Paper Review

Current Challenges in Practical Object-Oriented Software Design

CSN- 291: OOAD

Jitesh Jain [19114039] - Overview and Challenge A

Ayushman Tripathy [19114018] - Paper Review

Shreyas Dodamani [19114079] - Challenge B

Shashank Aital [19114076] - Challenge C

Piyush Dagdiya [19114025] - Future Research

Overview

In the paper, the authors focus on developers' various practical challenges while designing **modern** software in an object-oriented way. They mainly focus on the obsolescence of the existing object-oriented design knowledge when solving contemporary challenges.

Summary

Throughout the paper, the authors describe the following three main challenges and propose solutions to solve them:

A. *The Impact of Software Architecture and Technology Stacks on Software Design*

- In today's world, no software development team starts developing software from scratch. Most developers use existing libraries, frameworks, or extensions to achieve their goals, which impose several constraints on the developers, and hence, the models need to be incorporated, satisfying the conditions.
- **Current Solutions**
    - Developers are encouraged to separate the development from the requirements of the application.
- **Proposed Solution**
    - We need to acknowledge the fact that modern software is often heterogeneous, distributed, and composed of a large set of different frameworks and libraries.

B. *Multiple Models in Large Complex Domains*

- The complex world domains often involve several business flow operations related to different stakeholders who develop different "views" of the same problem, which forces the development of several other model representations.
- In practice, this means that developers should not only model the main entities and their actions through different perspectives, but they should also model business events and how these events change the current state of the model.

- **Current Solutions**
    - The authors mention Evans' Domain-Driven Design approach that aims to divide large models into different "Bounded Contexts" and build a "Context Map" that explicitly shows the relationships between the different contexts as a possible current solution to the challenge.
- **Proposed Solution**
    - There needs to be a collaboration between software engineers and requirements engineering researchers to understand how multiple models interact, evolve, and are maintained together, not only from an abstract level but also from the implementation point of view.

C. *Contextually Measuring the Quality of Object-Oriented Software Design*

- The authors suggest that the existing metrics fail to capture that software system's context (architectural-wise or domain-wise).
- In the paper, the authors conjecture that an essential cause for the large number of false positives that the static analysis and code quality

measurement tools currently generate is their lack of context. They further suggest that understanding the context in which a measurement makes sense is essential.

- **Current Solutions**
    - In the present situation, Code-Smells are used as a measure to determine good and bad parts of a code, and correction of code is done based on them.
- **Proposed Solution**
    - The authors suggest developing empirically derived theories on the characteristics of complex models that should be considered high quality and in which context and numeric ways to measure such features.
    - Machine learning and data science may play an essential role in the field.

Paper Review

Strongholds of the Paper

- The paper takes into account the problems faced by the conventional design techniques.
- The problems are presented in a **realistic manner** in tune with the current software-design problems encountered by contemporary software engineers.
- The author suggests some ideas to tackle the problems discussed.
- It gives the reader an all-new thought to a problem that was not addressed earlier in a proper manner and given much thought to.

What the paper lacks

- The ideas proposed by the author are quite abstract and **generic**. These don't cater to a specific problem encountered in a specific domain, which raises questions about their **feasibility**.
- Solutions proposed haven't been discussed in detail. These are just enough to initiate the thought process. A lot of work needs to be done for implementing the proposed ideas.
- The author suggests having a **generic metric** to measure software quality. But, this would lead to the algorithm becoming **bulky** and difficult to code and adapt to as this is a highly abstract concept. This raises questions about the **practicality** of the idea of having a quality measurement technique. Also, no proof has been given that making such a method is even possible.

Future research

With the growing complexity of software in the coming years, handling and evaluation are becoming more and more difficult. Thus, comes the need for proper design and monitoring of the software, so that the software becomes extensible and easy to debug. So, a lot of work needs to be done.