# Assignment For SOF204 #1

Name: Liu Aofan - Student ID: SWE2009510

Course: Software Architecture and Design Patterns- Profession: Dr. Tee Sim Hui

**Table of Contents**

[Assignment For SOF204 #1 1](#_Toc88568587)

[1. Introduction 2](#_Toc88568588)

[2. Bank System Analysis 2](#_Toc88568589)

[2.1. Purpose 3](#_Toc88568590)

[2.2. Requirement Analysis 4](#_Toc88568591)

[3. Reasons for Software Architecture 5](#_Toc88568592)

[2.3. Inhibiting or enabling a system’s quality attributes 6](#_Toc88568593)

[2.4. Architecture facilitates communication among stakeholders 8](#_Toc88568594)

[2.5. An architecture may influence the organizational structure 8](#_Toc88568595)

[2.6. Architectures improve cost and schedule estimates 12](#_Toc88568596)

[4. Conclusion 13](#_Toc88568597)

[References 15](#_Toc88568598)

[Marking Rubrics 16](#_Toc88568599)

## Introduction

Software architecture, or organizations of a system, which refers to the **fundamental structures** of a software system as well as the **discipline** of creating such structures, plays an important role in the maintenance of software systems (Cuesta, 2010).

It helps to **expose** the structure of a system **hiding behind** what we can see from the surface. Instead of focusing on the source code, it mainly pays attention to how the components and elements **interact with** each other.

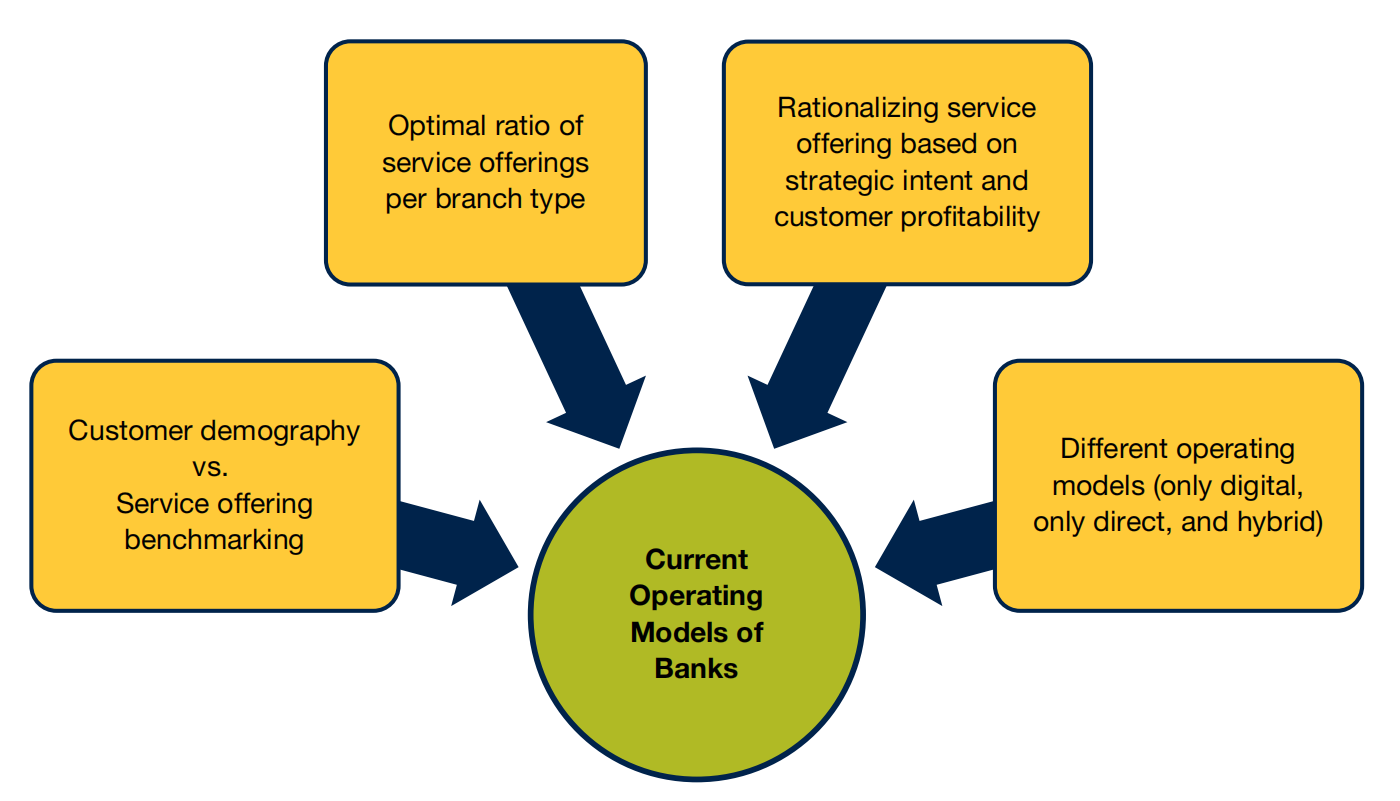
Some people argue that it is just the ability to draw 7 boxes and connect them with lines, while others consider it as a **critical section** that even a small decision made at individual source code line level will have an important affection on central architecture quality attributes (Bosch, Gentleman, Hofmeister and Kuusela, n.d.).

However, no matter what opinion you take, there is no denying that software architecture a software architecture is important in all areas of 21 century. Without it, much of what we know and use today could not be possible.

Today, in this paper we shall discuss the **case of the bank system** and **explain why** software architecture is important in such a system.

## Bank System Analysis

With the number of online banks and non-banks increasing rapidly in the bank value chain, traditional credit-based banks face great competitive pressure (Garlan, 2014). Therefore, reconstructing the bank system core is bound to happen to alleviate the cost pressure and evolve the operating model.



1. Current Operating Model of Banks (Capgemini, 2021)

### Purpose

The purpose of analyzing the requirement of bank is to **standardize** the writing of software and the **design** of software architecture, which aims to **improve the visibility** in the software development process and facilitate the control and management of the software development process (Bosch, 2004).

The analysis also facilitates the **communication and collaboration** between programmers and customers and serves as the original basis in order to obtain a wider application of this document to further customize the details of software development.

In addition, we also need to **clarify software requirements**, arrange project planning and progress, organize software development and testing, and facilitate the coordination between users and developers (Garlan, 2000).

Generally speaking, Bank's IT system has three main parts:

* + User deposit

The user needs to fill in the deposit form, and then submit it to the evaluation and verification personnel system, which records the depositor's name, address, ID card, number, deposit type, and deposit date. Information such as interest rate and password (optional) will be reflected to the user by the system after completion.

* + User withdrawal

If it is a withdrawal, the user fills in the relevant information about the withdrawal (withdrawal amount, withdrawal currency) and submits it. The system requires the user to enter the correct input after the system calculates the interest and prints the interest to the user.

* + User transfer

If it is a transfer, the user fills in the relevant information for the transfer and submits it. The system requires the user to enter the password to recognize the identity. After the password is verified, the system calculates the interest and reflects the information to the user. The system updates the database in a timely manner.

Based on the above real-world scenario, let’s define the function requirement and non-functional requirements for the bank IT system.

### Requirement Analysis

Part of Function requirements of a bank system:

1. Functional Requirement of Bank System

|  |  |
| --- | --- |
| **Requirement Name** | **Description** |
| **Saving Account** | A Saving account only has two types of basic functions: withdrawal and deposit. |
| **Checking Account** | Customers of banks can have multiple checking accounts and their checking account shall be able to do transactions based on saving accounts. |
| **Authentication** | The system shall be able to verify the user through either SMS or OTP to authenticate any kind of risky business. |
| **Transaction** | Users shall be able to transact after authentication or they can assign an agent to help them transact. |
| **Query** | Users can inquire about information such as how much money they have deposited through our bank counters or online banking. |
| **Administrative functions** | The administrator should have the authority to freeze the account or freeze an ongoing transaction. |
| **etc.** | ….. |

Part of Non-function requirements of a bank system:

1. Non-Functional Requirement of Bank System

|  |  |
| --- | --- |
| **Requirement Name** | **Description** |
| **Concurrency** | The system should be able to carry tens of millions of levels of concurrency |
| **Availability** | Internet banking should return the query result within 0.3 seconds with a probability of 99.8%. |
| **Simplicity** | Staff should be able to use the system proficiently after one week of training. |
| **Availability** | The system shall be able to use in most mainstream platforms such as Windows Linux & Android. |
| **etc.** | ….. |

After **identifying** both functional requirements and non-functional requirements, we can start **building up** the architect of the system.

## Reasons for Software Architecture

The architect is most useful when we are dealing with a project with a size ranging from medium to large scale as it acts as the foundation of the software system. For this bank project, we advocate a middleweight approach to architects which contains the following aspects (Gharbi, Koschel, Rausch and Gharbi, n.d.):

* + **Design the software architecture**
  + **Do realistic schedule based on the architecture**
  + **Quickly get into the market and incremental analysis**

These four aspects are not isolated and are determined by a series of decisions. These decisions are also the **reasons** why software architecture is important for banks.

### Inhibiting or enabling a system’s quality attributes

Software architecture determines whether or not a system can satisfy its required quality attributes. It is the **supremely architectural** that guarantees the functionality and quality required by the bank (Jaiswal, 2019).

A wide range of quality attributes is included in software architecture ranging from **reliability** to **security**. And we can evaluate these quality attributes according to the **goals** or **standards** and they can be either subjective or objective..

Lacking some kind of quality attributes can be deadly. A study conducted by the National Institute of Standards and Technology (i.e., NIST) shows that software defects caused by **lacking some kind of quality attribute** led to the **loss of US$59.5** billion each year in the United States. For example, in 2007, 12 F-16 fighters lost control and could not find the direction due to a small error in the GPS system. And one of the pilots died.

However, it is extremely expensive and time-consuming to address all the quality attributes. Most of the time, we are just focusing on some of them that is essential for our project. For the bank, there are some points which describes the need as well as the quality attributes that are of great importance for bank IT system.

1. **Bank systems need high performance.**

We are supposed to be very careful when dealing with the **time-based behaviour of elements**, especially the **critical section** that cannot be accessed by multiple users and the resource that needs a long time to execute. The bank system shall always be useful to ensure the **reliability** of the system (Johann, 2015).

1. **Bank system needs to be highly secure.**

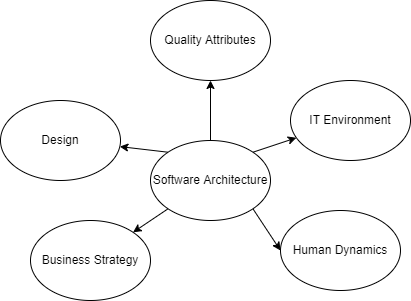
There is no denying that a bank system needs to be **highly secure** to protect people’s wealth. The **security** is so important that we may sacrifice a part of the user experience to improve the level of security. Varieties of methods should be taken including OPT verification, U disk shield, and SMS.

1. **Bank systems need scalability.**

**Incremental subset**s are likely to happen in the bank IT system because we may use this system for more than 20 or 30 years. All kinds of change should be taken into consideration during such a **long period** such as security teams may be retired or replaced, even the technology trend may change. Hence, an open architecture that can be added to subsets is very important for the **scalability** of bank systems.

All of these quality attributes are **determined** by the software architecture. It would not be possible to proceed without these **foundations** of system architecture (Maier, Emery, and Hilliard, 2001).

However, the architect itself does not mean **function** or **quality** and it’s just the starting point of the architecture. We need to have good architecture, but it’s **not enough** to just have architecture. There are many factors contributing to the software architecture.



1. Several Contributing Factors of Software Architecture

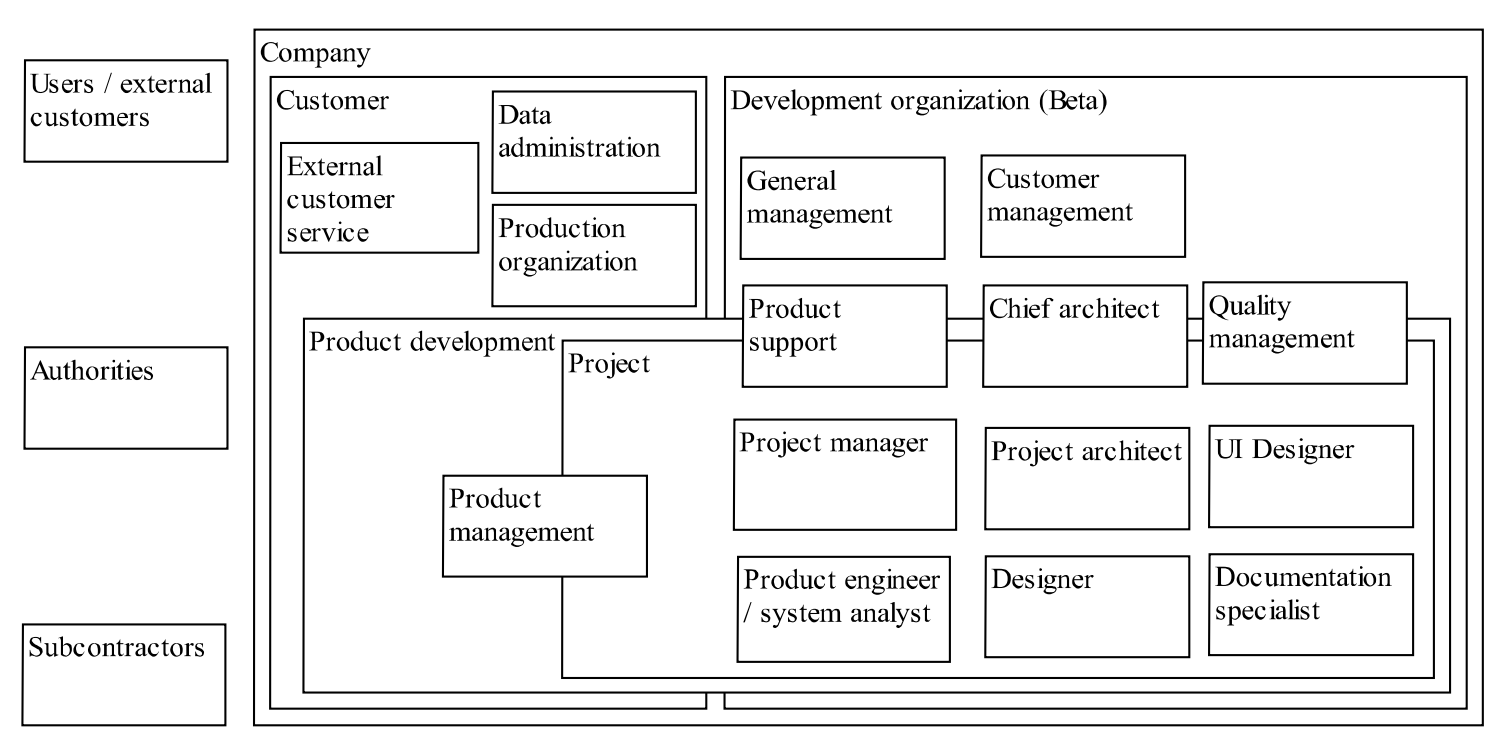
Generally speaking, in addition to the points discussed above, for the bank system, we still shall also check the following three categories:

1. The quality of the business affected by the architecture (for example, the time to market of the bank system).
2. The quality of the Bank system such as performance, usability, security, testability, modifiability, and usability.
3. Quality, such as conceptual integrity, is related to the architecture itself.

### Architecture facilitates communication among stakeholders

Most of the time, the software organization of the system is **determined** by the needs of stakeholders. However, these needs sometimes **contradict** each other. For example, the salesperson wants the banking system to be as simple as possible, while the risk control team wants the system to require users to fill in all the information in detail, so the not all **needs of each stakeholder can be met**.

However, a good architect will **compromise** these requirements as much as possible, and finally, deliver a banking system with good quality attributes to the demander. Software architecture act as a **basis for mutual understanding**, **negotiation**, **consensus**, and **communication**.



1. Mapping of identified stakeholder roles

Here are the main reasons why architecture **facilitates** communication among stakeholders and how software architecture is conducted to allow communication among stakeholders:

1. **Architecture allows stakeholders to understand**

Software architecture is so **abstract** and **hides the details** behind it that it allows most of the stakeholders to **understand** and conduct the software system without the help of engineers although they may have different perspectives about the system.

It is **primordial** for negotiation and represents a common abstraction of the communication between stakeholders (MEI, 2006). In this way, the architect can most effectively **capture** these requirements and **balance** points. They can understand why another aspect is important except their own concerning point. Consensus can be achieved. It would be extremely hard for them to understand each other without the help of the software architect.

1. **Architecture makes stakeholders compromise with each other**

Each **stakeholder** of the bank system, user, project manager, developer, banker, and so on, **compromises** with each other, and finally the architects gather and **integrate** their opinions to determine our software architecture and corresponding attributes (Qin, Xing and Zheng, n.d.).

For example, users may want to make it as convenient as possible. But the banker wishes to develop it with minimum cost. The manager is worried about whether or not the team can work independently with the system. What architects finally achieve is the **balance among several parties**.

Just like people say, there are a thousand Hamlet in the eyes of a thousand people, and there are a thousand concerns in the eyes of a thousand people. An architect plays the role of a common language in which different concerns can be negotiated and intellectual managed. Different requirements for a characteristic are expressed in this way.

1. **Revise project objectives based on the analysis of project stakeholders**

In order for all matters of concern to be able to cooperate fully, sometimes due to various reasons of stakeholders, we will have to modify the structure and architecture of the project to ensure that they can benefit from it. Although this approach may bring about an increase in scale and the risk of obscuring the original goal.

We may revise project objectives based on the analysis of project stakeholders, and accordingly, we will also let them make some compromises to ensure the overall system design.

Besides, the following graph can give us a quick review about the role of each stakeholder.

1. Role of each Stackholder

|  |  |  |
| --- | --- | --- |
| **Stakeholder Name** | **Emerging and emphasized problems** | **Emphasized rationale for architecture description** |
| **Customer** | Technical maturity  Communication and interpretation of meanings and descriptions  Security constraints | Communication  Understanding  Evaluation and deciding |
| **Designer** | Isolated views  Interpretation of descriptions | Understanding  Design & implement  Reuse  Documentation |
| **Project manager** | Resource or schedule  Constraints | Project planning  Communication  Understanding  Evaluation & deciding |
| **Architect** | Skills and experience of other stakeholders  Lack of resources | Communication  Evaluation & deciding  Documentation |
| **Customer management** | Understanding the descriptions  Skills and experience  Making resource estimates  Isolated views | Communication  Understanding  Licensing and pricing  Resource planning  Selling |
| **General management** | Knowledge management  Interpretation of descriptions  Organizational obstacles | Understanding  Communicating  Resource planning  Quality management |

\*Adapted from graph of Kari Smolander

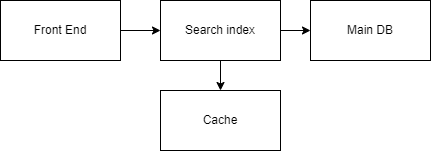
### An architecture may influence the organizational structure

The analysis generated from the software architecture not only will influence the software structure but also will **affect** the **organizational structure** of the development team. For a large project, we use the method that we called **work-breakdown**. We **divide the labor** and **assign the tasks** according to different portions of the system. This in turn dictates units of planning.

For example, our architecture can have a data access layer to process user input and output. Of course, we will **assign** this **module** to a team with corresponding skills to maximize benefits. Finally, we can **divide the task** and create a team that can perfectly work together to optimize the performance of the system.

This is especially the case for a huge project like a bank system. We may have a special team for each module. The **organizational** and **work-breakdown structure** is always based on its architecture. Each team is **responsible** for a **module** and they focus on it without concern on the other parts except the final incorporation to check the system integrity (Schmidt, Stal, Rohnert and Buschmann, n.d.).

The following graph gives us an example of mapping from architecture to the team that built the application.



1. Mapping from architecture to development team

### Architectures improve cost and schedule estimates

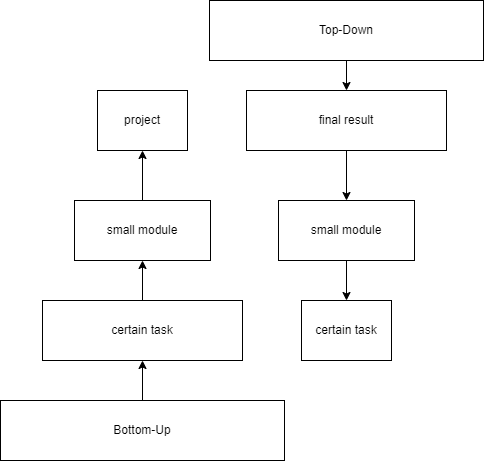
**Cost and schedule** are always the key **concerns** for project management. In many cases, funding will be an important part of ICT projects. For example, in 2003, the British government spent 2.3 billion pounds on ICT projects, which was higher than the 1.4 billion pounds spent on railway projects. Nearly half of the budgets are useless because the government refused to add more money. If the funding of an ICT project cannot be estimated and managed well, it is very likely that the project will fail.

Meanwhile, establishing a software architecture is a good way to **manage the cost** of developing such a bank system. An architect can help the project manager have a **bottom-up** understanding of the development of the system and create an accurate estimate of the schedule and budget of the teams.

*Top-down: Start from the final result and divide it into small modules.*

*Bottom-up: Start from a certain task and combine them into the project.*

Without architecture, it would not be possible to predict when and where the system will encounter the problem and get the key information required to manage the development life cycle for the project manager. A **top-down** estimate will also be conducted to double-check and find the consensus with the bottom-up approach (Soni, Nord and Hofmeister, 1995).



1. Top-Down vs Bottom-Up

Based on the understand of the system pieces and estimate the cost of each of them will be more **accurate** than the prediction made just according to the experience and overview of the requirements. The more knowledge about the system, the more accuracy you will get.

## Conclusion

Although most of the time, we just want to jump in and start coding. However, as the foundation of the software system, software architecture, has a profound effect on the quality of what is built on top of it.

Acting as the blueprint of the system, software architecture is about making fundamental structural choices. It includes specific structure options for the design ranging from IT environment to business strategy. Additionally, perhaps we can get more benefits from software architecture if we can properly handle it (ZHANG, 2008).

software architecture, like what we are talking about in the bank system case, is a series of decisions that can strongly affect the whole organization. It is also the earliest decision that determines the design of the bank system and plays a vital role in future implementation.

**References**

Bosch, J., 2004. Software Architecture: The Next Step. *Software Architecture*, pp.194-199.

Bosch, J., Gentleman, M., Hofmeister, C. and Kuusela, J., n.d. *Software Architecture: System Design, Development and Maintenance*.

Capgemini, 2021. [online] Capgemini.com. Available at: <https://www.capgemini.com/wp-content/uploads/2017/07/simplifying\_the\_banking\_architecture\_2015.pdf> [Accessed 23 November 2021].

Cuesta, C., 2010. Software Architecture and Future Challenges. *International Journal of Software Architecture*, 1(1), pp.1-2.

Garlan, D., 2000. Software architecture. *Proceedings of the conference on The future of Software engineering - ICSE '00*,.

Garlan, D., 2014. Software architecture: a travelogue. *Future of Software Engineering Proceedings*,.

Gharbi, M., Koschel, A., Rausch, A. and Gharbi, M., n.d. *Software architecture fundamentals*.

Jaiswal, M., 2019. Software Architecture and Software Design. *SSRN Electronic Journal*,.

Johann, S., 2015. Software Architecture for Developers. *IEEE Software*, 32(5), pp.93-96.

Maier, M., Emery, D. and Hilliard, R., 2001. Software architecture: introducing IEEE Standard 1471. *Computer*, 34(4), pp.107-109.

MEI, H., 2006. Progress of Research on Software Architecture. *Journal of Software*, 17(6), p.1257.

Qin, Z., Xing, J. and Zheng, X., n.d. *Software Architecture*.

Schmidt, D., Stal, M., Rohnert, H. and Buschmann, F., n.d. *Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects*.

Soni, D., Nord, R. and Hofmeister, C., 1995. Software architecture in industrial applications. *Proceedings of the 17th international conference on Software engineering - ICSE '95*,.

ZHANG, L., 2008. Software Architecture Evaluation. *Journal of Software*, 19(6), pp.1328-1339.

**APPENDIX1**

**Marking Rubrics**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Component Title** | **Assignment** | | | | | **Percentage (%)** | **15** | |
| **Criteria** | **Score and Descriptors** | | | | | | **Weight (%)** | **Marks** |
| **Excellent**  **(5)** | **Good**  **(4)** | **Average**  **(3)** | **Need Improvement**  **(2)** | **Poor**  **(1)** | |
| **Format** | **Strict compliant** | **Compliant in most parts of the document** | **Compliant in some parts of the document** | **Non-compliant in most parts of the document** | **Completely non-compliant** | | **3** |  |
| **Explanation about the significance of software architecture in a bank** | **Relevant and comprehensive** | **Mostly relevant and comprehensive** | **Relevant and comprehensive to the moderate extent** | **Mostly irrelevant and non-comprehensive** | **Completely irrelevant and non-comprehensive** | | **12** |  |
|  |  |  |  |  |  | |  |  |
|  |  |  |  |  |  | |  |  |
| **TOTAL** | | | | | | | **15** |  |

Note to students: Please print out and attach this appendix together with the submission of coursework