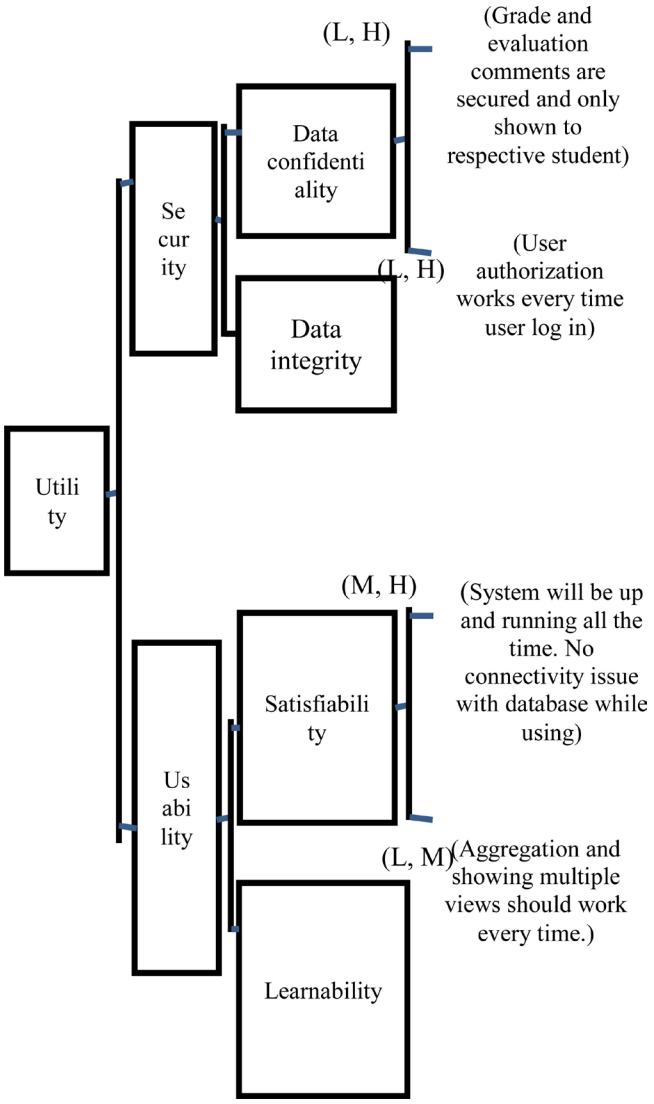
The functionalities (or capability) of a system under construction does play a key role in the selection of appropriate design [3]. For example in a grade evaluation system, the system provide a blue login button in the home page. This functional requirement can be achieved even with one html page and direct calling of the database to store or retrieve the login detail of the user. The architecture comes in place when there is a need to provide answered to the following design questions that impact overall quality and integrity of a system. Examples of questions may include:

- How much time it should take for a user to log in?
- What if in the browser another user is already logged in and again hit the login button with another user detail?
- What if system crashed on clicking on the login button, how long should it take to repair?
- How easy will it be for one new user to find and understand the functionality of it?

Thus the quality attributes are called upon. For one web based education application it is really important to make the system:

- Easy to use, really secured as the system deals with sensitive data like grades.
- It will always be accessed either by the students or instructors or the administrators—hence maintainable.

The following section will study on the quality attributes for scenarios that may occur to the system and the tactics that can be used to implement them. The utility tree on Security and Usability attributes and the requirements scenario



L: Low, H: High, M: Medium

Figure 5. Utility tree on usability and security QAs inspired by Bass *et al.* [3].

under the mentioned QAs is shown in Figure 5. The parameters (L, M) indicate that (Difficulty level identified during design, Actual difficulty level during implementation). So the Data confidentiality achievement for a secured application though seems achievable for Student grade system, but the secured connection, user validation, data confidentiality management is time and effort consuming during implementation [3].

3.1. Usability Tactic Analysis

Usability is concerned with how comfortable it is for the user to accomplish a desired task and the kind of user support the system provides. The Table 1 discusses about the usability tactic scenario and the tactics that can help to overcome specific situations.

Tables in this section are structured to analyze the issues may occur to a certain type of users while using the application. Further, the tables propose the ex-

pected response, tactics, which can be used to implement the solution. For Usability and Security QAs, the tables have been created to identify the problem and feasible solution of the problem using certain tactics [3]. For example, the administrator might get disconnected while adding new entry at runtime. The system should have mechanism to cancel, undo the process if the entries are not completed and it also should reuse the previous entered data to better assist the user (see Table 1).

Like the template presented in Table 1, Table 2 captures another design problem that may occur while using the system and how the usability attribute and its associated tactics can help to mitigate this usability problem.

3.2. Security Tactic Analysis

Another very important quality attribute is Security. As the system is Grade Evaluation web based system which deals with highly sensitive data like grades, the security is a big challenge. Based on the overview of security tactics introduced by Bass [3], they are divided into three distinct categories:

- Resisting attacks covers security measures which can be applied in order to prevent attacks. This tactic addresses the confidentiality and integrity security attributes of a system.
- Detecting Attacks and Recovering from an Attack aim at handling successful.

Table 1. Usability tactics analysis on user entered data reuse.

Usability QA	
User	Administrator of the system
Problem	Disconnected while adding new entry
Environment	Runtime
Artifact	System
Response	Reuse of already entered data
Tactic	**Cancel, Undo, User Model**
Rationale	The user may cancel or undo the previous entered data and the system should assist the user

Table 2. Usability tactics analysis on aggregation.

Usability QA	
User	Instructor of the course
Problem	Issue while comparing the grades of students in one class
Environment	Runtime
Artifact	System
Response	Interface is familiar/usable to the user
Tactic	**Aggregation, Show multiple views, task model**
Rational	Provide a clear interface to the user along with some helps so that new user should know how to use and do.

attacks, where recovering from an attack focuses on availability issues of a system.

- User authentication, authorization, providing well-defined, authorized data, protection of data, Session termination, Limited Access, Protected Access play a vital and essential role for such kind of web based system.

Table 3 represents a scenario in which the system administrator addresses security risks issues such as, attempt to change grades and attempt to read the assignments before final exam. The system should provide mechanisms to control access to the sensitive data. These safeguards include authentication to support confidentiality of data and authorization to support integrity of data.

Table 4 captures yet another problem that can occur while using the system and how the security attribute and associated tactics like limited exposure, limited access can help the system to overcome the problem:

- Restoration may conflict with maintaining Data Confidentiality if multiple copies of a system are to be maintained.
- Limit Access can be applied in that case to provide limitation to the attack surface.

Quality vs. Quality

Many tactics can be replicated using the architectures that are available and vastly used in software engineering. To draw a firm conclusion on suitable architectural design, Table 5 will do the next level of comparative study to analyze

Table 3. Security tactics analysis on unauthorized access.

	Security QA
User	System Administrator/Students/Hacker
Problem	Attempt to change the grades, attempt to check the assignments
Environment	Normal Operation
Artifact	System's database
Response	Grant or withdraw permission to modify the data
Tactic	**Authenticate user, Authorize the user, Maintain Data Confidentiality, Maintain Integrity**
Rationale	Modification can only be done by authorized users

Table 4. Security tactics analysis on an unauthorized access.

	Security QA
User	Hackers
Problem	Making the system unresponsive
Environment	Normal Operation
Artifact	System
Response	Detect attack, notify administrator, make some service disable to make the system stable
Tactic	**Limit Exposure, Limit Access, Intrusion Detection**
Rational	Detect the attack, prevent the attack

Table 5. Quality vs. Quality.

QAs	Usability	Security
Usability	NA	2
Security	2	NA

4-5: Very Strong, 3: average, 2: low, 1: poor

QAs vs QAs. In case of Student grade evaluation system Usability and Security has been taken as important QAs, hence Table 5 will check the compatibility of the mentioned two QAs before doing the comparative study among QAs and Tactics.

Based on Kazman *et al.* [3] who proposed the architectural trade off analysis method (ATAM)—achieving usability and security attributes together is very difficult; based on the requirement, one can be compromised to achieve another to some extent. Hence the compatibility is low among the mentioned QAs.

As shown in the Table 5 (compatibility chart) has weighed 2 for Usability-Security correlation. The sub attributes of the usability are: satisfiability (satisfactory user experience), memorability (login credentials are easy to memorize). A Highly secured system will have multiple authorization and authentications steps to login. Examples of these steps may include RSA encryption login, secret questions validation, lengthy and complex password which may affects the sub attributes of usability such as memorability and ease of use.

Similarly, the sub attributes of Security are: confidentiality (data protection from unauthorized disclosure), Integrity (data and process protection from unauthorized modification), Availability (protection from denial of service of an authenticated user)—to maintain the confidentiality and integrity, the satisfiability or enriched user experience of the application get compromised. As such, the compatibility is below average; this has been represented by the weight in Table 5.

Quality vs. Tactics

Tactics are a set of well-known and proven design decisions [3]. As the quality attributes are impossible to achieve in isolation, the same is true for the tactics because tactics for all the mentioned qualities will similarly correlate each other in positive or negative ways. The set of important tactics for the grade evaluation system discussed can be:

- Support User Initiative (SUI)
- Support System Initiative (SSI)
- Modifiability (Mod)
- Understandability (Und)
- Persisting Attacks (PA)
- Detecting Attacks (DA)
- Recovering From Attacks (RFA)

In what follows, we discuss the correlation between the quality attributes and tactics. As per the Table 5 the Support User Initiative, Support system initiative, Modifiability tactics defines usability QA, hence in Table 6 the weightage is very

strong (4 or 5), whereas as the Security tactics like: Persisting attacks, detecting attack and recovering attack has low or poor weightage with usability (1 or 2). Similarly the usability tactics get below average (1 or 2) with security attributes whereas Persisting attacks, detecting attack and recovering attack has strong weightage (4 or 5) with security.

Making a system secure requires (at least) considering tactics such as detecting or persisting attacks. But making the system usable at the same time to support user initiative is where the problem lies.

Tactics vs. Tactics

Based on analysis so far: To achieve "Support user initiative" tactics along with Supporting system initiative will be easier since both are usability tactics and achievable in isolation, hence in Table 6 the weightage shown as very strong (4-5). "Similarly User initiative support", modifiability and understandability goes hand in hand as the more the system supports the initiative taken by the user the more the system is understandable to the user and also can be enhanced- hence the correlation among them shown as very strong (4 or 5).

Table 7 will discuss the correlation between these tactics. In Table 7, we used the same scale used in the Table 6. Understandability is achievable even when the system is highly secured or the system is easily usable hence the weightage for understandability with other tactics mentioned as 3. Whereas the SUI directly opposes the attacks persistence, detection and recovery as to make the system highly secure, user initiation has to be restricted and closely monitored. So in Table 6 the correlation among them are below average (1-2).

Table 6. QAs vs. Tactics.

QA vs. Tactics	SUI	SSI	Mod	Und	PA	DA	RFA	Total (35)
Usability	4	5	4	5	2	2	1	23
Security	2	2	1	3	4	4	5	21

4-5: Very Strong, 3: average, 2: low, 1: poor; Support User Initiative(SUI), Support System Initiative (SSI), Modifiability (Mod), Understandability (Und), Persisting Attacks (PA), Detecting Attacks (DA), Recovering From Attacks(RFA).

Table 7. Tactics vs. Tactics.

Tactics	SUI	SSI	Mod	Und	PA	DA	RFA	Total (35)
SUI	NA	4	4	4	2	2	1	20
SSI	5	NA	4	4	2	1	1	17
Mod	4	4	NA	4	1	2	2	17
Und	4	4	3	NA	3	3	3	20
PA	2	1	1	3	NA	4	4	15
DA	1	2	1	4	5	NA	3	16
RA	1	1	1	3	5	5	NA	16

4-5: Very Strong, 3: average, 2: low, 1: poor; Support User Initiative(SUI), Support System Initiative (SSI), Modifiability (Mod), Understandability (Und), Persisting Attacks (PA), Detecting Attacks (DA), Recovering From Attacks(RFA).

Tactics vs. Architectural Designs

As per Bass [3] and Cysneiro [7], the tactics can be utilized to build the quality attributes of a system and can be utilized to achieve the most suitable architecture. Hence the Table 8 will finally do a comparative study on tactics as has been discussed in this paper with the architectural designs and will search for suitable design achievable:

Blackboard architecture [7] allows the progressive definition and implementation of the different agents, which can also be reused hence achieving modifiability and understandability tactics is achievable, hence weighted as 3 and 4. As a drawback, the centralization of the data in the Blackboard [Figure 1] can negatively affect the security, performance, and maintainability. Hence for security QAs tactics weightage is low (1 or 2) for blackboard architecture in Table 8.

MVC [8] has more to do with reducing code complexity, increase reusability hence achieving the user or system initiative support or making the system understandable or modifiable is achievable with MVC and therefore weighted between average to good (3 or 4) but security is a drawback of MVC hence in Table 8, security related tactics rated low (1 or 2) for this architecture.

Client Server Architecture [9] centralized, back-up and data recovery is achievable, scalable, and accessible. Hence the user initiative, system initiative, understandability and modifiability have been weighted as average or above average (3 or 4). Whereas on the other hand CS is: moderately secure, congestion prone not so robust and highly expensive. The tactics for security are rated low (2) per the mentioned behavior of the architecture.

Finally as per the last row of the Table 8, SOA [2] receives high score on reusability, maintainability, greater reliability, location independence, scalability, availability and most effectively on platform independence but the major drawbacks are increased overhead and high investment costs. Therefore, the tactics weightage has been provided on the plus points of this architecture, because average or above average (3 or 4) were used.

4. Conclusion and Future Works

In this work, a desire motivated the architectural study to make rational choices among competing architectural designs based on well-documented analyses of system attributes, and tactics [3]. This analysis also serves as a means of trans-

Table 8. Tactics vs. Architectural design.

Tactics vs SA	SUI	SSI	Mod	Und	PA	DA	RFA	Total (35)
Blackboard	2	2	3	4	2	1	1	15
MVC	3	3	4	3	2	1	1	17
CS	3	3	4	4	2	2	2	20
SOA	3	3	4	3	3	3	3	22

4-5: Very Strong, 3: average, 2: low, 1: poor; Support User Initiative(SUI), Support System Initiative (SSI), Modifiability (Mod), Understandability (Und), Persisting Attacks (PA), Detecting Attacks (DA), Recovering From Attacks(RFA)

portation for the early clarification of requirements.

Considering the comparative study on the grade evaluation system, it began with vague requirements and enumerated few architectural alternatives. The analytical framework and study of the quality attributes together with tactics helped to identify the essential features of each of design alternative and highlighted the costs and benefits associated with the architectural features. In this work, we applied the modified version of quality attribute scenario and tactics to identify the optimal architectural style [10] [11] to develop a user-friendly web-based system. Toward this end, the paper has identified Usability and Security as one of the key quality attributes. Other attributes like Deploy ability, Scalability, and Maintainability are also equally important.

Currently the further evaluation of Service Oriented Architecture is in progress to find out and validate the usefulness of the approach. More experimentations and case studies are needed to establish the practicality of attribute driven and tactic to select the proper design.

References

[1] Knodel, J. and Naab, M. (2014) Mitigating the Risk of Software Change in Practice—Retrospective on More Than 50 Architecture Evaluations in Industry. *IEEE CSMR*-18/*WCRE*-21 *Software Evolution Week*, Antwerp, 3-6 February 2014, 2-17.

[2] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M. (1996) *Pattern-Oriented Software Architecture Volume* 1: *A System of Patterns*, Wiley & Sons, New York.

[3] Bass, L., Clements, P. and Kazman, R. (2012) Software Architecture in Practice. 2nd Edition, Addison-Wesley Professional, Boston.

[4] Zhao, W. (1999) A Study of Web-Based Application Architecture and Performance Measurements. School of Information Systems Curtin University, Perth.

[5] Giesecke, S., Marwede, F., Rohr, M. and Hasselbring, W. (2007) A Style-Based Architecture Modeling Approach for UML 2 Component Diagrams. *IASTED International Conference Software Engineering and Applications*. Cambridge, 19-21 November 2007, 530-538.

[6] Barr, A., Cohen, P. and Feigebaum, E.A. (1989) Handbook of Artificial Intelligence. Addison-Wesley, Boston.

[7] Simon, H.A. (1981) The Sciences of the Artificial. 2nd Edition, The MIT Press, Cambridge.

[8] Garlan, D. (1995) Architectures for Software Systems. *Proceedings of the* 1*st International Workshop CMU*, **20**, No. 3.

[9] Monroe, R., Wile, D. and Garlan, A.D. (1997) CME: An Architecture Description Interchange Language. The Centre for Advanced Studies on Collaborative Research. IBM Press, Indianapolis.

[10] Reza, H. and Grant, E. (2005) Quality-Oriented Software Architecture. *International Conference on Information Technology*, Las Vegas, 4-6 April 2005, 140-145. https://doi.org/10.1109/itcc.2005.237

[11] Shaw, M. and Garland, D. (1996) Software Architecture. *Perspective on an Emerging Discipline*, Prentice Hall, Upper Saddle River.