Enhancing Software Reusability through Value Based Software Repository

Muhammad Zahid Khan and M. N. A. Khan
Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, Islamabad, Pakistan
khanzahidkhan@hotmail.com, mnak2010@gmail.com

Abstract

Profitability is prime goal of every software organization. In the recent years, software development paradigm has vastly changed due to rapidly changing business requirements. Now customers demand for their desired products to be delivered in a minimal timeframe. To cater for these demands, different rapid development techniques termed as agile development and reusability phenomena are introduced in the software industry. With the help of reusable components, developing and deploying software products can be much easier and economical. Such approaches can help save cost and working hours so that the developing company can utilize its scarce resources on other projects. But achieving accurate and precise reusability is not a piece of cake because along with economical and time saving benefits it comes with some shortcomings, such as it can increase the system complexity, can harm the architecture, degrade quality of the system etc. A proper storage facility of reusable components is required where an organization can save its important data and retrieve it when required. A wide range of standard repositories are used in various organizations for this purpose where important data is stored and retrieved. But in reality, every repository contains huge volumes of data and mostly no proper business standards of storing mechanism are applied. Though there are some repository maintaining software available in the market, but they only provide mechanism for storing artifacts. Maintaining the huge data volumes necessitates providing ample search assistance and putting in place precise reusable components suggestion procedure; therefore, researchers have to sift through loads of volumes of data within the repository for locating the intended component. In this paper, we discuss a novel concept of Central Value Based Software Repository (CBVSR). The proposed approach can assist users in finding the more appropriate artifact as it provides the best possible match against the user query. Proper standards have been followed in storing, deleting, amending and discarding data from the CBVSR. Data quality, security and integrity have been taken care of and indexing, tagging and classification are performed to provide technical assistance to developers.

Keywords: Central Value Based Repository, Software Reusability, Value Based Artifacts, Ranking and Keyword Tagging

1. Introduction

A software repository stores various software components or artifacts for future reuse. It is placed over a local or global network depending upon organizational requirements. A user can access the software repository across the network directly or indirectly without physically travelling to its location [1]. Software repositories are intended to be malware free and meant to store reusable and valuable artifacts. The main aim of knowledge management is to make the best use of an organization’s
intellectual and critical data [4]. When this concept is used in the field of software development, it entails maintaining all the pertinent information related to various software components/artifacts. This knowledge can only be useful when it is well structured, stored and reproducible. Also, it should be available to all the concerned employees of the organization [4].

A value-based repository can respond to different queries and provide intended object, code, model, test case, dataset, artifacts, table or module etc. according to the stakeholders’ value propositions. There are different stakeholders’ value propositions, such as:

a. Values related to different economic aspects like cheaper development costs, greater efficiency, and lower deployment budgets.

b. Aspects related to various psychological properties like attractive designs, human friendly models etc.

c. Various social values like status, image, reputation etc.

d. Different factors related to health and security.

Software companies develop products to capture greater market share by attracting targeting customers or stakeholders. The whole software development process has been changed with the introduction of agile techniques. Now the developers aim at delivering the required products within the minimal timeframe. Due to this, the reusability of different software components is ideal for such kind of situations. But reusability has certain limitations. For example, wrong integration of reusable code can harm the whole system architecture. It can also increase complexity of defect detection and their removal during the testing phase. Injecting an unsuitable component into software design can also collapse the whole system and so on. The above mentioned problems in software reusability can cost organizations a lot of economy and time. To cater for these issues, a Value Based Software Repository is required which can store different software components; classify them in to different categories; index them; and assign different ranks to them to make the retrieval process of intended reuse item fast and easy. Value based repository can contribute key role in the field of software engineering to make reusability more precise and accurate.

2. Literature Review

Software reuse is a process to implement or update the software system by using pre-existing software components [5]. Well defined organized software reuse method is very important for survival of any organization because due to this the overall production of any company can take a boom. By the virtue of employing reusability, quality of the products increases and development cost along with the product delivery time is minimized [6, 7]. Through software reuse, organizations can save their resources in term of human efforts and eliminate schedule constraints. By reproducing same items again and again enable organizations to maintain quality among the reproducing items and also the known risks are dealt with a great sense of responsibility [7]. Incremental development was termed by some researchers as software reuse because latest versions of software, component or products are based upon older versions.

COTS usage is increasing due to the changing software development processes and rapidly growing economic conditions to cope with the recent situations and to enhance productivity. In conventional software process models, these components should be easily accessible to the development team of the organization in order to save organization time and capital. A constant change is occurring in the processes which are altering the convention of creating an application from company point of view towards
development of products from customers’ point of view by embedding different codes for functionality enhancements through component reusability. Considerable amount of user requirements can be provided by COTS with minimal development time and limited cost [8]. From reusability prospective, software inconsistency, unusual behavior, operational failure, service suspension, minor system faults are known as risks. The risk factor if not handled properly can harm the system architecture and degrade quality of service. If managed amicably, it can save money and other resource for organization.

Software repository provides a better management approach to reusability by containing majority of software codes, scripts, module and other artifacts which can be used for reusable purposes. Each of the scientific and computer based repositories are built for some specific job [9]. Similarly, for open source software, different services and tools such as “SourceForge” [10] provide a platform for the development and distribution of multifarious software. On the other hand, Eclipse [11] provides an open platform for various integration tools based upon variety of plug-ins. The constructions of intelligent information repositories for software components (executable, forms, modules, tools, techniques etc.,) which can be separated from each other and stored and reuse accordingly has been a major issue for quite a long time [12]. The vision of developers is almost necessary to evaluate usefulness of pre-existing methods and techniques [2]. In distributed architecture, subjective information is used for selection of relevant development software methods and automated tools. Due to the unavailability of systematically collected and structured data, it is really hard to evaluate the extent that certain tool, method and practice deals with current software development problems [2].

Enterprise Architecture Framework of an organization is built on models that evolve over the period of certain time. Integrated models in every organization are considered as more crucial as they should be managed very carefully. The Enterprise Architecture Framework can be considered as a mechanism for managing any organization integrated system assets. To develop an effective system which properly manages the system assets, developers and architects of the system need those techniques and methods which support reuse/impact/gap and redundancy analysis [3].


3. Existing Repository Systems in Practice

Repository is the basic necessity of any software organization, where most important and reusable data, classes, scripts, design documents and various software components are stored to be used in future. Given below is a standard repository structure.
3.1. Description of the Existing Repository Systems

A brief description of components and their interactions is shown in Figure 1 is given below:

- The oval shape icons in the figure represent different groups of people within and outside the organization e.g., Client/End-users indicates various clients and CEO/Project Manager indicate governing bodies of an organization.
- Requirement gathering team represents the requirement elicitation group.
- Development team represents the different development personnel from various development departments.
- The cylinder like shape shows the standard repository of any organization.
- Arrows indicate the interaction way which is two sided in most of the cases.
- As there is no check and balance, so all the users/developers or stakeholders communicate with the repository in two way fashions i.e., they can store and retrieve a document from the repository at the same time.

3.2. Drawbacks of the Existing System

The current repository system widely used in the organizations bears several shortcomings. Some of them are mentioned below:

- Since there is no standard procedure of storing data in a repository so one can create a folder of own choice. In some cases, developers create folders with their names instead of the content information residing in that folder. Therefore, other developer cannot gain meaningful information by seeing that folder which increases the scale of search efforts and complexity of the system.
- In a software organization, there are different developments teams working on different projects by using various programming and scripting languages; and all of them save their important components in a shared repository which makes the repository storage more and more complex as no standard repository classification technique is available.
c. Systematic software reuse cannot be applied in this scenario, so reuse creates problems instead of solving them.
d. Management or audit system of people accessing the repository does not exist and as a result, repository mostly contains unnecessary data which degrades the overall system efficiency.
e. Other bureaus of the company located worldwide or in other cities of the country cannot access the repository because they are only locally accessible.
f. There is no content or document management policy to classify and store the items having similar characteristics in one location.
g. In the absence of proper access management or audit system, integrity of data residing in the repository cannot be achieved and numerous data redundancy issues occur.
h. No mechanism of differentiation among old and newly stored software artifacts is defined which makes finding the newly developed items more hectic and time consuming.
i. There is no support of legacy system components.
j. In the absence of systematic content removal process, the repository becomes overloaded with junk stuff after few years.
k. There is no proper automated ranking method present in the existing repository system to rank the most important artifacts, so a searcher cannot easily find the latest or closely matched component.
l. Absence of indexing of the components residing in the repository makes tracking of the software components difficult.
m. A proper tagging mechanism which helps classification of similar objects into one category is missing.

4. Proposed System Architecture for Value-based Repository

A proposed framework (Figure 2) for value based repository is based upon the following characteristics:

a. A globally accessible value based “central repository” for storing different software artifacts (such as code, scripts, design documents, SQL queries etc.,) is proposed for pertinent reusability measures.
b. An “access management/audit system” is included in the proposed framework through which users will interact. Users would have only the read access and team leaders and executives of every department would have the write access.
c. A proper “business standard” will be followed for storing reusable and other important data in the value-based repository. Every data must have to be approved by the administrator of the auditory system of that department. As there are various departments in each organization, so every department will have its own audit administrator who will store or discard any item from the central repository.
d. “Classification standard” is also incorporated into the value-based repository which will classify alike items to one category so a user can search and get its desired item in a minimal timeframe.
e. Proper “Tagging mechanism” will be applied for each software artifact for quick access of that software component in future. Tagging is performed by the concerned developer, designer or architect of that particular software artifact and once it is verified by the team lead, it would be stored in the value-based software repository.
f. Value-based repository will maintain an “index” of all the stored items for search optimization and quick retrieval.
g. “Systematic ranking” will be done for each software component which will be based on importance and priority of that specific component. Value-based repository would assign the newly developed item a higher rank.

h. Value-based repository will have a directory of “customized reusable artifacts” which will contain generic scripts, codes or modules. This would help improve quality of reuse items.

i. A systematic “searching and reporting” procedure has been implemented to report user about the availability of reuse artifacts and other important stuff.

j. All outfits of the company will be connected to the central repository through secured communication lines.

k. Developers, testers, designers and architects can add reusable components into the central repository after the approval of their local audit system.

l. The regional heads or CEOs will manage the local repositories and would have the right to delete or modify their own regional data only.

m. If any modification or deletion is required, the concerned person will generate a change request to the administrator of the audit system which will be entertained accordingly.

n. A support for legacy software artifacts will be provided in value-based repository in case of modification of any legacy system.

o. Software components older than five years will be automatically moved to the “Legacy Components” folder.

p. Legacy components folder will be sub-divided into various sub-categories of design document, techniques, various codes and other reusable and important contents.

q. With the help of access or audit management procedures, data integrity of all software artifacts residing in the central value-based repository will be taken care of to avoid any redundancy issues.

Figure 2. Proposed System Containing Value-Based Central Repository
5. Description of the Components of Value-based Central Repository

Given below are the components of Value-based Central Repository along with their interactions as mentioned in Figure 2.

a. The smiley face represents groups of clients or customers outside the organization who will interact with the organization for their specific business needs to fulfill their job assignments.

b. The oval shapes in the figure indicate various groups of people working within the organization like CEO/Project Manager indicate the governing bodies of an organization.

c. Requirements capturing team in the figure represent the requirement elicitation group which will interact with the user and capture requirements with the help of different requirements elicitation methods and techniques.

d. Development team corresponds to development personnel, designers, architects, testers etc. within the organization.

e. The cylindrical shape shows the Central Value-Based Repository of the organization.

f. The rectangle shape shows users, developers and architects etc., who are located outside the organization i.e., across the country or continent.

g. The round rectangle icons show sub-processes which are performed as a result of user activity with the system.

h. The square shapes indicate the audit system/access procedure to the central repository. This system will make sure that data is added into the repository correctly and is well referenced so retrieval should be easy and systematic.

i. Arrow shape indicates the communication mechanism which varies from person to person according to their authorization.

j. Technical persons from within or outside the organization will only have read rights during accessing the central repository i.e., they only search/read or copy the software component but cannot modify or delete it from the central repository.

k. Administrator of the audit system and regional heads of the organization will have the write permissions only for their own regional data i.e., they can modify or delete specific software artifacts which are in their own regional jurisdiction not the data of some other region.

l. A searching and report generation process will exists in the system which will inform the CEOs or project manager about the number of reusable components so they can communicate with the clients about the product delivery and make decisions about the project.

m. Within the Value-based Central repository, tagging will be done for each and every software artifact or component. This will help in the classification of components having similar properties or attributes.

n. Ranking will be performed for each and every software component. Ranking will be based on the “most recent one” i.e., newly stored components in the repository will have high rank as compared to the earlier stored components.

o. Value-based repository will have a directory of “Customized Reusable Artifacts” which will contain generic scripts, codes or modules.

p. Requirement gathering or eliciting team will send a copy of final requirements document to the value-based central repository which will be added into the repository after the approval of Access management or Audit system.

q. The “Legacy Component” support directory will store all the software artifacts supported by the legacy systems. Legacy components support directory will be sub-
categorized into different directories according to various third party tools, codes, test cases, design documents etc.,

r. A software artifact which has been in the repository for more than five years will be moved to legacy component support directory and will be stored in its relevant category.

s. Those software components within legacy components directory whose residing time will exceed the limit of 20 years will be discarded from the legacy components support directory and also from the central repository to avoid any unnecessary storage burden.

t. Security is the core requirement of every system as every software company keeps its business secrets out of the reach of its rival organizations and other intruders so every user will be provided its unique ID and password to keep the company assets safe from unauthorized persons.

6. A Case Study based Validation

Repositories are widely used in every organization. A repository provides different type of services to their clients or stakeholders. In this research, we define case study of a multinational company. Details of its repository structure are given below.

6.1 Repository Structure

The repository of the company resides on a simple FTP server. It structure is given below:

- End user requirements specification documents
- A lot of entity relationship diagrams (ERDs)
- Various user request forms (URFs)
- Commercial-off-the-shelf (COTS) component
- Various data flow diagrams (DFDs)
- Development codes, SQL procedures, scripts and java libraries etc.
- A number of folders created by different departments

We found some major shortcomings in the existing repository which include:

- Finance personnel experience various delay issues in data retrieval from repository during their financial analysis. Because there are huge chunks of data residing the repository so finding the latest one is not that easy as no standard procedure is implemented in data storing. This cost the organization in terms of resources and employee working hours.

- R&D department face the same problem as their three researchers left the organization last month and it took them four working hours to search their ex-employees’ findings about the ongoing project.

- Human resource department experience other type of issues as there is no data integrity policy applied in the repository, majority of their employees complain that HR provides them incorrect and outdated letters which shows absence of regular updating procedure. HR personnel point of view was that someone had deleted the updated information.

- During the annual audit, production department faces difficulties in finding relevant documents as their personnel put everything in the repository without filtering the information and marking/tagging the important data.

- The problem with the marketing and sales department is that majority of their time is spent on finding progress of other department to chalk out future planning.
The people who are mostly affected by this repository structure are programmers, developers, designers and architects of the organization. Due to rapidly changing business requirements, introduction of new services and policies of the organization, these people work very hard to do the job within the planned timeframe. To cope with these, reusability of different software components is applied which is a hectic process for them as data in the repository is unstructured plus there are majority of programmers in the department and most of them do not want to share their code with others.

Agreements and other legal purchasing document units are stored at one place without proper versioning criteria which also creates complication for the users as well as for the repository.

There is not any repository management system in place.

Absence of auto components discard mechanism leads to excess usage of the repository with a lot of junk data.

Other bureau of the organization located at different cities cannot access the repository as no central repository structure exists.

Every user is granted with full read/write access. The main drawback of this policy is that an employee can delete or modify others data. A proper right management system needs to be put in place to grant the necessary privileges to the users.

No indexing procedure is adopted to index the data according to its importance. As a result, the repository looks like a mess as huge number of folders of different names and number exist in the repository.

Proper search technique is not imbedded or introduced in this repository structure, so a user spends a lot of time to search the required artifact within the repository.

6.2. Implementing Central Value-based Repository through Rule-based Validation

To enhance performance of the existing repository, we modified its architecture and introduced a set of rules. We implemented the value-based repository concept which produced promising results in searching, management, classification, better reusability suggestions and improvement in the overall system efficiency. The set of rules that we applied include:

a. Classification: The reusable documents, codes, scripts and other component are firstly classified before storing into the repository. We classify them into distinct categories based on document type and functional capabilities. Two main categories used in this work are: Documents (with sub-categories Important Official Correspondence, Technical Reports, RFPs, MoUs, etc.), software artifacts (with sub-categories: code, script, design documents, architectural patterns, SQL queries etc.).

b. Customization: Developers were asked to save all generic codes in the customization directory of value-based repository, so that we get all software component having generic properties in this folder.

c. Tagging: Each user was asked to enter appropriate keywords for each item/artifact that he/she wanted to save in the repository. The keyword tagging helps other users to quickly find that specific component easily for reusability purpose.

d. Indexing: A keyword indexing was performed for search optimization for the reusable components stored in the repository.
e. **Ranking**: Proper ranking was performed and priority levels were set as (high, low, medium) depending upon the value (or worth) of the reusable component. The ranking rule was applied for the recent stored artifacts.

f. **Legacy Support**: Software components whose repository age exceeded five years would be retained in legacy support directory. This attribute helps sift the recent or latest components. The searched results for the reusable components are sorted both on their legacy support as well as the ranking.

g. **Aging**: The reusable components which attain the timeframe of twenty years are automatically discarded from the repository on the pretext of being too old to be reused.

h. **Audit**: Each user prior to saving the artifacts into the repository identifies appropriate keywords which better explain the contents available in the artifact. However, administrator level verification is performed and only verified artifacts are allowed to enter the repository.

i. **Access Management**: Users can only create search and copy items from the central repository whereas deletion, modification, updation lies with administrators of the system and executives of the organization.

j. **Web-based Access**: A proper web-based access mechanism was provided so that users from other branches of the same company can access the repository.

### 6.3. Transformation of Existing System into Rule-based System

We applied our proposed framework of reusability of artifacts being managed by the multinational company. The software artifacts managed by the existing system were of different types which include verity of documents, technical reports, software codes, database queries, script etc., the existing component handling methods of the company had majority of flaws as already discussed at length in Section 4.2. Given below is the list of major flaws that we took care of while transforming them into our proposed framework.

a. No tagging mechanism was present in the existing system.

b. Artifacts were just labeled and stored in the repository and no classification rule was applied.

c. No proper search or retrieval method was provided for the users.

d. System was not user friendly as there was no user interface present and users have to manually search of the reusable content.

e. Proper indexing of the components was lacking.

f. No web-based access was provided; therefore, users from other branches of the company were unable to search the components.

g. Priority of reusable items was not assigned.

h. Access or rights management system was missing.

i. Verification of the importance of the reusable component was not catered for.

To transform the existing system into rule-based as defined in our proposed framework (Section 5), we took the followings steps to overcome deficiencies of the existing system:

a. We classified software artifacts into two main categories and corresponding sub-categories. This information along with other attributes was stored into a database table in structured format.

b. A web-based interface was developed to access the repository so that users from other branches of the same company can access the repository.
We introduced a proper tagging mechanism for each artifact before saving it to the repository. The tagging involved identification of the other index terms or keyword through which the artifact can be searched by the users.

A verification system was introduced which enabled checking the appropriate keywords of reusable artifact by the administrator. Only the verified artifacts were allowed to be stored into the repository.

Keyword indexing was performed on the database table for search optimization.

An interface was created to provide proper search facility to users. The interface supports advanced searching features for performance enhancement.

Importance/worth of each reusable component was assessed by assigning appropriate priority to each reusable artifact.

Legacy support was introduced to sift all legacy artifacts.

Component aging mechanism was introduced to take care of repository volume maintenance. An item exceeding the age limit is automatically discarded from the repository once it becomes more than twenty years old. Aging mechanism was applied while storing items into the repository.

Access management procedure was introduced to grant privileges to the users as per their intended usage of the artifacts.

The attributes stored into a database table for reusability of the artifacts is provided in Table I below.

### Table I. Description of Attributes of Rule-based System

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
</tr>
<tr>
<td>2</td>
<td>Main Category (1. Documents, 2. Software Artifacts)</td>
</tr>
<tr>
<td>3</td>
<td>Sub-Category (1a. RFPs, 1b. URFs, 1c. DFDs, 1d. ERDs, 1e. Design documents etc.)</td>
</tr>
<tr>
<td>4</td>
<td>Location/Path</td>
</tr>
<tr>
<td>5</td>
<td>Date of Creation</td>
</tr>
<tr>
<td>6</td>
<td>Priority (Low, Medium, High)</td>
</tr>
<tr>
<td>7</td>
<td>Index Terms</td>
</tr>
</tbody>
</table>

Description of the attributes defined in Table I is as under.

**ID** indicates the reusable item number saved in the database.

**Category** corresponds to the relevant artifact category. There are two main categories: documents and software artifacts. Numeric numbers are used for each category.

**Sub-category** provides further details about type and functionality of the artifact. Alphanumeric numbers are used for each sub-category.

**Path** relates to physical path of the artifact in the repository.

**Date** stores enrolment date of the reusable component into the repository.

**Priority** pertains to importance of the reusable artifact.

**Index Term** contains appropriate keyword representing domain information of the artifact.

Implementation of rule-based system for the existing component handling system produced some promising results as mentioned below:
a. Organization of the reusable artifacts according to their priority.

b. Transformation of data into structured form.

c. Protection against unauthorized deletion or saving reusable artifacts eliminated chances of redundancy and inconsistency.

d. Categorization of reusable artifacts helped enhancing the development process.

e. Web-based access to the repository enabled remotely working employees to share the reusable artifacts.

f. Keyword indexing made the search process easier and faster.

g. Considerable increase in system efficiency was observed.

7. Conclusion

In this paper, we have introduced a novel framework for storing the reusable software artifacts including important documents through a value-based central repository. The value-based repository helps manage the contents in an efficient manner. It facilitates quick and efficient retrieval of the stored components and helps improve overall efficiency of the organization. We have introduced the concept of tagging and prioritizing the artifacts so that users can search their desired component in a shorter span of time. The repository also assigns a weight to the components to reflect its importance. We validated efficacy of our framework through enrolment of software artifacts of a multinational company. The results due to the implementation of our proposed framework were promising and helped the company in better managing its reusable software components.

References


