A COMPARATIVE ANALYSIS ON SOFTWARE ARCHITECTURE STYLES

Feidu Akmel [1], Ermiyas Birhanu [2], Behar Siraj [3], Seifedin Shifa[4]

Lecturer, Department of Computer Science, Department of Software Engineering,
Wolkite University, Wolkite, Ethiopia
Assistant Lecturer, Department of Information system,
Wolkite University, Wolkite, Ethiopia

ABSTRACT

Software architecture is the structural solution that achieves the overall technical and operational requirements for software developments. Software engineers applied software architectures for their software system developments; however, they worry the basic benchmarks in order to select software architecture styles, possible components, integration methods (connectors) and the exact application of each style.

The objective of this research work was a comparative analysis of software architecture styles by its weakness and benefits in order to select by the programmer during their design time. Finally, in this study, the researcher has been identified architectural styles, weakness, and Strength and application areas with its component, connector and Interface for the selected architectural styles.

KEYWORDS

Architecture Styles, Components, Connectors, Interface

1. INTRODUCTION

Today the business world is very dynamic and the organization often change their business processes to be competent in the market. These businesses supported their through software systems. Moreover, size and complexity of software systems increases, Software Architecture is emerging as an important research area in software engineering [1]. Software architecture is the process of defining a structured solution that meets all of the technical and operational requirements, while optimizing common quality attributes such as reusability performance, security, and manageability [2][3]. Software architects use a number of commonly recognized styles to develop the architecture of a System [3, 4]. Software architecture styles are an abstract framework developed for a family of systems to have general solutions to common problems that arise in the software development process. It is accountable to offer a lexicon of connectors and components with principles on how they can be combined, improve partitioning and allow the reuse of design by giving solutions to frequently occurring problems and describe a particular way to configure a collection of components which has a module with well-defined interfaces and re-usable connectors of communication link between modules [5].

This research presents to evaluate five architectural styles in order to minimize the confusion of the designers and programmer while selecting appropriate architecture style for their specific need. The researchers selected thus five commonly used styles that are applied for various types of software application areas in real world scenarios as we shown Table 1 .These software architecture styles are: Message Bus Architecture Style, Component-Based Architecture Style, Layered Architecture Style, Object-Oriented Architecture Style and Service Oriented Architecture Style (SOA)

2. STATEMENT OF PROBLEMS

According to, Sommerville definition regarding to Software Engineering [6] to develop, Manage and evolving software systems Software Engineering gives an emphasize on methods, theories and tools. Software Engineering has different models and each model has its own lifecycle mainly which includes requirement analysis, Design (System and Object), Implementation, Testing, Delivery and maintenance. During design phase, we have considered an issue regarding to software architecture, which are high-level design stage.

The main idea behind software architecture is to decompose the system in to a group of different components and then develop components and related connectors ,to achieve this decomposition and selection of an architecture styles [7]. An architecture style is defined by the component sets and the interactive rules between them. Most common known example of architectural style types are Component**-**based**,** Layered, Client -Server**,** Message Bus and Service Oriented Architecture as we shown Table 1.

Anubha et al [8] attempts to discussed the software architecture styles, importance and classifying it according to its benefits and its application, but they did not consider an emphasize about their interaction methods and its components.

Most of software engineers worry what are the basic benchmarks in order to select software architecture styles, possible components, integration methods (connectors) and the exact application of each style. In this study, we consider the selected architectural style strength and weakness using its parameters in addition to detail explanations of components, connectors and application areas.

Thus, the research is attempted to answer the following research questions:

- Which type of connectors and components are applied by software architecture styles?
- How data is communicated through software architecture styles?
- What are pros and cons we considered for each architectural style?

2.1.General objective

The main objective of this study was comparative analysis of software architecture styles in order to identify the application areas, advantage, disadvantages and how component of software system could be composed.

3. RELATIONSHIP BETWEEN ARCHITECTURE STYLES AND DESIGN PATTERNS

The most common known definition of design pattern was by Christopher Alexander there is a problem which occurs again and again in a given system and solved by available patterns many times[9], from this definition we asses four basic elements in design patterns which are patterns name, solutions, problems and consequences. The concept of design patterns is not about design of linked list and hash tables, but it also gives an emphasize to complex and domain specific applications (namely classes and objects). Architectural styles indicate that an established of design rules that classify the type of connectors and components that is used to applies to compose a system or subsystem, together with global or local constraints that are executed[10].
The relationship between styles and patterns are discussed in two perspectives, first, architectural styles can be viewed as kinds of patterns or perhaps more accurately as pattern languages. The second method that design patterns are associated to styles is that for a specified style there may exist a set of idiomatic uses of it. According to, Robert T et al view about styles and patterns are complementary mechanisms for encapsulating designs. Architectural styles are a collection of building blocks design elements, rules and design whereas patterns is expressing by making with solutions to lower level programming ways, rather than software system structuring issue[11].

3.1 Software Architecture Styles

Software architecture styles are patterns or an abstract framework developed for a family of systems to have overall solutions to problems that arise in the software development life cycle processes. There are set of principles and guidance's to shape or define the components and connectors that compose a solution and their relations. It 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. The most crucial benefit is that, they can provide a common language by being an opportunity for conversations that are technology agnostic because this helps a higher level of conversations that is inclusive of principles and patterns without becoming into specifics. Software architectural style is responsible to provide a connectors and components with rules on how they can be combined, improve partitioning and allow the reuse of design by giving solutions to frequently occurring problems and describe a particular way to configure a collection of components which has a module with well-defined interfaces, reusable and replaceable and connectors of communication link between modules. Each architecture style defines a system family that includes a set of component types that perform a required function by the system, semantic constraints which define how components can be integrated to form the system and a topological layout of the components indicating their run time interrelationships. There are different kind of architectural styles in software development world and varieties and categories of styles depend on their focus area, as we showed Table 1 below.

Table 1: Patterns (Category) and Architectural Styles [12]

Category	Architecture styles
Communication	Service-Oriented Architecture (SOA), Message Bus
Deployment	Client/Server, N-Tier, 3-Tier
Domain	Domain Driven Design
Structure	Component-Based, Object-Oriented, Layered Architecture

3.2.Client-Server Architecture Styles

Nowadays most communication like internet through client server architectural style, which means there is request and response take place. There are mainly three components such as Client, Server and Medium of communication[8]. Medium of communication between client and server are File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) and Hypertext Transfer Protocol (HTTP)[13].Client Server system has popular models namely 3-tier and n-tire architectures. From figure 1, we observed that the general layout for client and server architecture look like.

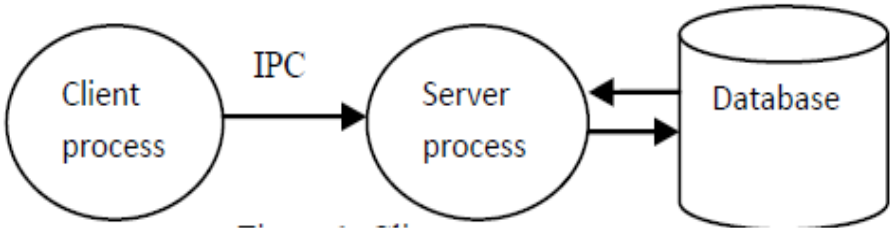

Figure1: Client Server Architecture frame work [13]

Application Areas

- **File Transfer:** client server architecture allows the user to store and retrieve their data to server such as movies, images and music's.
- **Mail Transfer:** Which provides the user to communicate with someone who is available somewhere through their mail using mail transfer protocol (MTP)?
- **Web based applications**[8]**:** Client Server architecture is used for internet applications , for instance AceProject, Ganttic, Celoxis etc.

Strength:

The basic benefits for using client server architecture are **higher security**: Since all the data are stored at server machine we can apply different security mechanisms to enhance its security than client machines. **Centralized data access**: any authorized user can able to access and updates data on the server because the data is available centrally. **Simple to maintenance**: Roles of a computing system are distributed among several servers that are recognized to each other through a medium or network. This ensure that a work stations keep unaware and unaffected by a server relocation, upgrade or repair [4].

Weakness /Challenges

There are a number of thing that are considered by any organization before they are going to deploy and use client server architectures[13, 14], which includes : The number of customers server by the model, It needs skilled peoples ,The server is quite expensive, Security issues , Management of applications.

3.3 Component-Based Architecture Style

Component based architecture is an architecture that focuses on decomposing software design into functional or logical components with their own methods, events and properties. In case of component based architectural styles no need of an issues like communication protocols and shared states[12]. The components are loosely coupled and reusable to provide modular programs that can be tailored to fit any need. A provided interface specifies what a component can provide to other components in the system. There are three types of components such as **user interface** components such as and buttons and grids and **helper and utility** components that expose a specific subset of functions used in other components. Other kinds of components are not frequently accessed, resource intensive and must be activated using the just-in-time (JIT) approach (common in remote or distributed component scenarios); and queued components whose method calls may be executed asynchronously using message queuing and store and forward.

As we have shown on Figure 2 there is five components such as User interface, Notification, Order Management, Accounting and persistence layer.

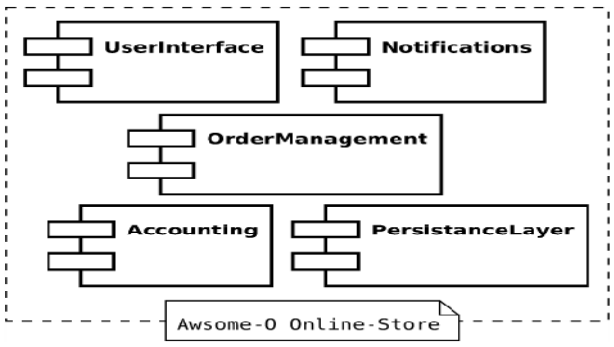

Figure 2 : Component based Architecture [8]

Strength:

The benefit of component based architecture, the use of components that are Ease of deployment, Reduced Cost, Ease of development, Reusable and Mitigation of Technical complexity[12] as researcher discussed here:- **Ease of deployment:** when new well-matched versions become available, we can replace the current versions without any drawback on the other components or the products as a whole. **Reduced Cost**: the use of third party components allows you to reduce the cost of development and technical support. **Ease of development**: components implement well-known interfaces to provide defined functionality, allowing development without impacting other parts of the system. **Reusable**: the use of reusable components are to extend applications or systems. **Mitigation of technical complexity**: components mitigate complexity through the use of a component container and its services. Example component services include component activation, life time management, method queuing and transactions.

Weakness:

The main weakness of component based architectural style discussed here: -**Message handling**: needs to be predefined for the components. Possibly it could be a limiting factor. **Reliance on third parties**: if your component come from a third party of some sort then you will be at the mercy for updates and changes to the component. **Complexity**: while it is designed to reduce complexity of systems it introduces a different type of complexity in terms of component-to-component interactions. **Testing**: can be difficult if the component doesn't come with its own execution environment. **Second system syndrome**: depending on how complex the components themselves are you can end up with a platform within a platform within a platform type problem.

Application Areas

The application area under component based architecture is structured; to create pluggable applications this architecture will be appropriate. Since whole picture of system is in terms of components, it results in high maintainability and portability[8].

3.4 Layered Architecture Style

The Layered Architecture style is focused around dividing software functionality into distinct layers that are interacted and stacked vertically on top of each other. Functionality within each layer is related by a common role or responsibility. Communication between layers is explicit and loosely coupled. The main application of layering helps to support great separation of concerns that in turn, support maintainability and flexibility. [12]. It is dependent on message passing between layers and clearly defined functional layers. Components for layered architecture are classified based on its layers namely presentations, business and data access.

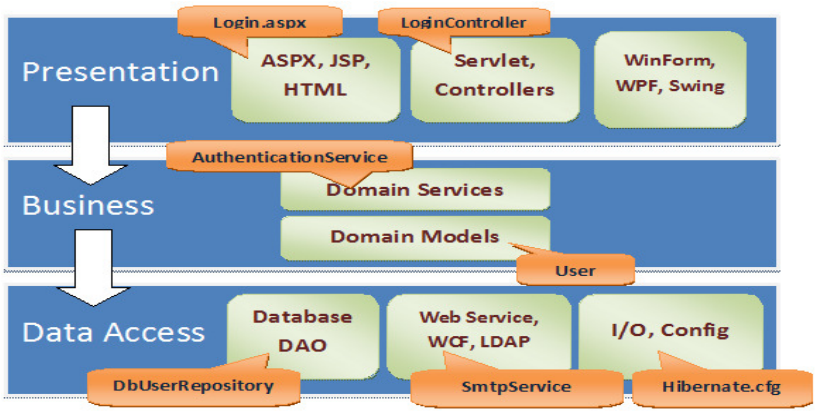

Figure3: Layered Architecture Style

Strength:

The main benefit of layered Architecture is: -**Abstraction**: layered architectures abstracts the view of the system as whole while providing enough detail to recognize the responsibilities and roles of individual layers and association between them. **Encapsulation**: no assumption need to be made about data types, methods and properties or implementation during design as these are not exposed at layer boundaries. **Clearly defined functional layer**: the separation between functionality in each layer is clear. **High cohesion**: well defined responsibility boundaries for each layer and ensuring that each layer contains functionality directly related to the task of that layer will help to maximize cohesion within the layer. **Reusable**: lowers layers have no dependencies on higher layers potentially allowing them to be reusable in other scenarios. **Loose coupling:** communication between layers is based on abstraction and events to provide loose.

Weakness:

Difficulties structuring systems as layered systems. Specifically, restricting communication to adjacent layers and keeping coupling between layers reduced. It required to consider a various presentation patterns if the user need to improved testability and simplified maintenance of user interface functionality or to separate task of designing the UI from the development of the logic code that drives it.

Application Areas

According to researchers the current application area within industry are OSI model (Networking), telecommunications domain, comprising architecture documents in Nokia (mobile phones and Internet services) and Nokia Siemens Networks (telecommunications infrastructure) [15], generally it applied for both networking and Mobile industries.

3.5 Message Bus Architecture Style

Message bus architecture gives detail about the protocols of using a software system that can accept and send messages using one or more communication channels, so that applications can interact without needing to know specific details about each other[4]. This style uses common buss for the interaction between applications which is accomplished by messages, mainly asynchronously. There are various implementation methods, the most common implementations of message bus architecture use either a Publish/Subscribe pattern or a messaging router, message Queuing. Message bus and service oriented styles are under categories of communication, but Service oriented architecture gives itself very well to utilizing a message bus architecture to facilitate the subscription of service providers to service requesters[16].

Strength:

The main strength for message bus Architecture are flexibility ,Extensibility, Scalability [16].**Extensibility:** Applications with message bus architecture support both adding and removing from the bus without having an impact on the existing applications. **Flexibility:** The set of applications that make up a complex process, or the communication patterns between applications, can be changed easily to match changes in business or user requirements, simply through changes to the configuration or parameters that control routing and application complexity is reduced because each application only needs to know how to communicate with the bus. **Scalability**: numerous instances of the identical application can be attached to the bus in order to touch multiple requirements at the same time.

Weakness:

The drawback of message bus architecture is difficult to modifiability, lower security and problem of down tolerances[16].

Lower modifiability: changes to the bus interface must maintain backwards compatibility, or every component that utilizes the bus will have to be updated in order to support the new interface .**Lowered security**: a broadcast based message bus offers no privacy without some form of encryption as messages are delivered indiscriminately to all connected nodes. **Lowered down tolerance**: The bus becomes a single point of failure for all communications across the application. Logic is not necessarily implemented for applications to manage their own messages while the bus is out of service, which can lead to message loss and failures of data integrity.

Application Areas

A message bus offers the capability to handle m**essage-oriented communications, complex processing logic, modifications to processing logic, integration with different environments** systems.

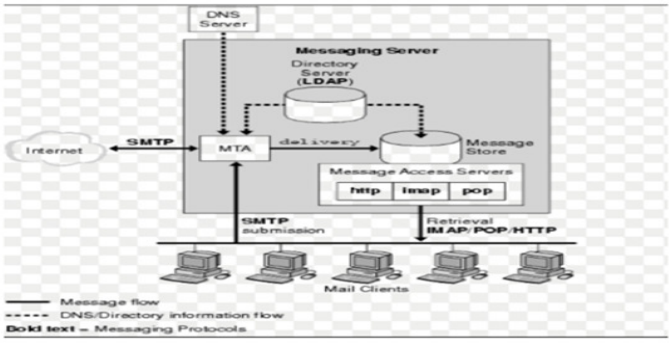

Figure 4: Message bus Architecture

3.6 Object Oriented Architecture Style

The basic feature of object-oriented architecture abstract objects that denotes data and the interaction method is whether functional or procedural [17]. One of the design principle which divided object based on its responsibilities is called object oriented architecture, for any products or Applications into individual reusable and self-sufficient objects, each containing the data and the behavior relevant to the object. Object- oriented design look like an application as sequence of cooperating objects, instead of a set of routines or procedural instructions[4]. Due to its concepts and implementations contribute to the getting acceptance of the architecture to expansion popularity in the world. These major principles are **Inheritance**: where objects can inherit the characteristics of other objects. **Encapsulation**: the internal of an object can be hidden from others so that only that object can manipulate its own state and variables. **Abstraction**: breaking down a system into logical components that can perform some sort of work and communicate with other objects in meaningful ways. **Polymorphism**: giving an object multiple forms. The component for OOP are objects and the communication between this object can be achieved through sending message or passing parameters from and to objects. The relationship between the object is depends on the application being developed by the programmer, as shown figure 5.

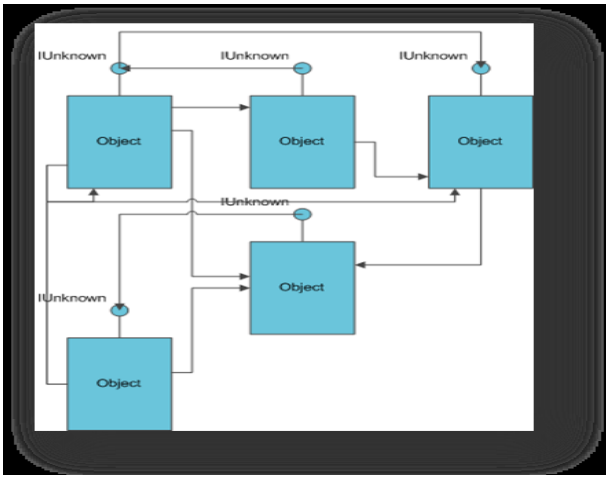

Figure 5: Object Oriented Architecture [17]

Strength

Reusability, Testability, Extensibility and highly cohesive are the strength of OOP styles, as they have the principle of polymorphism, abstraction, encapsulation and polymorphism. Thus, Object interacts with others, if the method which is available within one object changes there is also affects the other.

Weakness

The main drawback of object-oriented architecture are service integration and strong coupling between objects, there is an integration of services into object oriented systems has been made however, as current business's market trends has moved towards a greater use of external services in line with the advancement of telecommunications capabilities expanding more, it is becoming a common problem in existing object-oriented systems. Strong coupling between super classes and sub classes, swapping out of super classes can break sub classes.

Application Areas

The application area of OOP is for complex business and application domains such as telecommunications, distributed medical imaging, and real-time applications

3.7 Service Oriented Architecture Style

Service-oriented architecture (SOA) makes application functionality to be provided as a set of services, and the creation of applications that make use of software services. It is an architectural style that supports service orientation. It is a way of thinking in terms of services, service-based development, and the outcomes of services. In SOA, the basic components are types are service provider, service users and Service registry. An approach that communicates between each component in SOA can be implemented differently. During pure web service solution, SOA protocol is used. However, software Architect wants the simpler approach such as Representational State Transfer (REST) as a means of communication between SOA components. There are other alternatives in order to communicate through message passing protocols like Microsoft MSMQ and IBM WebSphere MQ (previously called MQSeries) [18].

Strength

Domain alignment: Reuse of common services with standard interfaces increases business and technology opportunities and reduces cost .**Abstraction**: Services are self-directed and accessed through a formal contract, which gives loose coupling and abstraction and services can representation descriptions that allow other applications and services to locate them and automatically decide the interface. **Interoperability**: Because the rules and data formats are based on industry standards, the provider and consumer of the service can be built and deployed on different platforms. **Rationalization**: Services can be coarse in order to provide specific functionality, rather than duplicating the functionality in number of applications, which removes duplication.

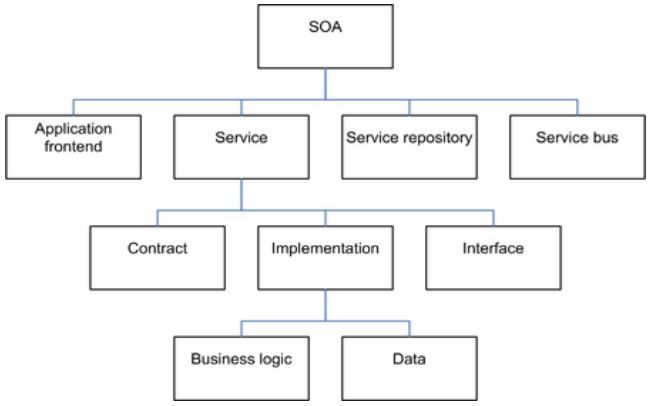

Figure 6: Service Oriented Architecture[18]

Application Area of SOA widely used for constructing large distributed systems.

4. CONCLUSION

Software architecture styles are patterns or an abstract framework developed for a family of systems to have general solutions to common problems that arise in the software development process.

In this study, the researcher's has presented a comparative analysis of various software architecture styles using the parameters like implementation architecture ,basic advantages and drawback of the architectural styles ,interface, component ,integration methods (connectors), and application areas of five architectural styles namely Component-Based Architecture Style, Layered Architecture Style, Service Oriented Architecture Style, Object-Oriented Architecture Style and Message Bus Architecture Style. According to the researchers knowledge recommended that, software developer should consider these styles in order to select the best architectures styles that fit with their problem domains based the above parameters. Moreover, the researchers recommended that for the future we need to give attention to design hybrid of architecture styles, components and connectors in order to communicate each other to design generic architecture styles that meets various problem domains.

REFERENCES

[1] Guozhen Tan, Xinpeng Li, Jiankun Wu, H. Z. and, and C. Li.

[2] Microsoft. (2013). What is Software Architecture Available: https://msdn.microsoft.com/en-us/library/ee658098.aspx

[3] Mei Hong, Chang Jichuan, and Y. Fuqing, "Software component composition based on ADL and Middelware," Science in China vol. Series F pp. 136-151.

[4] Microsoft. (2013). Chapter 3: Architectural Patterns and styles Available: https://msdn.microsoft.com/en-us/library/ee658117.aspx

[5] D.Garlan and M. Shaw. (1994). An introduction to software Architecture Available: http://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/intro_softarch.pdf

[6] I. Sommerville, Software Engineering. Addison-Wesley Pub. Co, 1996.

[7] C. Hai-Shan, "Survey on the Style and Description of Software Architecture," The 8th International Conference on Computer Supported Cooperative Work in Design Proceedings, 2003.

[8] Anubha S, Manoj K, and S. A, "A Complete Survey on Software Architectural Styles and Patterns," 4th International Conference on Eco-friendly Computing and Communication Systems, 2015.

[9] KevinZhang, Design Patterns Elements of Reusable Object-Oriented Software.

[10] Somo S, "Software Architecture: Styles and Representational Schemes " MSc Interdepartmental Program in System Sciences National Institute of Technology, Rourkela, India,, 2004

[11] Robert T, Andrew K, Ralph M, and D. B., "Architectural Styles, Design Patterns, and " IEEE Software.

[12] P . U. Chavan, Dr. M. Murugan, and P. P. Chavan, "A Review on Software Architecture Styles with Layered Robotic Software Architecture," presented at the International Conference on Computing Communication Control and Automation, 2015.

[13] Haroon S, "Client-Server Model " IOSR Journal of Computer Engineering (IOSR-JCE), vol. 16, 2014,.

[14] S. Tayib, "Title," unpublishedl.

[15] Juha S and Varvana M, "Layered Architecture Revisited – Comparison of Research and Practice " IEEE 2009.

[16] D. Tody. (1998). Message Bus and Distributed Object Technology Available: http://adass.org/adass/proceedings/adass97/todyd2.html

[17] M. Boddoohi, "An Evaluation of Software Architectures – Using Aspects " MSc MSc, Information Systems and Operations Management, University of North Carolina Wilmington, 2010.

[18] Phil B, Rick K, and Paulo M, "Evaluating a Service-Oriented Architecture," Carnegie MellonUniversity,USA200