**LALITHA MADHURI PUTCHALA**

**My Central ID: 700657631**

[**Lxp76310@ucmo.edu**](mailto:Lxp76310@ucmo.edu)

**CS-4910, CRN -13944 FALL 2016**

**Paper Title - *Design Reuse Through Frameworks and Patterns***

**Author: *Peter W. Fach***

**Summary:**

This article reviews briefly the reasons for implementing software **“design reuse”** methodologies in large scale organizations like banks and insurance companies. The article comes up with the idea of developing frameworks and design patterns for reuse functionalities. This notion is well backed up by a real-time example where a German company RWG, used an application software “Gebos” based on these frameworks to provide adaptability reuse service to more than 17000 banks for past 10 years. Initially, an ephemeral description of **“frameworks”** is provided in the article supported by the concept of **“sedimentation”** through which the robust design reuse functionality is achieved. The architecture feasibility to accommodate business layer and application domain layer (ADL) is discussed along with operations among them for **“intensive design reuse”.**

Secondly, the article focused on the **“Design Patterns”** to document behavior of the frameworks and the interplay between them. Two types of design patterns are discussed – a) The Tool – Material Pattern b) The Role Pattern. The **“tool-material pattern”** describes the interplay between application domain objects and objects for interaction purposes. This pattern is a good example how we can get conceptual models to converge. The **“role pattern”** makes use of everyday business transactions to serve as metaphors for structuring the software architecture. This pattern handles technical challenges like fast system response time and smooth sedimentation process in diverse adaptable workstations. The role pattern is applicable when developing frameworks for industries where application domain has been stable for many years.

Thirdly, the author stresses on the fact that the **“application adaptability”** could be achieved. As individual frameworks all use the same customer core but operate independently, we can easily develop task-specific applications in an elegant way. Role pattern offers practical and elegant possibilities for tailoring an application to particular activities. The Tool – Material pattern provides adaptability features by making a set of tools and materials available, thus addressing the distinct requirements of different types of workplaces. To ensure that all tools are reused precisely for larger applications, “style guides” are required. Style guides for framework development must take in to account these two issues – a) Criteria for reuse b) Usage Models. Failing to tackle above issues leads to misuse of the tools, thus, slowing the process.

Lastly, the article focusses on the impact of **“human interaction” (HCI)** and user participation in the design reuse in the early stages of software development. It states that the users should have typical knowledge as well as the technical design process of frameworks. Users not only help to develop suitable metaphors for design patterns but also ensure that these metaphors are realized consistently in the development process. Complete catalogs of technical solutions, documented as design patterns using metaphors are now available These have potential to support the convergence of design models and conceptual models. Ex: Tool Material and Role Patterns. Therefore, with the help of frameworks and appropriate metaphors for design patterns, it should be possible to realize in software development that a pattern language would provide well-designed and successful solutions.

**Strength**:

The paper is written in very clear, strong and articulate language. The main strength of the paper lies in author explanation of each concept with a demonstration of practical example. The author quoted a real-time example of a German company RWG which provided a robust adaptable software solution to almost 400 heterogeneous group of banks. The paper research is right up to the point and the concepts are described clearly at every point. The architecture was well supported by a diagrammatic description. The research paper covers the main idea and was written in a proficient way. The structure of the article is even and flow of the information is well managed, making easily understandable for the users. All the sources of information are well referred which will be helpful for the fellow researchers reading this passage. In particular, the second diagram is innovative enough to understand the scenario of the application adaptability of design reuse, which is quite informative.

**Weakness and Suggestions:**

Even though the paper explained the concepts with examples, the description is too short which will become tough for readers who have no knowledge in software development life cycle process. The paper lacks major phase of software development cycle i.e. testing the quality of the frameworks and design patterns. It would have been great if the article mentioned about the usability testing methods for the frameworks and design patterns for reuse. The architecture diagram is too vague, it could have been more clear and include important functionalities as components of the diagram. This would give clear idea about the end to end process of the design reuse concept in a software development lifecycle. The article missed the proper conclusion and summary at the end. It neither spoke about the past literature review present on the design reuse nor mentioned about the future implementations in frameworks and design reuse patterns. All and all it is just a black box explanation and in-depth concepts are covered. It could have been great if the research spoke about the disadvantages of using such kind of frameworks, the other side of the coin. Also, it did not mention whether these reuse methods are applicable for small scale organizations. It would have been complete article journal, if it has covered and demonstrated more clearly all the above missing points.

**Future Work:**

As the software development happens in different ways in different scenarios, a focused research has to be developed in-order to tackle the challenges with the induct of new functionalities in the final stages of development. As we are entering into **big data** era, the studies should concentrate on developing frameworks and design patterns for huge data applications. The solutions should integrate big data applications like Apache’s Hadoop, Spark and Pig in the development of the frameworks for large scale industries. The **Internet of Things (IoT)** is coming and as of 2020, according to Gartner there will be 34.8 billion devices connected making the architecture more complex with heterogeneous technologies. The innovative approaches are to be followed to withstand the challenges that will be introduced of IoT networks in future. The paper never mentioned about the integration of frameworks in different platforms. More software engineering and testing research has to be concentrated for verifying the integration of frameworks and other design patterns in a software development life cycles. The **novel** **software standards** have to be developed to accept the frameworks and design patterns in to the market.