Modernizing Legacy Systems: A Re-Engineering Approach

Article *in* International Journal of Web Portals · July 2018

DOI: 10.4018/IJWP.2018070104

2 authors, including:

Neelanarayanan Venkataraman
VIT University
64 PUBLICATIONS **171** CITATIONS

SEE PROFILE

International Journal of Web Portals
Volume 10 • Issue 2 • July-December 2018

Modernizing Legacy Systems:
A Re-Engineering Approach

Aparna Vijaya, Vellore Institute of Technology, Chennai, India
Neelanarayanan Venkataraman, Vellore Institute of Technology, Chennai, India

ABSTRACT

Cloud computing is a paradigm which has changed the way organizations develop, manage, and deploy their applications. Most of the resources are available at low costs in this technology and it creates new opportunities for organizations to move their existing applications to the cloud for modernization. But this modernization also comes at a price. The authors present a model-driven approach to address the challenges in application modernization and focus on the application migration to cloud. Most of the cloud applications are very specific to a particular vendor's features or services. The proposed methodology addresses the challenge of vendor lock-in also. The authors present their theoretical details and experience with two pilot projects where they have applied the proposed approach.

KEYWORDS

Business Rules, Legacy, Modernization, Reengineering, Software Maintenance

1. INTRODUCTION

Legacy systems can be defined as software systems that have been built years ago by using old technologies and methods but they are still an integral part of business. Many organizations have a lot of legacy systems up and running even today and they find it hard to update and maintain such systems. These systems have high operation cost, do not take advantage of the latest computing environments, and are difficult to maintain and modify. Literatures and even industry claim that (Cornelius, 1989; Ransom, 1998) of the most expensive and critical phase in the life cycle of a software system is its maintenance. According to a recent Standish group survey (Standish, 2010), the maintenance phase can consume upto 80% of the total cost of the development of software systems. A huge amount of effort and time has to be invested to change the legacy software systems in order to keep their core functionalities and business process intact. This process of conversion, rewriting or porting of a legacy software system to modern programming languages or new open computing environments is termed as software modernization. Hence many industry surveys (Standish, 2010) show legacy application modernization as one of the top priorities. The various challenges in maintaining a legacy software system can be summarized as follows:

DOI: 10.4018/IJWP.2018070104

- The source code of the legacy software systems has been written years ago with some obsolete technologies and without sufficient documentations. Hence it would be a tedious task to maintain the code for which there are not much resources and eminent programmers;
- The source code must have had a lot of bug fixes and patches associated which might not have been documented and making even minor modification would become costly in terms of risk;
- Lack of an appropriate method for identifying the business rules from legacy system in order to modify it depending on the market needs;
- Legacy applications are often limited to specific environments or do not provide support for the alignment with newer technologies;
- Economical reasons (cost saving, limited internal support);
- Any other technical reasons (optimum resource utilization, unlimited scalability of resources, less maintainability, accessibility and availability).

2. NEED FOR LEGACY SYSTEM MODERNIZATION

New software technologies are entering the market every day. Even though many of the core business strategies remain the same every year the quick and emerging technology and information trends in market necessitate every organization and their products to stay up-to-date on the latest business strategies offered. Hence there is an increasing need to evolve the existing software assets in order to take advantage of the new technology. But re-developing applications to become compatible with newer technologies becomes difficult due to the time and resources required. So in order to renew a legacy system we need to understand the design of the software at the beginning. Due to incomplete or outdated documentations and reference materials, this can be a tedious and very complicated task.

2.1. Case Study

Nexflix (Izrailevsky, 2016) took around 8 years for migrating their application to cloud. Initially they tried with the easiest way to move to the cloud which is to forklift all of the components without changing and drop them in AWS. But in doing so, they discovered that they ended up moving all the problems and limitations of the legacy application along with it. So, they decided to choose the cloud-native approach by rebuilding all of the technology and changing the way it was developed. They migrated from a monolithic app to hundreds of micro-services in the architectural level, and incorporated NoSQL databases.

2.2. Challenges of Architecting Cloud Applications

According to literature (Ransom, 1998) most of the legacy applications are monolithic or stovepipe in nature. When modernizing such applications to leverage the benefits of underlying cloud platform needs a lot of effort to re-engineer or restructure the legacy code into a layered component-based architecture. Also, there is a need to ensure portability across multiple cloud providers which is challenging task because of the heterogeneity in the semantics provided by them. It requires a deep understanding of the concepts and terminologies offered by various cloud providers and the information required by each provider to deploy and run artifacts related to an application in their environment. It also involves complex decisions that are specified by various stakeholders which includes financial decisions as to which cloud platform have to be chosen to reduce costs and architectural decisions regarding partitioning the application.

3. APPROACHES FOR SOFTWARE MODERNIZATION

Legacy systems are often custom made and have not supported code reuse nor best practices. These applications are also often written with outdated development languages and run on platforms