Software Reuse: A Systematic review
Deepshikha Jamwal
Research Scholar
Jamwal.shivani@gmail.com
Department of Computer Science & IT
University of Jammu

ABSTRACT
Reuse-based software technology is processes of designing software for the reuse purpose, which can great, reduce the time and expenses of developing and can enhance the flexibility, maintainability and reliability of the software. Software reuse processes have been under continuous attention in the software engineering and software reuse research communities during past years. Although several processes have been investigated to develop reusable software, there are not available studies that compare them. In this way, this paper presents a detailed survey on software reuse processes, challenges and some problems, which comes while software reusing. Also author has given some tips to improve reuse. As Systematic software reuse is the most effective way to significantly improve software development.

KEY WORDS
Software factories, software productivity, software reuse.

OBJECTIVES
• To explain the benefits of software reuse and some reuse problems.
• To describe different types of reusable component and processes for reuse.
• To find challenges and problems while software reusing.

1. INTRODUCTION

The software-reuse is a process of reusing “the software designed for reuse purpose”. Software-reuse is the solution that avoids a repeated labor in the software development and can make use of the knowledge and experience getting from the past software development and concentrates the especial part of application. It’s aim is making the software development from zero no longer.

Software reuse is the process of creating software systems from existing software rather than building software systems from scratch. Anything that is produced from a software development effort can potentially be reused. Software reuse is the process of implementing or updating software systems using existing software assets.

A good software reuse process facilitates the increase of productivity, quality, and reliability, and the decrease of costs and implementation time. An initial investment is required to start a software reuse process, but that investment pays for itself in a few reuses [1].

1.1 THE SCOPE OF SOFTWARE REUSE
Software-reuse can be divided into *product reuse* and *process reuse* according to the reuse object. Software-reuse can be divided into product reuse and process reuse according to the reuse object. The product reuse means the reuse of software component, getting a new system from component integration and construction. The process reuse means the reuse of past software development process, automatically or half-automatically producing the system using the reuse generator. The process reuse depends on the applicable to some special applied domain currently, but the product reuse is a realistic and essential path now.

### 1.2 THE MODE OF SOFTWARE-REUSE

The software-reuse can be divided into *black box reuse* and *white box reuse* according to the reuse mode of reuse information. The black box reuse means to use the component directly without modification. This is the ideal way. The white box reuse means that the component can’t meet the customer need completely also and cannot be used until be properly modified according to the customer need. But in the mostly applied construction process, the adaptability modification of the component is essential [2].

### 1.3 THE PROCESS OF SOFTWARE-REUSE

**a. Domain Analysis Phase.**
This phase is to come certain whether deserve to reuse the infrastructure for the domain development mainly through the definition and analysis of application domain.

**b. Domain Engineering Phase.**
This phase is to acquire general system structure according to the domain commonness getting from domain analysis phase and regulate how the property match the system structure and how to bind variable point.

**c. Property Obtaining Phase.**
This phase includes development of reuse, may also includes some exterior adopt of reuse property.

**d. Property Categorizing Phase.**
This mission is a database management mission actually, including categorizing and saving reuse property.

**e. Property Maintaining Phase.**
This mission is a maintenance mission actually, and to combine configuring management and edition control.

### 1.4 REUSE PLANNING FACTOR

- The development schedules for the software.
- The expected software lifetime.
- The background, skills and experience of the development team.
- The criticality of the software and its non-functional requirements.
- The application domains.
- The execution platform for the software.

### 1.5 TYPES OF REUSE

**Horizontal Reuse**
Horizontal reuse refers to software components used across a wide variety of applications. Horizontal reuse can also refer to the use of a commercial off-the-shelf (COTS) or third-party application within a larger system, such as an e-mail package or a word processing program. A variety of software libraries and repositories containing this type of code and documentation exist today at various locations on the Internet.

**Vertical Reuse**

Vertical reuse, significantly untapped by the software community at large, but potentially very useful, has far reaching implications for current and future software development efforts. The basic idea is the reuse of system functional areas, or domains, which can be used by a family of systems with similar functionality [3].

1.6 **REUSE COSTS - THE INVESTMENT**

There is no denying the large cost associated with starting a reuse program. It is an extra cost on top of the traditional development costs, since designing reusable assets takes more time and care than designing a one-time specific system. The upfront investment spans organizational, technical, and process changes, as well as the cost of tools to support those changes, and the cost of training people on the new tools and changes.

2.1 **PROCESS**

The software development process must be enhanced to include reuse activities. A reuse library or repository must be created and maintained, and tools must be acquired or developed to access the assets, and many new procedures must be specified:

- Procedures for developing reusable assets and inclusion of assets in the repository.
- Procedures for domain analysis and architecture design and modification.
- Procedures for configuration management and control of reusable assets.

**SOFTWARE REUSE PROCESSES**

For defining processes for specific software development tasks should be a familiar one. A well-defined process can be observed and measured, and thus improved. Moreover, the adoption of processes also allows the dissemination of effective work practices to occur more quickly than the building up of personal experience. An emphasis on process helps software development to become more like engineering, with predictable time and effort constraints, and less like art. Besides the issues related to non-technical aspects, a software reuse process must also describe two essential activities: the development for reuse and the development with reuse. In the literature, several researchers works that study efficient ways to develop reusable software can be found. These works focus on two directions: domain engineering and, currently, product lines, as can be seen in the next sections [4].

2.2 **ANALYSIS AND SOFTWARE ARCHITECTURE DESIGN**
To implement a product line approach, a group of domain experts must be established and maintained to perform domain analysis and develop architectures for the domain. In their analysis, this group must partition the domain into segments that can be developed independently and can evolve for future the people. If the people in the organization do not understand the concepts behind reuse, and do not see the benefits, reuse won't happen. Since software reuse is not a common standard, staff training and subsequent buy-in must be accomplished for a reuse effort to succeed.

3. REUSE ADVANTAGES

With all the costs and prerequisites outlined above, software reuse may seem like more effort than it is worth. Higher quality products are produced due to repeated use and test, and intentional design for robustness and reuse. Each successive use of a given software asset will retest it, and the more tests performed, the more likely defects will be found and corrected. Every successful reuse of an asset increases its reliability level, increases its usefulness in the reuse repository, and decreases the risk of failure.

- Increased Dependability

Reused software, that has been tried and tested in working systems, should be more dependable on new software.

- Reused Process Risk

If software exists there is less uncertainty in the costs of reusing that software than in the costs of development. This is an important factor for project management as it reduces the margin of errors in project cost estimation.

3.1 NECESSARY TOOLS FOR CHANGE

Another key for successful reuse is the organization and accessibility of the common reusable assets. Asset management tools, such as repositories, for architectures, designs, documentation, and code must be developed and maintained. Also needed are tools to aid in the integration of architecture, design, and software products, in order to speed prototyping, full-scale development, modifications, and maintenance.

3.2 PEOPLE - TRAINING AND REWARDS

By far the Domain most important part of the reuse process is

- EFFECTIVE USE OF SPECIALIST

Instead of application specialists doing the same work on different projects, these specialist can develop reusable software that encapsulate their knowledge.

- STANDARD COMPLIANCE

Some standard, such as user interface standards, can be implemented as a set of standard reusable component.
3.3 REUSE PROBLEMS

Creating and maintaining a component library: populating

- INCREASED MAINTENANCE COSTS

If the source code of a reused software system or components is available then maintenance cost, may be increased as the reused elements of the system may become increasingly incompatible with system changes.

- LACK OF TOOL SUPPORT

It may be difficult or impossible to integrate the tools like CASE tools with component library systems.

- “NOT INVENTED HERE” SYNDROME

The fact that writing original software is seen as more challenging than reusing other peoples software.

- MAINTAINING A COMPONENT LIBRARY

Populating a reusable component library for classifying, retrieving software components and cataloguing the software developer use this library, which is very expensive.

- FINDING AND ADAPTING REUSABLE COMPONENT

Software components have to be discover in a library, understood as sometime adapted to work in a new environment [5].

4. CHALLENGES IN SOFTWARE REUSE

Software reuse is not always successfully by organizations. There are certain challenges should be overcome.

TYPES OF CHALLENGES IN SOFTWARE REUSE.

Typical challenges faced when moving to software reuse include:
1. Technical concerns on software development including design for multiple uses, managing versions, dependencies, configuration and reuse without source.
2. Organizational aspects:
   o Organization structure enabling communication of developer needs and business requirements especially in a large organization.
3. Cultural aspects:
   o Economic investment needs taxation or charge-back schemes to found reuse groups.
   o Administrative problems how to catalog archive, and retrieve reusable assets across multiple business units within large organizations.

5. REUSE PROBLEMS

   o **Increased maintenance cost**
     - External changes to a reused component may force changes to the reusing software.
     - Required changes to a reused component may be more difficult to have than for a custom component.
   o **Lack of tool support**
     - There is little support for maintaining a catalog of reusable components and for finding components.
     - Development tools do not support development with all kinds of reusable components well.
   o **Not-invented-here syndrome**
     - Many software engineers tend to prefer rewriting a component to reusing it.
   o **The reuse barrier**
     - A software developer will only search for a reusable component if s/he expects to find an appropriate one.
     - S/he will only find it is described in a way s/he can recognize as providing a solution.
     - S/he will only use it if s/he can understand it.
     - Use will only be successful if the component can be adapted to the problem at hand.
     - The whole process will start only if the developer expects all steps to be successful in the beginning.
   o Reuse works well for many general-purpose components, but is difficult for highly domain-specific ones [6].

6. HOW TO IMPROVE REUSE

   To improve systematic reuse concentrate on:
   1. Education about reuse
   2. Developers understanding on the economic feasibility of reuse.
   3. Introduce a common development process that promotes reuse.
   4. Make high quality assets available to developers.

7. SUMMARY

   Adopting a software reuse process is an effective way for organizations to obtain the benefits related to software reuse, such as quality improvements and time-to-market reduction. However, the choice of a reuse process is not a trivial task, even considering the division between domain engineering and product lines. In this paper the author work on challenges and problems that comes while software reusing and ultimately have describes some tips to improve reuse. This work may serve as a guide to aid
organizations in the adoption of a reuse process, or as a basis for a formation of a reuse process.

8. FUTURE WORK

The future work of the author is to work on two main challenges to make effective reuse i.e 1. Technological and 2. Organizational. Also to overcome the major challenge with software reuse is that of introducing framework and method into a company is the future work of the author. Also the author is working on the software reuse process formalization, in the domain analysis step, defining methods for domain scoping and feature modeling.

References