Software Reengineering: Reverse Engineering with Using 4+1 Architectural Views and Forward Engineering with MVC Architecture

Keywords:
Software Reengineering, 4 + 1 Views, 5W1H Re-Doc, MVC

Abstract

Software complexity is increasing exponentially in our modern era. As this trend continues, software architecture becomes increasingly important. Without proper software architecture, an application must be rewritten for different uses such as web or mobile app. The separation of tasks through architecture allows each programmer the opportunity to limit their need of understanding to only the portion of code for which they are responsible for, thus saving large amounts of time. This research was implemented through the use of 4 + 1 architectural views using a Unified Modeling Language (UML) for reverse engineering as well as the forward engineering phase in order to redesign an existing legacy game application into Model-View-Controller (MVC) architecture. Software re-engineering involves first reverse engineering the legacy system to a visual model in order to understand the design and its functionality. The legacy system is reverse engineered using 4 + 1 architectural views in order to comprehend the different aspects of the system. After the 4 + 1 architectural views are created by using 5W1H Re-Doc methodology with UML (package, class, deployment, component, sequence, and use case diagrams) and the functionality of the system fully understood, a new target system is modeled with MVC architecture using the 4 + 1 architectural views. The different functionalities of the system are modularized into classes and methods. The target system's code is then written using the new architectural models, maintaining the same functionality as the legacy system. The unit testing class ensures the functionality of the game as well as identifying any errors in the business model. Input data validation is added to the software game to handle exceptions along with a testing class. Reverse engineering using 4 + 1 views with UML diagrams provides an abstraction that allows for quicker comprehension of the legacy system. Reengineering code into MVC using this level of abstraction helps to prevent confusion of all the different interrelated parts. MVC architecture itself provides an intuitive organizational scheme that helps to structure the code in a meaningful way.