Hybrid Software Architecture Design Pattern Model

Asebe Jeware
VisionFund Ethiopia, Addis Ababa, Ethiopia
asebejeware@yahoo.com

Nassir Dino
HiLCoE School of Computer Science and Technology, Addis Ababa, Ethiopia
nassir@hilcoe.com.et

Abstract

The concept of implementing software architectural pattern in designing software structure is key for system development process efficiency and well accepted by the society of software engineering. Though there are different ideas regarding the detail implementation and documenting the structure of a software system, finally, the size and complexity of these systems compelled all stakeholders to have it in a well organized manner. Therefore, the implementation of single or uni-architectural pattern throughout in solving system recurrent problems and designing the whole system has its own limitation. Monolithic approach, in general, has a limitation of preserving the advantage of multi-pattern capabilities that is not found fully in one pattern.

The research had gone through extensive reviews of a number of related patterns and their characteristics, observation of working systems mainly in the area of banking solutions, informative interviews, and critical document and literature review. This contributes to design highly efficient system architecture model known as hybrid architecture that allows us to consider more than one architectural pattern in a system. Thus, the specific result of this research is a model with the procedure in implementation of hybrid architectural design. The solution pattern model result is validated using a case study.

Keywords: Software Architecture; Designing Software; Software System; Hybrid Architecture

1. Introduction

The emergent of architectural patterns with some different qualities and purposes follows selection of architectural patterns in designing software architecture. These architectures have their own significant contribution for software development but most of the time one architectural pattern may not be sufficient in designing the whole aspects of system attributes, especially when it is used to implement program architecture that encompasses different projects in the enterprise. In order to solve this, there should be a systematic approach to hybridize different software architectural patterns that we can implement on designing the appropriate architecture for enterprise wide program architecture.

The rest of the paper is organized as follows. Section 2 covers the background. Section 3 presents related works to identify the gap in prior works. Section 4 discusses the proposed solution, and finally in Section 5, the work will be concluded with some concluding remarks.

2. Background

Hybridizing means a process of bringing two or more different things into one by mixing them together. In this paper we consider architectural patterns as a thing to be hybridized in order to create a hybrid architecture which can maximize bringing of a solution for recurrent problems in software development.

As the authors in [1] discussed, the notion of a pattern is "geared toward solving recurring problems in design." But a pattern is more than just a battle-proven solution to a recurring problem. The proposed solution involves some kind of structure which balances these concerns, or "forces", in the manner most appropriate for the given context. So a definition which more closely reflects its use within the patterns community is [1]:

> Each pattern is a three-part rule, which expresses a relation between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain software configuration which allows these forces to resolve themselves.

In this paper, pattern as an element in the world is applied to implement a relationship in a certain context, a certain system of force to resolve a problem which occurs repeatedly in that context and a certain spatial configuration by hybridizing two or more architectural patterns within single software architecture.

3. Related Work

In this Section different related literatures are reviewed in order to find the gap between what is done and what is expected to be done.

As the authors in [2] discussed, an Information Filtering system of HTML/Text documents is collected from the World Wide Web (WWW), based on a representation of user interests as inferred by the system through a dialogue. One of the distinguishing features of the system is the use of a hybrid approach to user modeling, in which case-based components and an artificial neural network are integrated into one coherent system. Moreover, in order to perform an accurate filtering, the system takes advantage of semantic networks and a well-structured database.

As Aslam *et al*. stated in [3], modern WSNs consist of a number of wireless sensor devices sometimes called "*nodes*". A node can be a low-end or a high-end device with different storage capacities, processing capabilities, and software architecture. These nodes form networks to collect sensory data and transfer it to a base station or sink, sometimes referred to as gateway. The data can be used by a number of different applications managing and monitoring the network. In certain environments, these networks also comprise of actuators forming Wireless Sensor and Actuator Networks (WSANs). Open Framework Middleware (OFM) is a distributed, service orientated model driven system. However, the core issue facing OFM comes from the node level where nodes limit the function of what OFM is intended for, i.e., using models to define WSN at all levels of the network.

The research has highlighted the limitations associated with the traditional layered protocol stack approach and has shown that it restricts network software modularity. The rigidity of the layered protocol stack approach and the lack of information sharing between protocol layers impede optimal network performance as shared layer information is a prerequisite for network optimization in wireless heterogeneous environments.

For OFM to fulfill its claims, a hybrid architecture is proposed which removes the stack based protocol layers and places emphasis on running a service oriented OFM micro middleware over the device abstraction level. However, it is fully concerned and implemented on network service and wireless connectivity enhancement.

As the author in [4] discussed, to distribute video and audio data in real-time streaming mode, two different technologies - Content Distribution Network (CDN) and Peer-to-Peer (P2P) - have been proposed. However, both technologies have their own limitations: CDN servers are expensive to deploy and maintain, and consequently incur a cost for media providers and/or clients for server capacity reservation. On the other hand, a P2P-based architecture requires sufficient number of *seeds* supplying peers to *jumpstart* the distribution process. Compared with a CDN server, a peer usually offers much lower out-bound streaming rate and hence multiple peers must jointly stream a media data to a requesting peer.

Although the implementation of this hybrid architecture is successful on streaming media distribution, it is not developed considering software and software architectural patterns.

4. The Proposed Solution

The proposed model is produced with the intention of implementing more than one architectural pattern within one system architecture in order to have highly strong system design which can implement all system requirements.

The hybrid architecture model will bring significant change for architects and developers, and minimize development time in having efficient and effective system architecture.

It is created based on the ISO-9126-1 quality attributes specification as the first and priority to proceed with this research steps.

It is listed 1 up to 8 as follows:

1. Analyze the main functional requirements and nonfunctional requirements for the system, to establish the quality requirements and quality goals.
2. Use the customized ISO 9126-1 quality model for the architecture as a framework. Some of the metrics could be further specified, according to specific components and/or connectors.
3. Present the initial candidate architectures based on project nature, quality requirements, and architectural patterns capability to solve quality requirements.
4. Construct the comparison table for the candidate architectures.
5. Prioritize the quality characteristics taking into account the system's quality requirements and quality goals. The customization of ISO 9126-1 to the problem domain can be used to organize hierarchically the characteristics.
6. Analyze the results summarized in the table, according to the given priorities obtained in step 5
7. Select the initial architecture, among the evaluated candidates, on the basis of the previous analysis.
8. If a finer analysis is required, scenarios or profile-based approaches could be used, considering only the quality characteristics relevant to the problem domain, obtained in step 5.

Once the aforementioned activities are completed as indicated, we need to follow the steps listed below to hybridized best fit architectural patterns. The steps listed starting from 9 up to 12 are the result of this research to hybridize different architectural patterns.

9. Based on step 7, check the selected architecture is best fit (if the pattern is recommended for that application) architectural pattern that can implement most of (if more than 50% of the quality requirements are implemented) the requirements specified and recommended to be applicable to that specific system from pattern database.
10. Identify the weaknesses that are found in the selected architectural pattern.
11. Select additional architectural patterns from the list which have the capability to give solution for the weakness that is found in the best fit architectural pattern.
12. Hybridize them taking into consideration the strength in which they are very effective (recommended) in the system in the way they can work together.

To consider one architectural pattern as best fit pattern, the following criteria should be fulfilled. The requirements specified and the pattern attributes strength should be aligned (most of the requirements specified must be addressed by that architectural pattern). The phrase 'most of the requirements' is used to show the number of requirement specifications in percentage which is approximately attainable above fifty percent of the lists. The reason why 'fifty percent' is given, is because of different architectural patterns capability to implement the system requirements is up to the level fifty percent is similar, but above that threshold, we can find only one architectural pattern.

4.3 The hybrid pattern model

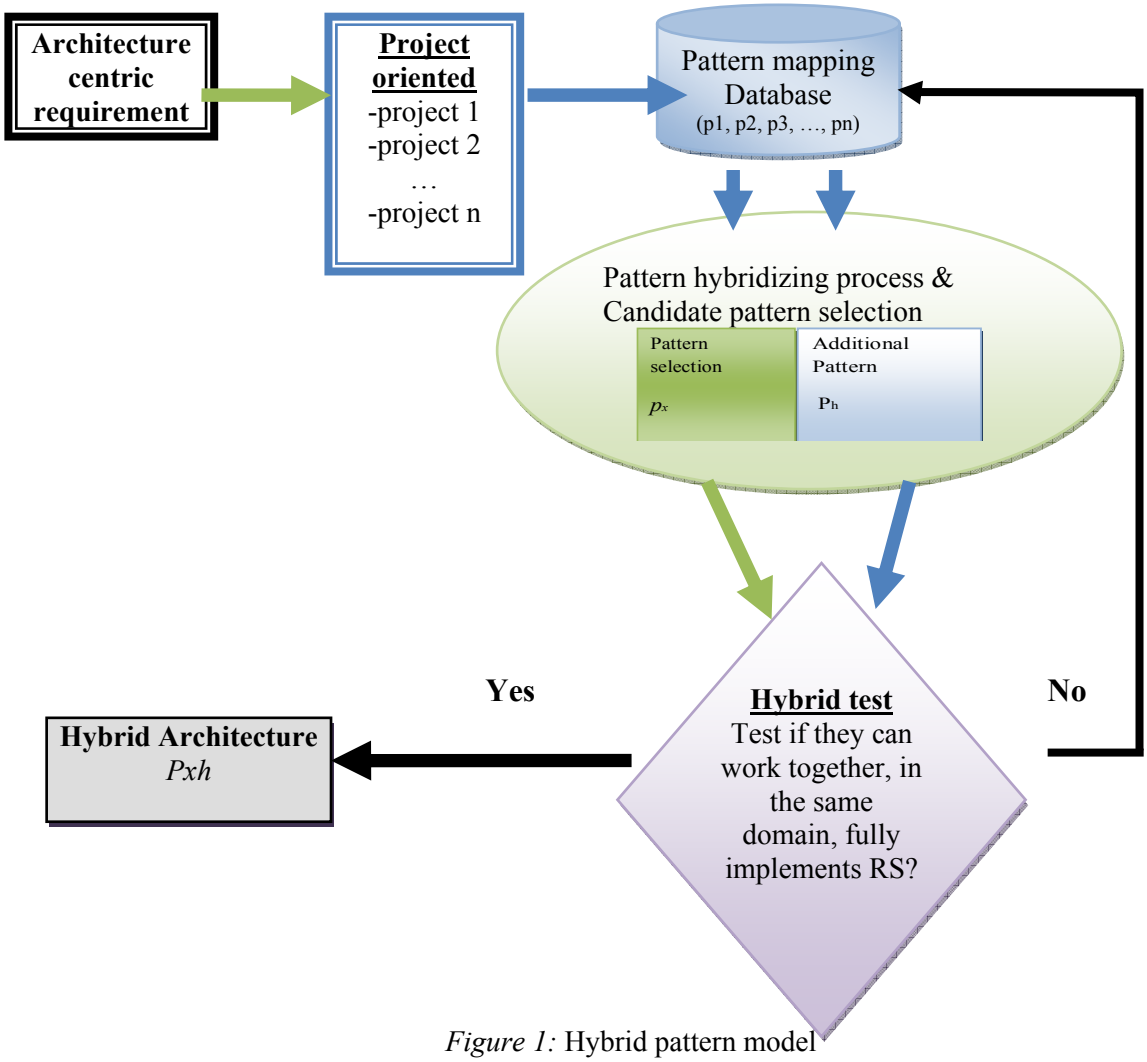

Figure 1: Hybrid pattern model

4.3.1 Architecture centric requirement

The first step is concerned to identify architecture centric requirements from all listed requirement specifications. Architecture centric requirement is a requirement mainly concerned with the quality attributes of the system.

4.3.2 Project oriented architecture

The second box represents selection of architectural patterns that can fit into the project nature. The whole project is divided into different projects and different architectures will be implemented based on what the specific project demanded. In this step we will look into the applicability of the specified architecture inside that project. The architecture selected should be fully capable to implement all the projects' requirements or at least most of the system requirements.

4.3.3 Pattern Mapping Database

Pattern mapping database is the repository of all patterns with their quality characteristics and sub-characteristics, applicability where they can be best implemented and well accepted in the implementation, with their strength and weakness to indicate architects and other stakeholders how and when to use them and with recommended combination of architectural patterns or architectures that can be hybridized with specific architectural pattern, which can simplify the process of architecture selection in the work of architects and to use them together in system architecture design.

4.3.4 Pattern hybridizing process

At this stage the process of pattern selection will take place inside pattern hybridizing process circled by oval shape. At this stage we should identify the

weaknesses of the first best fit architectural pattern that make it insufficient from implementing the whole requirements. The second architectural pattern that is going to be hybridized with the previous architecture should have the upper hand weakness of the first architectural pattern quality attributes. That means it should have strong quality attributes that can solve the weakness of the first architectural pattern. Once we identify the additional pattern in which we can resolve the weakness of the first pattern, we hybridize them to implement it in the whole system architecture design. This process will continue till we finish solving all quality requirements found in the system and the weakness that are associated with the selected architectural patterns.

4.3.5 Hybrid test

Hybrid test is made to check or to test if the selected architectural patterns work together, if the second architectural pattern is in the same domain.

4.3.6 Hybrid architecture

The hybrid architecture is produced considering all requirements listed in the quality requirement and passes through the test case found in hybrid test stage.

The designed solution architecture considers additional architectural patterns and implements their strength in the manner of resolving the weaknesses found in the best fit architectural pattern. In order to select the second architectural pattern, as it is mentioned earlier, we should identify the weaknesses found in the first architectural pattern and then select the second architectural pattern which can resolve or give remedies for most or all of the weakness found in the first architecture.

5. Case study

In validating the proposed solution, a case study has been conducted on one institution which is engaged on micro financing activities. In the case study, four architectural patterns have been deployed to give solution for each and every system requirements. They are OOA, SAO, N-tier, and Layered architecture which are all capable to work together.

6. Conclusion

This paper focused on how to hybridize architectural patterns into the system architecture of any kind of software system without losing each pattern's respective essence. As a result, one general hybrid architectural model is produced, which can simplify the work of architects and all stakeholders; practitioners and the academia. The research uses selected architectural patterns' attributes and each characteristic of most architectural patterns and system requirements. It assists in grouping quality attributes and newly emerging quality attributes that should be considered in system architecting.

It is not recommended to conclude that architecture is all about hybridizing and to implement hybrid architecture for all types of software development. It is very efficient if we use it for big multi-project software development and for complex systems like core banking systems and huge machine controlling systems.

References

[1] Hassan Saad Almari, "Investigation of the Relationship between Software (Architecture/Design) Patterns and Quality Attributes", The Australian National University, Australia, 2009.

[2] Leonardo Ambrosini, Vincenzo Cirillo, and Alessandro Micarelli, "A Hybrid Architecture for User-Adapted Information Filtering on the World Wide Web", Università di Roma Tre, Italy.

[3] Muhammad S. Aslam, Susan Rea, and Dirk Pesch, "A Vision for Wireless Sensor Networks: Hybrid Architecture, Model Framework and Service based Systems", NIMBUS Centre for Embedded Systems Research, Cork Institute of Technology, Ireland.

[4] Dongyan Xuyz, Sunil Suresh Kulkarniz, Catherine Rosenbergz, and Heung-Keung Chaiz, "A CDN-P2P Analysis of a Hybrid Architecture for Cost-Effective Streaming Media Distribution", Department of Computer Sciences, School of Electrical and Computer Engineering, Purdue University, USA.