Student: Yifei Pei ID: 1611648 Title: Report 1 Course: Software Architecture

Lecturer: Katrina Falkner Deadline: 28 August 2013

***The difference between architectural styles and design patterns is…?***

**Introduction**

With the exploding development of the academic exploration in Software Architecture, the knowledge of Software Architecture and software related architectural concepts have enormously spread. Some general terms of Software Architecture are also mentioned massively in both academical papers and business reports. As beginners of Computer Science, some pioneers have revealed that not all terms are defined clearly enough for them to understand. Furthermore, they even found out that some experts in Software Architecture may also get confused about some specific terms, or on the other hand, they just have different academic assertions for the same concepts. The vague defined concepts, architectural styles and design patterns, are one pair of those frustrated concepts. In plain English, it seems style and pattern are synonyms. In the context of software developing, architecture and design are also not something totally different. Some beginners start to ask, how do we differentiate them? Unfortunately, there are no industrial agreed definitions for them to clearly differentiate them since two different understandings can be found in academic papers. In Gorton[1], it introduced the two concepts at the same thing by saying “Patterns are styles” and use pattern as the preferred concepts in Software Architecture because it extended the context of the concept and generalise the concept better. However, in Taylor *et al*.[2], the two concepts have different definitions, which give the word “pattern” the recurred form and more extended insight than the more generalised “style”, even the two definitions are still read similar. The multiple definitions gain more debates on the two concepts. In this short article, their definitions will be discussed first, and then some insight for the difference will be given.

**How do we define Architectural styles?**

The original meaning of architectural style referred to the term of real world construction which characterise the different design of architectures into different styles. The usage of this term in real world construction is a bit excessive, which also explained why people tended to use “pattern” rather than “style” in software architecture. If only “architectural styles” is searched, some professional definitions can be found.

Shaw and Garlan[3] defined it as

> An architectural style defines: a family of systems in terms of a pattern of structural organisation; a vocabulary of components and connectors, with constraints on how they can be combined.

From this definition, we can infer Architectural styles as a magazine for powering software architecture to give characterised architectural solutions for softwares which can solve complicated problems. Those styles are general norms for software architectures that can be clustered and scalable.

Normally people may regard the last definition something too simple or too general, furthermore, Taylor *et al*. gave us a more specified definition, which is

> An Architectural Style is a named collection of architectural design decisions that (1) are applicable in a given development context, (2) constrain architectural design decisions that are specific to a particular system within that context, and (3) elicit beneficial qualities in each resulting system.

This definition gave us more details to follow and more implementable illustrations for applications. Comparing with the last definition, this definition gives specifications for architectural styles that the ranges of the styles which can deal with different specific problems are also key for the concept in spite of the generalisation for the concept. Later, let’s get some insight for design patterns so the comparison can continue.

**How do we define Design patterns?**

Design pattern is the concept derived in Software Architecture which describe the specific formats for the different parts in software that can help solve problems. Generally, from the name, we can differentiate design patterns as something more specific than architectural styles, though some industrial professionals still believe the two concepts are not differentiated magnificently.

Taylor *et al*. also defined design patterns that

> An Architectural Pattern is a named collection of architectural design decisions that are applicable to a recurring design problem parameterised to account for different software development contexts in which that problem appears.

This definition proved the deduction in last paragraph. From the definition, we can tell that design patterns need to deal with some specific parts of the software or specific problems, which may be concrete to the software architect in a macro point of view. Other than architectural styles, design patterns have some certain types in terms of different aspects for designing the software but not in terms of different environments.

For example, design patterns can be characterised as algorithm strategy patterns, computational design patterns, execution patterns, implementation strategy patterns,structural design patterns, etc., which are different parts inside the software or different implementation level for the software. Otherwise, architectural styles have some types for the softwares that have different usage or implementation environment. For example, structure type can fit for different requirements, messaging can fit for different implicit modes, distributed system can fit Internet based remote controls.

However, Martin[4] also stated that at higher level there are also design patterns that are large in scope, describing patterns followed by entire system.

**The difference between the two concepts**

As stated by Microsoft[5], some people treat design patterns and architectural styles as the same, however, some treat styles as specialisations of patterns. They have something in common, because they are both idioms for software architects to use, which provide a common “language” or “vocabulary” to describe classes of systems. Nevertheless, some also believe they have differences. For instance, one of the main differences is that a pattern can be seen as a solution to a problem, while a style is more general and does not require a problem to solve for its appearance.

Rotem[6] described the differences between the two concepts as the distinction between architecture and design, since the explicit wording difference can be easily identified. In his argument, architectural styles effected the solution globally or at least it affected major parts of the solution and not solve local issues. He also mentioned that some architecture styles had been well-known before the notion of design patterns for software was introduced.

Depending on the above discussions, if we do want to treat the two concepts as two different attributes for software architecture and we want to differentiate the two concepts in more details, we can summarise as follows. The styles describe types of architectural elements, types of relationships and rules about the elements and the relationships themselves. On the other hand, patterns include rationale about how the solution really solves the problem in the specified context.

In an application view of using software architecture, we can describe the difference more expressively. Normally, a software project will contain multiple architectural styles and design patterns. Even styles are for the macro relationships between the elements inside the project, there may be multiple styles to define different aspects of the software. For example, if the software wants to control a robot remotely, it might use client-server

style for the whole flow control and the communication between the host and robot and also use event-driven structure for the different modes implemented by the robot. They are defining the interrelating way for the project. Simultaneously, the designer should also specify the use of interfaces and abstraction for the different classes inside the structure depending on the requirements for the project and the flexibility for the communication between different parts. For instance, designer needs to decide whether to use adapter patterns or bridge patterns to make abstractions. With the interaction of different styles and patterns, a optimised architecture can be designed for the best fit to the requirements, environments, and the nature of the software.

**Conclusion**

Generally speaking, architectural styles and design patterns are quite common in strategically and tactically designing the software architecture in practice. Even though there are many referenced articles can specify the differences between them, academically they have many overlapped elements. This article just discuss the two concepts in an introductory level, more detailed conclusions need further research to support.

**References**

[1] Gorton, I. (2011) Essential Software Architecture, 2nd ed. Chapter 1. Available at: http://forums.cs.adelaide.edu.au/file.php?file=%2F512%2FGorton-Chapter.pdf

[2] Taylor, R. N., Medvidović, N., and Dashofy, E. M. (2009) Software Architecture - Foundations, Theory and Practice, 1st ed. John Wiley & Sons, Inc.

[3] Shaw M. and Garlan D. (1996) Software architecture: perspectives on an emerging discipline. Prentice Hall.

[4] Martin, R. C., (2000) "Design Principles and Design Patterns".

[5] Microsoft developer network, http://msdn.microsoft.com/en-us/library/ee658117.aspx

[6] Rotem, A., (2010) Modeling - Architectural Styles. Available at: http://arnon.me/2010/05/modeling-architectural-styles/