Software Reuse
CHAPTER 16

Asif Mohaimen
2012331054

April 20, 2016

1 Introduction

Here in this specific chapter of the book the topic which is mainly discussed is the *Software Reusing* & it's related factors.

The key things which are discussed in this chapter are listed below:

- The reuse landscape
- Application frameworks
- Software product lines
- COTS product reuse

1.1 Benefits

There are some key benefits of *software reuse*. They are pointed below:

- **Increased dependability:** Because of testing in real environments it becomes much more reliable than the new software to be coded.
- **Reduced process risk:** It takes the processing and managing risks to a minimal level.
- **Effective use of specialists:** Because of reusing specialists can make their products much more effective to use.
- **Standards compliance:** Familiar interface makes it more convenient for the users.
- **Accelerated development:** Because of the reusing the development gets boosted. Thus, both development and validation time may be reduced.

1.2 Drawbacks

There are some key problems of *software reuse*. They are pointed below:

- **Increased maintenance costs:** Because of *reusing* the maintenance cost goes higher.
- **Lack of tool support:** As some software doesn't provide adequate support for *reusing* the tools support becomes less.
- **Not-invented-here syndrome:** Because of not being their own code engineers don't feel good to reuse the existing software.
- **Creating, maintaining, and using a component library:** Component libraries adaptation process becomes much difficult.
- **Finding, understanding, and adapting reusable components:** Lack of confident people to reuse the additional components of the existing software.

2 The *reuse* landscape

There are many different approaches of *reuse* which are discussed in this section. It covers the range of possible reuse techniques.

2.1 Approaches that support software reuse

- **Architectural patterns:** Standard software architectures that support common types of application systems are used as the basis of applications.
- **Design patterns:** Generic abstractions are represented as design patterns to show abstract & concrete objects and their interactions.
- **Component-based development:** Systems are developed through the integration of components to meet the component-model standards.
- **Application frameworks:** To create application systems abstract & concrete classes are adopted.
- **Legacy system wrapping:** Legacy systems are implemented in this approach.
- **Service-oriented systems:** Systems which are developed by linking shared services can be externally provided.
- **Software product lines:** To provide support for different customers a common architecture is adopted.

- **COTS product reuse:** Systems are developed by the integration & configuration of different systems.
- **ERP systems:** These system encapsulate generic business functionality and rules are configured for an organization.
- **Configurable vertical applications:** These systems are built to meet the needs of different customers.

3 Application frameworks

Another fact about *software reuse* is using frameworks. Application frameworks are abstract objects that are designed for reuse through specialization. We can not directly modify the framework codes. We can just use the features provided by it. To use it we mostly extend the classes or objects or implement the methods provided to us. For example now there are many web frameworks available to us that makes it easy for web developers to make web applications for example Spring,Hibernate,Struts(for Java), Laravel & Codeigniter (for PHP) etc. They usually incorporate good design practice through design patterns.

4 Software product lines

Software product lines are related applications that are developed from a common base. This generic system is adapted to meet specific requirements for functionality, target platform or operational configuration. It allows us to modify the codes beyond any limit. They usually arise from existing systems or applications. Software product lines are designed to be reconfigured and this reconfiguration may involve adding or removing components from the system, defining parameters and constraints for system components.

5 COTS product reuse

In COTS products a variety of customers are supported without changing the source code. These applications usually have a large number of functionality. Most of the open source software are COTS products because they are vividly used by other people without looking at the source code. ERP systems provided by the SAP and BEA are another example of COTS product. Large business institutions use such enterprise application software where they can configure and make their own software by selecting the functionality they want to use provided by the software vendors. But they have some limitations for example they increase dependency over software vendors. Also lack of flexibility is another major concern for such systems because very often it might not include exactly what you want. In that case it would raise complications which may result in raise in cost because ultimately they might have to implement a whole new system.