# Common Architectural Styles Ranking Based on the Availability, Security and Performance Quality Attributes

Safieh Tahmasebipour, stahmasebi89@gmail.com

Seyed morteza Babamir *, babamir@kashanu.ac.ir

## Abstract

An architect designs software using architectural styles and quality attributes. To satisfy quality attributes architectural *tactics* are used. Every architectural tactic is a design strategy associated with an individual quality attribute. Indeed, architectural tactics determine how to implement quality attributes in architectural styles. Therefore, interaction between architectural tactics and architectural styles plays an important role in meeting quality attributes. In this paper, the interaction between architectural tactics of *security, availability and performance* and architectural styles of *pipes and filters, layered, blackboard, client-server and broker* is evaluated. A new approach is proposed to rank the architectural styles. Using this approach, the best architectural style for every individual quality attribute or each combination of quality attributes is obtained.

**Keywords**: Software architecture, Software style, Quality attribute, Style ranking, Architectural tactic

## 1- Introduction

Several definitions have been proposed for the concept of architectural style [1,2,3,4]. They only describe the concept of architectural style from different points of view. An architectural style indicates the structure and common properties of a family of software rather than of single software. A style provides a vocabulary for describing components and a set of connectors that can be used to compose the components into configurations [1]. An architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined [2]. Style could be supposed as a set of governing principles on architecture.

Non-functional requirements play a critical role during software development, serving as selection criteria for choosing among myriads of alternative design and ultimate implementation [5]. Non-functional requirements are properties and qualities the software must possess while providing its intended functional requirements or services [6]. In other words, quality attributes affects final operation of the software and to satisfy quality attributes, special structures should be used in software production process. In this research the quality attributes of security, availability and performance are investigated.

*Tactic* is a design decision that affects the realization of a quality attribute [7]. Indeed, implementation of an architectural tactics results in meeting a special quality attribute in an intended architectural style. As shown in Fig. 1, architectural tactics are intersection point of architectural styles and quality attributes. For instance, redundancy is used to promote availability. Any tactic may require essential changes in the structure and behavior of style or it may need a new structure and behavior. Tactics are divided into design time and run time categories. For example, "hide information" tactic is a design time tactic that improves modifiability. "Authenticate" tactic is a run time tactic that enhances software security. In this study, the run time tactics are analyzed.

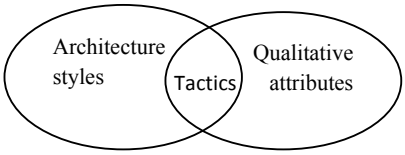


Fig.1. Relation between architectural styles, quality attributes and tactics

An architect uses tactics to an appropriate design using architectural styles. In other words, a set of tactics are implemented in any architectural style. For example, in order to satisfy the quality attribute of security in the pipes and filters style, "Maintain data confidentiality" tactic is used. In this tactic, a filter is placed in the output of data stream in order to encryption and another filter is placed in the input of data stream in order to decryption.

The *pipes and filters, layered, broker, blackboard and client-server* architectural styles and the *security, availability and performance* tactics are considered in this paper. The pipes and filters, layered, broker, blackboard and client-server architectural styles introduced in [2, 4, 8, 9, 10, 11, 12] and the availability, security and performance tactics introduced in [7]. In this research, the interaction between above mentioned tactics and common architectural styles is obtained. Then, the architectural styles are ranked using our proposed approach where, at first the priority of the quality attributes is determined by software manufacturer and then the best architectural style is suggested.

The next section discusses the related works and Section 3 describes the architectural tactics and Section 4 relates interaction between architectural tactics and architectural style. Sections 5, 6 and 7 determine the interaction between the architectural tactics of security, availability and performance and common architectural styles, respectively. Section 8 discusses common architectural styles ranking using our proposed approach and finally the concluding remarks is given in Section 9.

## 2- Related Works

In this paper, interaction between the tactics and common architectural styles to satisfying quality attributes is discussed. Tactics were introduced by the Bass et.al [7, 13] to meet quality attributes in architectural styles. For example, for the quality attribute of availability, availability tactics were introduced. Availability is concerned with software failure and its associated consequences. A failure occurs when the software no longer delivers a service that is consistent with its specification and this failure is observable by the software users. That recovery or repair is an important type of availability. The tactics will keep faults from becoming failures or at least bound the effects of the fault and make repair possible. Availability tactics are divided into three types: a) fault detection, b) fault recovery and c) fault prevention and for fault detection, three tactics have been introduced as follows:

- ping/echo: One component issues a ping and expects to receive back an echo, within a predefined time, from the component under scrutiny. If echo is not broadcast within a predefined time, the under study component would be considered as a failed component.
- Heartbeat: In this case one component emits a heartbeat message periodically and another component listens for it. If the heartbeat fails, the originating component is assumed to have failed
- Exception: One method for recognizing faults is to encounter an exception, which is raised when one of the fault.

But in [7], the interactions between architectural styles and tactics have not been studied. In 2010, a technique was presented to describe the interaction between architectural style and architectural tactics [14]. A few interactions have been introduced using six types of change. To investigate the interaction, concept of impact magnitude was defined for each change type. Magnitude of impact shows that how much a style should be changed to implement tactic in the style. The interaction between a group of architectural tactics and a set of architectural styles and also architectural styles ranking to satisfy quality attributes have not been presented by this reference.

In the present research, the interaction between *the pipes and filters, layered, broker, blackboard and client-server* architectural styles and the *security, availability and performance* tactics is obtained. Then, the common architectural styles are ranked based on the quality attributes using a new proposed approach. Accordingly the best architectural style for every individual quality attribute or each combination of quality attributes is suggested.

## 3- **Architectural Tactics**

To satisfy every quality attribute, several tactics have been presented. The tactics are divided into design time and run time categories. For example, "hide information" tactic is a design time tactic that improves modifiability and "Authenticate" tactic is a run time tactic that enhances software security. In this study, the run time tactics that have been introduced in [7] are analyzed. Security tactics (Fig. 2) are divided into three types: a) resisting attacks, b) detecting attacks and c) recovering from attacks. Fig. 3 shows the availability tactics that are divided into three types: a) fault detection, b) fault recovery and c) fault prevention. Performance tactics (Fig. 4) are divided into three types: a) resource demand, b) resource management and c) resource arbitration [7, 15].

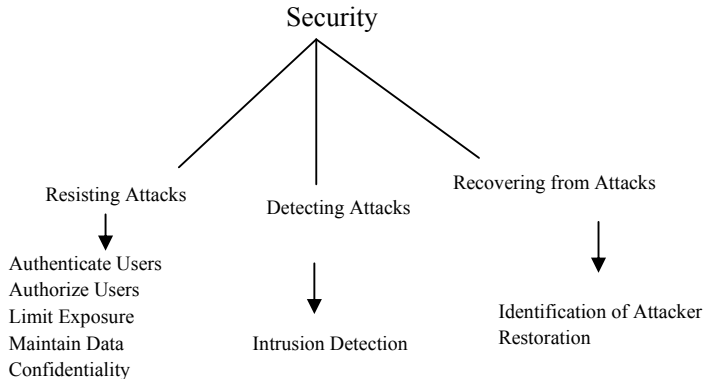


Fig.2. Architectural tactics of security [7]

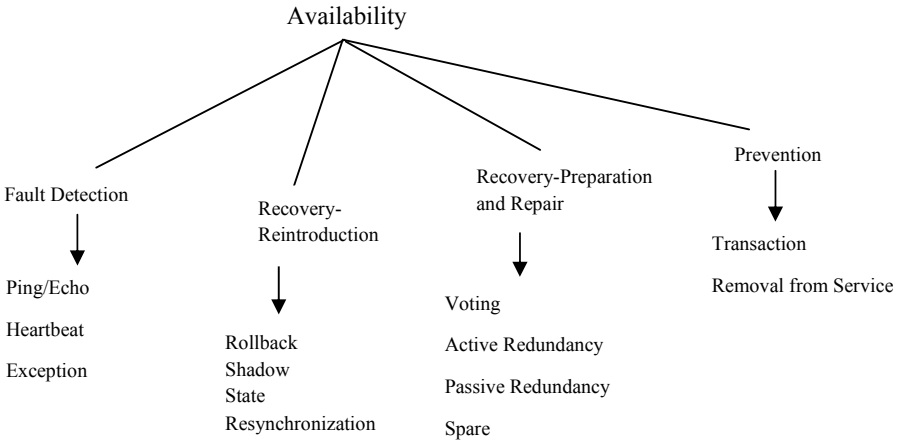


Fig.3.Architectural tactics of availability [7]

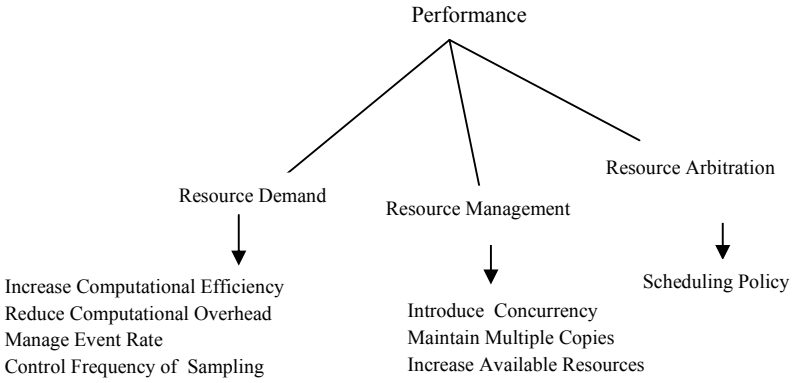


Fig.4. Architectural tactics of performance [7]

## 4- Interaction Between the Tactics and Software Architectural Styles

As mentioned in Section 1, architectural tactics are the intersection point of quality attributes and architectural styles. Indeed, tactics determine how to implement quality attributes in architectural styles. To investigate the interaction between tactics and styles, six types of change in structure and behavior of the architectural styles have been introduced [14]. While a tactic is implemented in an architectural style, it may require more than one change. The six types of changes are: "Implemented in", "Replicates", "Add in the Style", "Add out of the Style", "Modify" and "delete" which are defined as follows:

- *Implemented in*: The tactic is implemented within a component of the style. Actions are added within the sequence of the component.
- *Replicates*: A component is duplicated. The component's sequence of actions is copied intact, most likely to different hardware.
- *Add in the Style*: A new instance of a component is added to the architecture. The component follows the structure of the architecture style. The new component behavior has to follow the constraints of the style.
- *Add out of the Style*: A new component is added to the architecture which does not follow the structure of the style. The component added will have its own behavior. The actions do not have to follow the style.
- *Modify*: A component's structure changes. The change to the structure of a component implies changes or additions within the action sequence of the component that are more significant than those found in "Implemented in".
- *Delete*: A component is removed.

Concept of impact magnitude is introduced to evaluate the implementing tactics in architectural styles [14]. Magnitude of impact shows that how much a style should be changed to implement tactic in the style. Magnitude of impact is determined by the following five scales:

- *Good Fit*: The structure of the style is highly compatible with the structural needs of the tactic. This is shown by "+ +" notation.
- *Minor Changes*: The tactic can be implemented with few changes to the structure of the style. This is shown by "+" notation.
- *Neutral*: The style and the tactic are basically orthogonal. The tactic is implemented independently of the style, and receives neither help nor hindrance from it. This is shown by "~" notation.
- *Significant Changes*: The changes needed are more significant. This is shown by "-" notation.
- *Poor Fit*: Compared with previous case a tactic requires more changes in order to be implemented in the style's structure. This is shown by "- -" notation.

The impact magnitude of the style changes is a function of the number of required changes which is shown in Table 1 [14].

Table 1. Impact magnitude as a function of the number of required changes [14].

| Change Type | Change Number | Impact Range |
|---|---|---|
| Implemented in | 1 | + + to + |
| Replicate | 3 or less | + + to + |
| | More than 3 | + to ~ |
| Add in the Style | 3 or less | + + to + |
| | More than 3 | + to ~ |
| Add out of the Style | 3 or less | ~ to - |
| | More than 3 | - to -- |
| Modify | 3 or less | ++ to - |
| | More than 3 | ~ to -- |

## 5- Interaction Between the Security Tactics and Common Architectural Styles

Security tactics are used to meet security quality attributes in the architectural styles. Therefore, interaction between security tactics and architectural styles must be obtained. Table 2 shows our findings about the security tactic. In this table, the interaction between security tactics and architectural styles of pipes and filters, layered, blackboard, client-server and broker is given. As stated in previous section, in order to determine the interaction between tactics and architectural styles, six types of change have been used. To implement a tactic in an architectural style, it may require more than one change. Change types include Implemented in (*I*), Replicates (*R*), Add in the Style ( *AI* ), Add out of the Style (*AO*) , Modify (*M*) and impact magnitude (--,++,+,-,~) are shown in Table 2. Five interactions between security tactics and architectural styles are described as follows:

Table 2. Interaction between security tactics and common architectural styles

| | Resisting Attacks | | | | | | | | Detecting Attacks | | Recovering from Attacks | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Authenticate Users | | Authorize Users | | Limit Exposure | | Maintain Data Confidentiality | | Intrusion Detection | | Identification of Attacker | | Restoration | |
| | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact |
| Pipes &Filters | AI ,AO | ~ | AO,M | -- | I | ++ | AI | + | AO,M | -- | AO,M | -- | AO,M | -- |
| Layered | AI | ++ | M | -- | I | ++ | AI, M | ∼ | AO,M | -- | AO,M | -- | I | ++ |
| Broker | I | ++ | M | - | M | -- | AO,M | -- | M | + | AO,M | -- | I | ++ |
| Blackboard | M,AO | -- | I | ++ | AO | - | AO,M | -- | M,AO | -- | AO,M | -- | M | -- |
| Client-Server | AI | + | AI,M | + | M | -- | AI,M | - | AO | -- | AO,M | -- | I | ++ |

**Implementing and Impact of "Authenticate users" in the Broker Style**: Client is not directly connected to the server and the broker component acts as an intermediator between client and server. Thus, the broker component should be changed so that authenticate users tactic be placed in it. Since this change occurs inside the broker component and there was a connection between client and broker before, the broker component is naturally an appropriate place for authenticate. Consequently, the change of this tactic lies inside "Implemented in" category. In this case, one component is changed and its impact magnitude is two plus (i.e. "++") regarding to the Table 1.

**Implementing and Impact of "Authorize users" Tactic in the Blackboard Style:** In the blackboard style, there is a control component that the "Authorize users" can be lid in to it. Because the control component controls the blackboard routinely, the type of this change is "Implemented in". Since only the blackboard's controlling component has been changed its impact magnitude is two plus (i.e. "++") according to Table1.

**Implementing and Impact of "Maintain data confidentiality" Tactic in the Layered Style:** If a software operates within a secure environment and all layers are belong to a process, the output of software will be encrypted and sent by a layer. In this case, the type of change is "Add the in Style" because it follow style structure. Since a layer is added, regarding the Table 1, its impact is two plus (i.e. "++"). If software operates within an insecure environment and each layer is a separate process, they should be changed so that the output of each layer be encrypted and every layer should decrypted them prior to process the entering data (from bottom layer). These changes are the "Modify" type because layers should be changed for encryption and decryption purposes. Since all layers should be changed and the number of layers is usually more than 3, regarding the Table 1, its impact magnitude is two minus ( i.e. "--" ). Thus, the impact magnitude of this tactic depends on the implementation of layer and it could be positive or negative. Consequently final impact magnitude of this tactic is "~".

**Implementing and Impact of "intrusion detection" Tactic in the Client-Server Style**: Intrusion detection which should supervise the open ports of software is implemented as an independent process. A monitoring component is added to monitor the open ports of software. Because the added monitoring component (with connections to style components) does not follow style structure, the type of this change is "Add out of the Style". Since, the number of changes is more than 3, the impact magnitude of this tactic is two minus ( i.e. "--" ) according to Table1.

**Implementing and Impact of "Restoration" Tactic in the Pipes and Filters Style**: Software incompatibility state should be converted to its compatible state. Therefore, compatible state should be saved and incompatible state should be restored to compatible state. While the pipes and filters style has no state and it act on a data stream. Consequently, a component should be added to monitor the style operation in order to save software states. Therefore, the changes of this tactic are as follows: in the monitor component the type of change is "Add out of the Style" as the added component does not follow the style. In the filters, the type of change is "Modify" because filters should be modified to send their states to the monitoring component. Thus, we can conclude that the number of changes is more than 3 and regarding the Table 1, its impact magnitude is two minus ( i.e. "- -" ).

## 6- Interaction Between Availability Tactics and Common Architectural Styles

Availability tactics are used to meet availability quality attributes in the architectural styles. Therefore, interaction between availability tactics and architectural styles must be obtained. Tables 3 and 4 show our findings about the availability tactic. In these Tables, the interaction between availability tactics and architectural styles of *pipes and filters, layered, blackboard, client-server and broker* is given. As stated in Section 4, in order to describe the interaction between tactics and architectural styles, six types of changes in the structure and behavior of architectural styles have been introduced. In implementation of a tactic in an architectural style, it may require more than one change. Change types include Implemented in (*I*), Replicates (*R*), Add in the Style ( *AI* ), Add out of the Style (*AO*), Modify (*M*) and impact magnitude (--,++,+,-,~) are shown in Tables 3 and 4. Five interactions between availability tactics and architectural styles are described as follows:

Table 3. Interaction between availability tactics and common architectural styles

| | Fault Detection | | | | | | Recovery-Reintroduction | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Ping/Echo | | Heartbeat | | Exception | | Checkpoint/ Rollback | | Shadow | | State Resynchronization | |
| | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact |
| Pipes &Filters | AO, M | - - | AO, M | - - | AO, M | - - | AO, M | - - | AI | + | AO, M | - - |
| Layered | AI, M | + | AI, M | + | I | ++ | I | ++ | R, AI | + | M, AI | + |
| Broker | I | ++ | M | + | M | + | I | ++ | I | ++ | I | ++ |
| Blackboard | M | ~ | M | ~ | M | - - | M | - - | AO, R | - | M | + |
| Client-Server | AI, M | + | AI, M | + | M | + | I | ++ | AI, R | + | AI, M | + |

Table 4. Continued Table 3

| | Recovery-Preparation and Repair | | | | | | | | Prevention | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Voting | | Active Redundancy | | Passive Redundancy | | Spare | | Transaction | | Removal from Service | |
| | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact |
| Pipes &Filters | AI | + | R | + | R, M | - | AI | + | M | ~ | AO | - - |
| Layered | AI, M | + | R, AI | + | M, R | - | M, R | ~ | I | ++ | M, AI | ~ |
| Broker | I | ++ | I | ++ | M | + | I | ++ | I | ++ | I | ++ |
| Blackboard | AI, M | + | R ,M | + | M, R | ~ | M, R | ~ | M | -- | AO | - - |
| Client-Server | AI | + | AI, R | + | AI, R, M | + | AI, R | + | I | ++ | AO | ~ |

**Implementing and Impact "Ping/ Echo" Tactic in the Layered Style:** In this tactic an additional component is required to send ping massage. If all layers belong to one process, the top layer responds to the layer sending the ping message via echo message. If each layer is separate process, an upper layer sends the ping message to the adjacent lower layer and this layer pings its lower layers prior to echo response to upper layer. Ping-echo messages are exchanged hierarchically. Thus, this change lies inside the "Add in the Style" type because the changes follow style structure .On the other hand, layers should be changed to respond to the ping message via the echo message. Therefore, this change lies inside the "Modify" category. Regarding to the Table 1, since the number of changes is more than one (an added layer to send the ping message with the impact magnitude of two plus ( i.e. "++" ) as well as changes of layers to respond via the echo message with the impact magnitude of minus ( i.e. "-") ), final impact magnitude of this tactic is plus ( i.e. "+") .

**Implementing and Impact of "checkpoint and roll back" Tactic in the Blackboard Style**: It is very difficult to determine checkpoints to which one can roll back. Determining these checkpoints is contradictory to the blackboard idea. An alternative method can be to log all inputs in order. Nevertheless this method is very difficult and need large storage. Therefore, the blackboard component requires significant changes. In this case, the type of change is "Modify" and the number of changes is more than 3. Thus the impact magnitude is two minus (i.e. "--") regarding the Table 1.

**Implementing and Impact of Shadow Tactic in the Broker Style:** The broker component is a natural place for monitoring servers that return to service. The broker component can save the states of intact servers and mark the servers acting as shadow until they return to their normal operations. Since the broker component typically controls servers operations, the type of change is "Implemented in" and its impact magnitude is two plus (i.e. "++") regarding the Table 1.

**Implementing and Impact of Voting Tactic in the Pipes and Filters Style:** Different filters are considered as voter and a given filter is considered as voting. The input of filter should be distributed among the voters. Therefore, a pipe with one input and several outputs should be used. Since the structure of style is preserved in one hand and on the other hand some additional filters have been added, this change lies inside the "Add in the Style" type. As the number of added components is more than 3 (some voter filters and a voting filter), its impact magnitude is plus (i.e. "+") regarding the Table 1.

**Implementing and Impact of "Active Redundancy" Tactic in the Client-Server Style:** In this tactic servers are replicated. During sending a request to the server, a central component is required in order to distribute the request among redundant servers simultaneously. The result of the main server is sent to client and the results of the redundant servers are sent to the central component. If a fault is occurred, the result is sent to the client via the central component. The central component plays client role to the server. Since changes follow style structure, the type of the changes is "Add in the Style" due to the addition of the central component and it is the "Replicates" type due to the replication of servers. Since the number of changes is minimum 2, its impact magnitude is plus (i.e. "+") according to the Table 1.

## 7- Interaction Between Performance Tactics and Common Architectural Styles

Performance tactics are used to meet performance quality attributes in the architectural styles. Therefore, interaction between performance tactics and architectural styles must be obtained. Table 5 shows our findings about the performance tactic. In this table, the interaction between performance tactics and architectural styles of *pipes and filters, layered, blackboard, client-server and broker* is given. As stated in Section 4, in order to describe the interaction between tactics and architectural styles, six types of changes in the structure and behavior of architectural styles have been introduced. In implementation of a tactic in an architectural style, it may require more than one change. Change types include Implemented in (*I*), Replicates (*R*), Add in the Style ( *AI* ), Add out of the Style (*AO*) , Modify (*M*) and impact magnitude (--,++,+,-,~) are shown in Tables 5. Five interactions between performance tactics and architectural styles are described as follows:

Table 5. Interaction between performance tactics and common architectural styles

| | Resource Demand | | | | | | | | Resource Management | | | | | | Resource Arbitration | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Increase Computational Efficiency | | Reduce Computational Overhead | | Manage Event Rate | | Control Frequency of Sampling | | Introduce Concurrency | | Maintain Multiple Copies | | Increase Available Resources | | Scheduling Policy | |
| | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact | Change Type | Impact |
| Pipes &Filters | M | - | M,AO | - - | M,AO | - - | AI | + | I | ++ | AO,R | - - | M | ~ | M | - - |
| layered | M | - | AO | - | M,AI | ~ | AI | + | R, AI | + | M | - | M | ~ | M | -- |
| Broker | M | + | M,AO | - - | M ,AO | - | M | + | M ،R | ~ | AO,R | ~ | M | ~ | M | + |
| Blackboard | M | + | AO,M | - - | AO | - - | M | + | I | ++ | AO,M, R | - - | M | ~ | M | + |
| Client-Server | M | + | M, AO | - - | M, AO | - | AI | + | R | + | R, M | + | M | ~ | AO ,M | - - |

**Implementing and Impact of "Increase computational efficiency" Tactic in the Layered Style:** In the layered style, each layer performs separate task compared with other layers. Therefore, embedded algorithms in layers should be optimized to increase computational efficiency. So, all layers should be changed. Thus, the type of this change is "Modify". Since some layers are changed, its impact magnitude is minus (i.e. "-"), regarding the Table 1.

**Implementing and Impact of "reduce computational overhead" Tactic in the Blackboard Style:** To implement this tactic, a monitoring component is required. The monitoring component monitors knowledge sources. If a knowledge sources has no processing request, it would be deleted from service. Therefore processing needs are reduced as well as decrease in computational overhead. Because the monitoring component does not follow style structure the type of change is "Add out of the Style". The monitoring component with connections to knowledge sources has been added and knowledge sources should be changed that the type of this change is "Modify". Thus the number of changes is more than 3 and its impact magnitude is two minus (i.e. "--") regarding the Table 1.

**Implementing and Impact of "Introduce Concurrency" Tactic in the Pipes and Filters Style**: Filters should work on requests in parallel. Since filters normally work on requests in parallel the type of change is “Implemented in “ and regarding the Table 1, its impact magnitude is two plus (i.e. " ++").

**Implementing and Impact of "Maintain multiple copies" Tactic in the Client-Server Style:** Server is copied and the copied server is placed in client side so that it becomes a local server for the client. The updates of main server are sent to the copied server and the main server is changed that the type of this change is "Modify". Indeed, during sending updates, the main server plays role of server and the copied server plays client role. Therefore, the style structure is preserved. The type of change is "Replicates" because the server has been copied. Since some copies has been added to the style, impact magnitude is plus (i.e." +") regarding the Table 1.

**Implementing and Impact of "Scheduling Policy" Tactic in the Broker Style:** In this tactic, requests enter the broker component at first. Therefore the component should be changed to enable itself to apply scheduling policy on the requests. On the other hand, servers should be changed to accept the scheduling policy. Since both broker component and server are changed, the type of changes is "Modify". Consequently the number of changes is less than 3, and the impact magnitude is plus (i.e. "+") regarding the Table 1.

## 8- Common Architectural Styles Ranking to Satisfy the Quality Attributes

We calculate the average of impact magnitude for each style using Tables 2-5, by assigning a number to each scales of impact magnitude (introduced in Section 4). In these tables, notations "++", "+", "--" and "-" are replaced by numbers: +2, +1, -2 and -1 respectively and also "~" notation is replaced by zero. Average of each row of the mentioned tables is given in Table 6.

In an architectural style, if the impact magnitude of a tactic is more positive, style structure has a better consistency to implement the tactic. According to the Table 6, the consistency of common architectural styles to implement security (a), availability (b) and performance (c) tactics are as follows, respectively:

a) Layered, Broker, Client –server, Pipes & filters and Blackboard

b) Broker, Client –server, Layered, Blackboard and Pipes & filters

c) Broker, Client –server, Blackboard, Layered and Pipes & filters

Table 6. Impact magnitude average of tactics in common architectural styles.

| | Security | Availability | Performance |
|---|---|---|---|
| Pipes &Filters | -0.714 | -0.75 | -0.75 |
| Layered | 0 | +0.917 | -0.375 |
| Broker | -0.286 | +1.75 | 0 |
| Blackboard | -1.286 | -0.5 | -0.125 |
| Client-Server | -0.428 | +1.083 | -0.125 |

We obtained interaction between tactics of the security, availability and performance with common architectural styles in Sections 5-7. Then we calculated the impact average of all mentioned tactics that the impact average indicates style consistency to implement the tactics. The tactics determine how to implement quality attributes in architectural styles. To rank architectural style based on the intended quality attributes, at first, Table 6 is defined as following matrix:

$$ST = \begin{array}{c} \begin{matrix} \text{Security} & \text{Availability} & \text{Performance} \end{matrix} \\ \begin{bmatrix} -0.714 & -0.75 & -0.75 \\ 0 & +0.917 & -0.375 \\ -0.286 & +1.75 & 0 \\ -1.286 & -0.5 & -0.125 \\ -0.428 & +1.083 & -0.125 \end{bmatrix} \end{array} \begin{matrix} \text{Pipes \& Filters} \\ \text{Layer} \\ \text{Broker} \\ \text{Blackboard} \\ \text{Client-Server} \end{matrix}$$

In the next step, we define priority of the quality attributes of security, availability and performance as a matrix $q_{3\times1}$ based on the software developers demand. A weight should be assigned to each quality attributes in range of 0% to 100% by the software developers. Then the weight is defined as a number between 0 to 1. The first to third entries in the matrix q indicates security, availability and performance quality attributes, respectively.

In the next step, we suggest equation (1) to rank the architectural styles based on the desired quality attributes.

$$R = ST \times q \tag{1}$$

The value of each entry of matrix R indicates the compatibility of an architectural style. Indeed, each entry of the matrix R determines the rank of an architectural style to satisfy quality attributes. The more positive the entry, the higher the rank.

If sole quality attribute that is important for a software developer is security, the weight of security item is considered 1. Thus, matrix $q_1$ is written as follows: $\begin{bmatrix}1\\0\\0\end{bmatrix}$. Using the equation (1) and replacing q by $q_1$, the rank of styles is obtained. Equation (2) shows the final result. According to equation (2), the most positive number has been assigned to layered style. Consequently, to design this software the layered style is the best choice.

$$R_1=\begin{bmatrix}-0.714\\0\\-0.286\\-1.286\\-0.428\end{bmatrix} \tag{2}$$

If the security (with weight 60%) and availability (with weight 40%) are important for a software developers, matrix $q_2$ is written as follows: $\begin{bmatrix}0.6\\0.4\\0\end{bmatrix}$. The rank of styles is obtained using the equation (1) and replacing q by $q_2$. According to the final result, equation (3), the broker style is the most positive. Therefore, broker style is suggested to architect and software developer.

$$R_2=\begin{bmatrix}-0.728\\+0.367\\+0.528\\-0.972\\+0.176\end{bmatrix} \tag{3}$$

## 9- Conclusion

Architectural tactics are the intersection point of quality attributes and architectural styles. Indeed, architectural tactics determine how to implement quality attributes in architectural styles. In this paper, all interactions between architectural tactics of the security, availability and performance with common architectural styles were obtained. According to average of interactions, the compatibility of common architectural styles to implement of security (a), availability (b) and performance (c) tactics are as follows, respectively:

a) Layered, Broker, Client –server, Pipes & filters and Blackboard

b) Broker, Client –server, Layered, Blackboard and Pipes & filters

c) Broker, Client –server, Blackboard, Layered and Pipes & filters

A new approach was offered to rank common architectural styles based on intended quality attributes too. Using this approach, software developers can determine the best architectural style for every individual quality attribute or each combination of quality attributes. In this method the following steps were performed:

- The interactions between mentioned tactics and common architectural styles were obtained
- The average of tactics impact were calculated and defined as a matrix.
- Priority of the quality attributes of security, availability and performance was defined as a matrix based on the software developers demand.
- Eventually, the best architectural style was obtained via definition of a new equation.

References**:**


1. Allen, R.," HLA: A Standards Effort as Architectural Style", Proc. of the 2nd Int'l Software Architecture Workshop (ISAW-2), pp. 130-133, 1996
2. Garlan, D., Shaw, M., "An Introduction to Software Architecture", Technical Report, CMU, Pittsburgh, PA, USA, 1994.
3. Perry, D. E., and Wolf, A. L.," Foundations for the Study of Software Architecture", ACM SIGSOFT Software Engineering Notes, Vol. 17, No. 4, pp. 40-52, 1992
4. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.,"pattern-Oriented Software Architecture: A Software of patterns", John Wiley & Sons, 1996
5. Chung, L., Nixon, B.A., Yu, E., and Mylopoulos, J., "Nonfunctional Requirements in Software Engineering", Kluwer Academic Publishing, 2000.
6. Saleh, K. , Al-Zarouni, A.,"Capturing Non-Functional Software Requirements Using the User Requirements Notation" ,The 2004 International Research Conference on Innovations in Information Technology,2004.
7. Bass, L., Clements, P., Kazman, R., "Software architecture in practice", Addison-Wesley, 2003
8. Avgeriou, P. and Zdun, U., "Architectural patterns Revisited– a pattern Language", In Proceedings of 10th European Conference on Style Languages of Programs (EuroPLoP 2005), pp. 1- 39, 2005.
9. Issarny, V., Benatre, J.-P., "Architecture-Based Exception Handling", In Proceedings of the 34th Annual Hawaii international Conference on Software Sciences (Hicss-34), 2001.
10. Shaw, M., Garlan ,D., "Software Architecture: perspectives on an Emerging Discipline", Prentice Hall, 1996.
11. Völter, M., Kircher ,M., Zdun ,U., "Remoting Styles: Foundations of Enterprise, Internet and Realtime Distributed Object Middleware", Wiley, 2005.
12. Medvidovic, N., Taylor, R.N., "Exploiting Architectural Styles to Create a Family of Applications", IEE Proceedings Software Engineering ,Vol.144, No.5, pp. 237-248, 1997
13. Bachmann, F., Bass, L., Nord, R., "Modifiability Tactics", Technical Report, SEI,pp.1-4, 2007.
14. Harrison, N. B., Avgeriou, P., "How Do Architecture Styles and Tactics Interact? A Model and Annotation", Journal of Softwares and Software, vol. 83, Issue 10, pp. 1735-1758, 2010.
15. Bachmann, F.,Bass,L.,Klein, M.,"Illuminating the Fundamental Contributors to Software Architecture Quality" , Technical Report, SEI,pp.26-30, 2002.