

# REUSABILITY VS EXTENSIBILITY

PRESENTED BY :  
SHIVANI KAPOOR,  
M.TECH(CSE),  
PUNJABI UNIVERSITY, PATIALA

# **CONTENTS :**

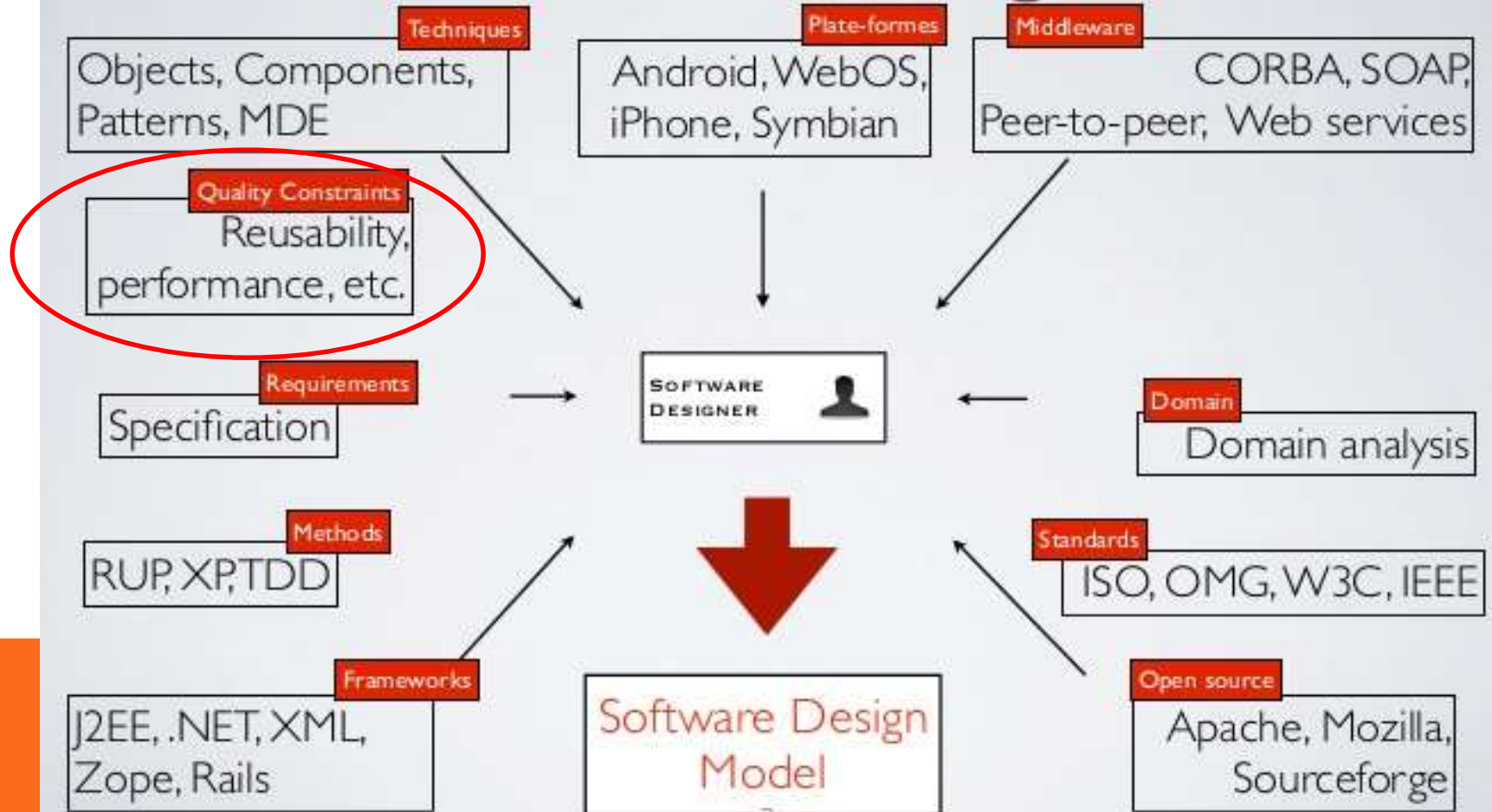
## **1. The Differences between Reusability and Extensibility as follows:**

- Definitions
- Frameworks
- UML
- Classifications

## **2. Similarities**



# Software Design



# 1. DEFINITIONS

# WHAT IS REUSABILITY?

**Reusability** is the use of existing assets in some form within the software product development process. Assets are products and by products of the software development life cycle and include code, software components, test suites, designs and documentation. A chunk of code is regularly organized using modules or namespaces into layers. Objects and software components offer a more advanced form of reusability.



# WHAT IS EXTENSIBILITY?

**Extensibility** is a system design principle where the implementation takes future growth into consideration. It is a systemic measure of the ability to extend a system and the level of effort required to implement the extension. Extensions can be through the addition of new functionality or through modification of existing functionality. The central theme is to provide for change typically enhancements while minimizing impact to existing system functions.



# **REUSABILITY**

1. Reusability should be increased where possible
2. Components should be designed to work on different context.
3. Generalize design as much as possible :
  - Use Frameworks, Patterns, and UML Collaborations.
4. Design the system to contain hooks .
5. Keep the design as simple as possible.

# **REUSE ANALYSIS, DESIGN, AND CODE**


- Reuse existing artifacts when possible, to take advantage of existing investment.
- Use Frameworks, Patterns, and UML Collaborations.






## 2. FRAMEWORKS

# **SOFTWARE FRAMEWORK**

- Represented by its code. , a Framework is
  - A set of classes, abstract classes and interfaces.
  - A set of behaviors, spread over these classes.
  - An incomplete application for a family of products.
  - A set of hooks, where subclasses can insert their specialized behavior.
  - The expectations placed upon the subclasses.
  - A logic decomposition of a problem
- 

# **FRAMEWORK GOALS**

- Reuse: code, design, domain analysis, and documentation.
  - Simplify software development.
  - Reduce code writing.
  - Allow inexperienced designers and programmers to develop good software.
  - Extract the knowledge of experimented designers and programmers
- 

# **REUSABILITY OF FRAMEWORK**

- Reuse of framework components improves developer productivity, as well as software performance, reliability, and interoperability.
- The stable interfaces define generic components that can be extended to create new applications.



# **EXTENSIBILITY OF FRAMEWORKS**

- A framework enhances extensibility by providing explicit hook methods for planned variability.
- Extensibility is essential to ensure rapid customization of new application features..



### 3. UML

# **REUSABILITY IN UML**

We investigate reusability definition, assessment, and analysis for the unified modeling language (UML), focusing on using UML via a tool (e.g., Rational Rose, Together Control Center, etc.) prior to the development of software (code). Thus, this work concentrates on reusing a “design model” and monitoring this reuse as the model is transitioned into software.



# **EXTENSIBILITY IN UML**

- **UML** extensibility features are simple way to store additional information in models
- Infact , they have a significant semantic impact in expressive power of UML by allowing modelers to extend UML with new modeling concepts.
- UML tools match domain components and frameworks .These tools used for rapid development using reusable assets .






## 4. CLASSIFICATIONS

# **CLASSIFICATION OF REUSABILITY MECHANISMS**

**Object composition and inheritance are two techniques for reusing functionality in object-oriented systems :**

*Class inheritance* allows a subclass' implementation to be defined in terms of the parent class' implementation. This type of reuse is often called **white-box reuse**.

*Object composition* is a different method of reusing functionality. Objects are composed to achieve more complex functionality. This approach requires that the objects have well-defined interfaces since the internals of the objects are unknown. Because objects are treated only as "black boxes," this type of reuse is often called **black-box reuse**.



# **CLASSIFICATION OF EXTENSIBILITY MECHANISMS**

## **1. White-Box Extensibility:**

White-box extensibility refers to the ways in which a software system can be extended by modifying or adding to the source code. This is the least restrictive and most flexible form of extensibility. Depending on the way changes are applied, we have to distinguish further between open-box extensibility and glass-box extensibility

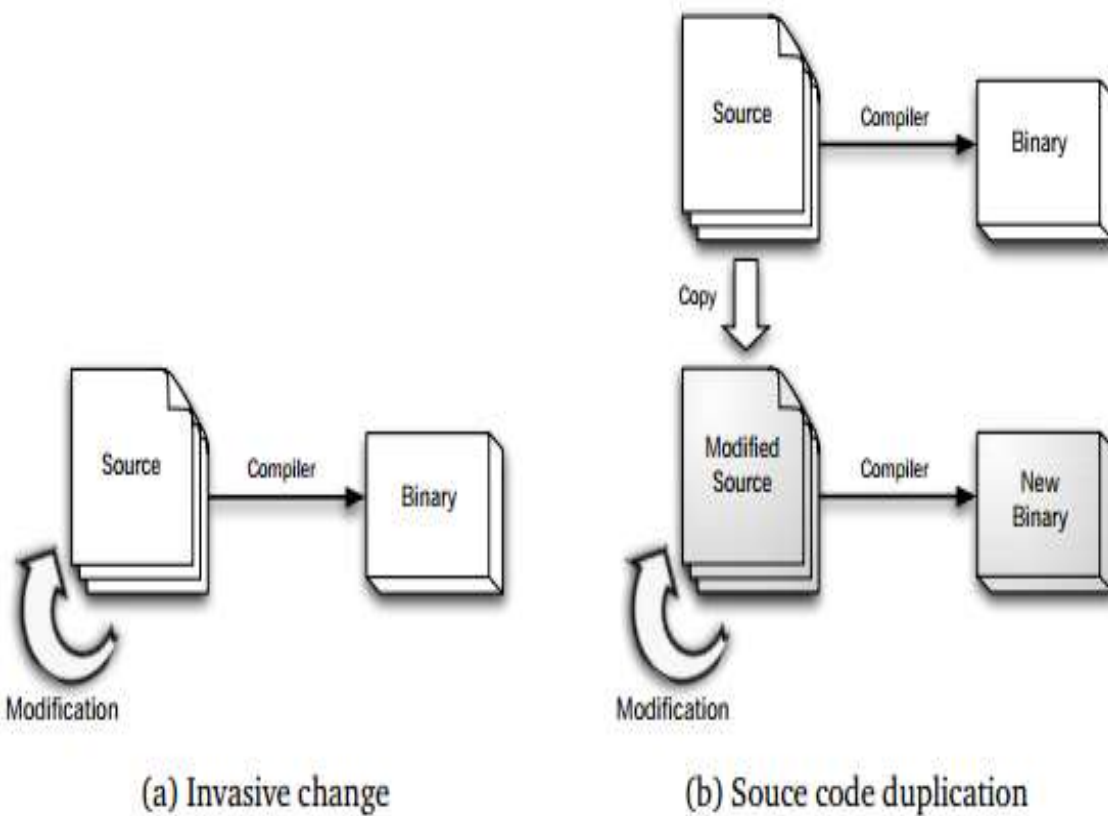


Figure 1.1: Extensibility based on source code reuse.

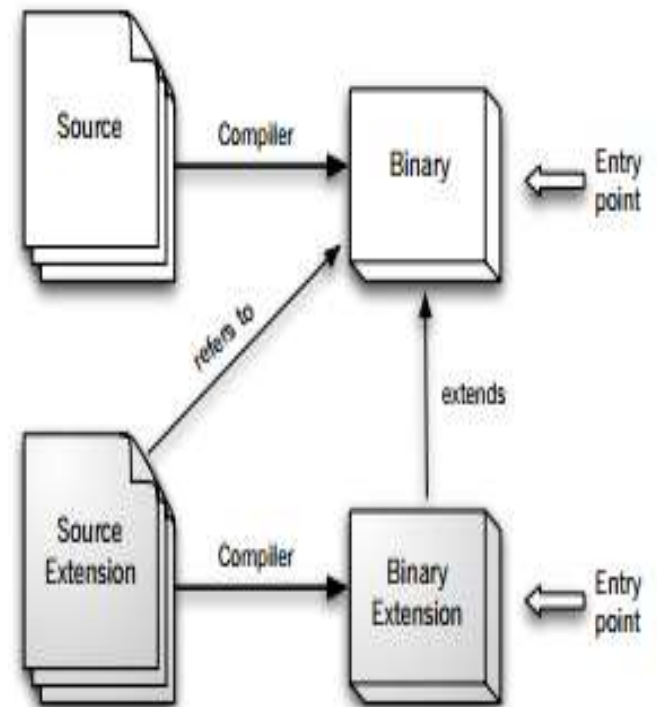


Figure 1.2: Extensibility based on binary reuse.

## 1.1 OPEN-BOX EXTENSIBILITY

## **1.2 Glass-Box Extensibility :**

Glass-box extensibility refers to the ways in which a software system may be copied, when the source code is available, but may not be modified. Programmers that want to extend the system can view the code, but they have to separate their extensions from the original system in a way that does not affect the original system.



## 2. **Black-box extensibility:**


Black-box extensibility refers to the ways in which a software system may be extended when no internal details about a system's architecture and implementation are available. Black-box extensible systems are deployed and extended only by using their interface specification. This approach allows system manufacturers to fully encapsulate their systems and hide all implementation details.



# **SIMILARITIES**

**Extensibility and reusability** have many emphasized properties in common, including low coupling, modularity and high cohesion.

**Software reusability** is boosted by **extensibility** and refers to software elements' ability to construct for many different software systems, which is motivated by the observation of software systems often sharing common elements. **Reusability together with extensibility** allows a technology to be transferred to another project with less development and maintenance time, as well as enhanced reliability and consistency



thank  
you

