Making a Single Repository: A Reusable Component Selection Technique (MSR)

Developed by

Younas Wahab (76-FAS/MSSE/F05)

Supervised by

Dr. Naveed Ikram

Department of Computer Science
Faculty of Basic and Applied Sciences
International Islamic University, Islamabad.

(2008)

International Islamic University, Islamabad
Faculty of Basic & Applied Sciences
Department of Computer Science

Dated: *July* 05, 2008

FINAL APPROVAL

It is certified that we have read the thesis, titled **"Making a Single Repository: A Reusable Component Selection Technique (MSR)"**, submitted by Younas Wahab (Reg. No. 76-FAS/MSSE/F05). It is our judgment that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University Islamabad for MS Degree in Software Engineering.

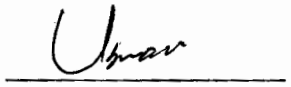
Project Evaluation Committee

External Examiner:

Dr. Iftikhar Azim Niaz,
Head of Faculty of Computing,
Ripah Institute of Informatics,
Islamabad, Pakistan.

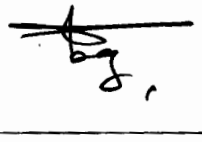
Internal Examiner:

Mr. Muhammad Usman,
Lecturer, Department of Computer Science,
Faculty of Basic and Applied Sciences,
International Islamic University,
Islamabad, Pakistan.



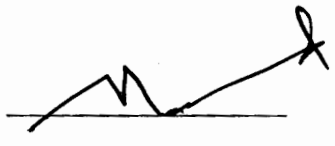
Internal Examiner:

Mr. Shahbaz Ahmad
Lecturer, Department of Computer Science,
Faculty of Basic and Applied Sciences,
International Islamic University,
Islamabad, Pakistan.



Supervisor:

Dr. Naveed Ikram
Associate Professor,
International Islamic University Islamabad (IIUI),
Islamabad, Pakistan.



In the Name

of

ALLAH

The Most Merciful

The Most Beneficent

A thesis submitted in partial fulfillment of the requirements for the degree of

MS in Software Engineering

in the Faculty of Basic and Applied Sciences,

International Islamic University, Islamabad, Pakistan

Project In Brief

Project Title: Making a Single Repository: A Reusable Component Selection Technique (MSR)

Organization: International Islamic University, Islamabad, Pakistan.

Objective: The objective of the thesis under the field of the Component Based Software Engineering is to provide a mechanism to select a best qualifying reusable component on the basis of functional requirements and also to store information about all the reusable components in a single repository and to transform it into one way of description by the consensus of the Owner Organization for the purpose to make selection among many available components easy.

Undertaken By: Younas Wahab

Reg. No. 76-FAS/MSSE/F05

Supervised By: Dr. Naveed Ikram,

Associate Professor,

International Islamic University Islamabad, (IIUI)

Islamabad.

Started On: September 2006

Completed On: June 2008

Research Area: Component Based Software Engineering

Tools: Teradata RDBMS V. 7.1

Abstract

This work contributes a novel approach for selection of reusable components on the basis of functional requirements using MSR (Making Single Repository) mechanism. Different repositories are available for reusable components and to select the best qualifying component from those repositories, Customers must visit all those one by one and will select the best qualifying component on the basis of their requirements. MSR selects the reusable component on the basis of the Functional requirements, and as it's a black box process so the source on the basis of which the component's qualification will be checked is the description of the component which will be provided by the *Owner Organization.* MSR main purpose is to provide a single point of the access to the customers from where they can select their required component instead of visiting all the repositories one by one. MSR extract information about the reusable component from different repositories, transform the description provided by the owner organization so that we can classify it and remove the redundant information and to store it on the single place i.e. MSR repository. MSR process consists of Extraction, Transformation, Loading and Component selection steps. In *Extraction step,* information is extracted from the Owner Organization, in *Transformation step,* information will be transformed, redundancy will be removed and components will be classified on the basis of Type of the components, in *Loading step* the extracted information will be loaded into the single MSR repository, and in the *Component Selection step*, customer will set priorities of all information provided and the component will be selected totally on the customer's choice. MSR has made the selection easy and requires very less time as compare to other approaches as all the components are available in one place and also gives us an accurate result.

Acknowledgements

I am grateful to all those who have provided me their support and guidance throughout this research. First and foremost I am thankful to my supervisor of this work, Dr. Naveed Ikram. Without his unending interest and insightful guidance this work might have remained incomplete.

Younas Wahab

76-FAS/MSSE/F05

Dedication

I would like to dedicate my work to

ALMIGHTY ALLAH*, MY *FAMILY and FRIENDS

Declaration

I hereby declare and affirm that this thesis neither as a whole nor as part thereof has been copied out from any source. It is further declared that I have completed this thesis and accompanied software application entirely on the basis of my personal efforts, made under the sincere guidance of my supervisor. If any part of this report is proven to be copied out or found to be a reproduction of some other, I shall stand by the consequences. No portion of the work presented in this report has been submitted in support of an application for other degree or qualification of this or any other university or institute of learning.

Younas Wahab

76-FAS/MSSE/F05

Table of Contents

Chapter 1

INTRODUCTION

1 Introduction

If we look around us, we will find data being digitized in all its possible form resulting in need of best quality software systems in least possible time. In this regard a big challenge that organizations are facing is to provide a best quality product within least time. If they do so by hiring more developer and experts for development of the high quality product with in less time they will result with a problem of high budget as they will pay more for more experts and low quality. Organizations face the problem to develop a software product of high quality with in less time and low budget. To face this challenge component based software engineering provides a solution for it. In component based software engineering we develop software from already build software component [7]. The already build software components which have already been tested in the same environment would be reused. Through reusability we can achieve our goal of developing a product of high quality and reliability and with in less time and low cost [9]. For reusability purpose software engineering is divided into Domain Engineering and Application Engineering [11]. In domain engineering component for reuse is created and stored in reuse library [12]. During Application Engineering we select our required component from the reuse library and reuse it for our purpose. The next problem that CBSE face is how to select the best qualifying component. Qualification means to check how much a reusable component is according to our requirements [11]. Different authors have presented different approaches for the selection of reusable component and these approaches have some limitations which have been discussed in sect. 3 of this document.

Main focus of this work is how to select the best qualifying component among the available components in different libraries. As discussed above there are many repositories available for the reusable components and to select the best required component it's a very difficult job to decide that from where the Customers should take start and how to compare the component to select the best one.

In this document MSR (Making Single Repository) approach has been presented. MSR's main focus is developing a central single repository from where Customers can select their required component through functional requirements.

Teradata has been chosen for implementation because of the reason that we are extracting information from heterogeneous source e.g. flat file, relational databases etc and then after transformation we load that data into a single repository. MSR approach is based on ETL (Extraction, Transformation, and Loading) and Teradata provides us utilities like FastLoad, MultiLoad and FastExport. Through Fastload and *MultiLoad* we can load data into a warehouse [17] from heterogeneous sources and through FastEport we can export data from Terdata RDBMS to Flat Files or Tables etc [18].

1.1 *The Research Perspective*

In Component Based Software Engineering, to select the best required based on the Customer's requirements is a big challenge. Different organizations are busy in making reusable component and storing them in a reuse library. Different components developed by different organizations and stored in different Reuse Libraries can have functionalities in common. Customer's are facing the problem that how to select the best qualifying component among the available component on the basis of functional requirements. The main reason of the problem is that every organization has their own way of describing the functionality of the component and there is no any available system to compare and tell the Customer that which component fulfills his maximum requirements.

Different organizations have their own way of describing the components and due to difference in description and different reuse libraries Customer must visit all the available repositories to select his best required component, but after visiting all the repositories still he is facing a problem that how to compare components in different repositories and to select the best one.

In this perspective MSR approach solves the problem discussed above, and makes the Customers able to select the required component in just a single go. MSR approach saves component selection time plus provides a best qualifying component to customer without visiting all the libraries.

1.2 *The Research Objectives*

The main objective of the thesis is to provide a single point of access to customers from where they can select their required component on the basis of functional requirements.
In MSR approach all the information about the functional requirements of the components from different repositories will be extracted and will be loaded into a single repository. MSR extracts information from different Owner Organization, transform it and classify it and then Load it to a single repository and from that single repository customers can select their required component instead of visiting all the available repositories. MSR saves selection time and provide accurate result totally based on the Customer's choice.

1.3 *Research Methodology*

To answer the research question, our research process consists of the following four steps: 1) Literature Survey 2) Problem 3) presentation of the solution to the problem 4) case studies to validate and implement the proposed solution.

The literature survey portion shows us the present approaches for the component selection and their limitations. It shows us that why MSR approach is required and how what problem related to the reusable component selection will be solved.

The Problem shows us all the problems that exist in the current available component selection techniques and shows all the question on the basis of which MSR approach has been proposed.

In the Solution portion of research, all the research questions have been explained in detail. Also the process and detailed process of the MSR approach has been presented and explained in detail.

In the Implementation part of this research two different case studies have been presented to validate the MSR approach and shows that how it answer the all the research problems.

1.4 *Problem*

Different organizations are busy in making the reusable components, and after making it they store it in their reuse library. All the organizations have their own way of describing their component's functionality and all of their information are available on their own repository.

Different components can be developed for the same purpose by different organization and ever component will be described in different way. Now if the Customer wants to select his best required component from different repositories and to select the one best component which fully qualifies his functional requirements, then he must visit all the available repositories and will compare the available components that have been built for the same purpose. But the problem is that as the components are stored in different repositories and also have no any description standard, so it's not possible to select the one best component by comparing them with each other. The problem has been discussed in detail, in chapter 3, where all the questions that can come into the mind of the Customer while selecting the reusable component has been stated, and then in Chapter 4, it has been explained that how MSR approach provides a solution to the above mentioned problem.

1.5 *Implementation Process for MSR*

To *Extract* information about the functional requirements of all the available components in different repositories and to *Load* them into a single repository and then to *Transform* it for the purpose to remove redundant information and to classify it on the basis of type of the component, we are using ETL process which is the most important Data warehousing process.

1.5.1 ETL (Extraction, Transformation, Loading)

We load data into a data warehouse by using ETL and that's why ETL is the most important step in building a data warehouse [19]. ETL process can be used to load data to any database or warehouse. ETL can be used for data migration i.e. to populate the data warehouse and then

update it later on [4]. To build a data warehouse we will populate it with the data available in different forms and from different sources. ETL which stands for Extraction, Transformation and Loading consists of the following main three steps.

i) Extraction
ii) Transformation
iii) Loading

i) Extraction

The first step in ETL is extraction i.e. to populate a Data Warehouse we will extract data from different source. Data can be extracted from many sources like flat files, RDBMS and non-relational database structures like IMS and also can load extract data from data structures like VSAM, ISAM [19].

ii) Transformation

Transformation stage modifies the source data that has been extracted from many sources by applying many functions and rules before loading into a target [19]. Transformation will be applied on the source data if it is required. Transformation supports different functionalities like selecting certain columns from source data, splitting one column into two, adding the missing values, and validation etc.

iii) Loading

After extraction and required transformation the next step in ETL is to load data into a target. The constraints and triggers defined in the database schema activates as the Load phase interact with the Database [19].

1.5.1.1 Available Utilities for ETL

For ETL, there are different utilities available and here we will discuss the following three utilities in detail which Teradata RDBMS provides us. Teradata also provides us utilities like Tpump, TPT but here we will only discuss the following three.

i) FastLoad
ii) MultiLoad
iii) FastExport

i) FastLoad:

FastLoad is a highly reliable utility and is designed to move large amount of data. FastLoad can collects data from different sources i.e. Channel attached or network attached, and loads it into empty tables [22]. FastLoad is best choice only in case when there is no data in the Table.

Following is a sample script given for FastLoad utility sample script taken from wikipidia.org [22].

****

A Sample Script

********** **FastLoad Script Start** **********

```
sessions 2;
errlimit 25;
logon tdpid/username,password;


CREATE TABLE employee (
  EmpNo SMALLINT FORMAT '9(5)' BETWEEN 10001 AND 32001 NOT NULL,
  Name VARCHAR(12),
  DeptNo SMALLINT FORMAT '999' BETWEEN 100 AND 900 ,
  PhoneNo SMALLINT FORMAT '9999' BETWEEN 1000 AND 9999,
  JobTitle VARCHAR(12),
  Salary DECIMAL( 8,2) FORMAT 'ZZZ,ZZ9.99' BETWEEN 1.00 AND 999000.00 ,
  YrsExp BYTEINT FORMAT 'Z9' BETWEEN -99 AND 99 ,
  DOB DATE FORMAT 'MMMbDDbYYYY',
  Sex CHAR(1) UPPERCASE,
  Race CHAR(1) UPPERCASE,
  MStat CHAR(1) UPPERCASE,
  EdLev BYTEINT FORMAT 'Z9' BETWEEN 0 AND 22,
  HCap BYTEINT FORMAT 'Z9' BETWEEN -99 AND 99 )
  UNIQUE PRIMARY INDEX( EmpNo ) ;
```

```
set record unformatted;
define
  delim0(char(1)),
  EmpNo(char(9)), delim1(char(1)),
  Name(char(12)), delim2(char(1)),
  DeptNo(char(3)), delim3(char(1)),
  PhoneNo(char(4)), delim4(char(1)),
  JobTitle(char(12)), delim5(char(1)),
  Salary(char(9)), delim6(char(1)),
  YrsExp(char(2)), delim7(char(1)),
  DOB(char(11)), delim8(char(1)),
  Sex(char(1)), delim9(char(1)),
  Race(char(1)), delim10(char(1)),
  MStat(char(1)), delim11(char(1)),
  EdLev(char(2)), delim12(char(1)),
  HCap(char(2)), delim13(char(1)),
  newlinechar(char(1))
file=insert.input;

show;

begin loading employee errorfiles error_1, error_2;
insert into employee (
  :EmpNo,
  :Name,
  :DeptNo,
  :PhoneNo,
  :JobTitle,
  :Salary,
  :YrsExp,
  :DOB,
  :Sex,
  :Race,
  :MStat,
  :EdLev,
  :HCap
);

end loading;
logoff;
```

********** **FastLoad Script End** **********

ii) MultiLoad

MultiLoad is a loading utility of Teradata Database. MultiLaod can insert, update, upsert and delete large amount of data into empty and populated tables. MultiLoad is parallel loading Utility and can load data to more than one table at a time [23].

Following is a sample script given for MultiLoad utility taken from wikipidia.org [23].

****

A sample script

********** **MultiLoad Script Start** **********

```
.LOGTABLE ${DX_LOG}.TAB1_LOG;
/*
################################################## ############################
# Logon String for Teradata
################################################## ############################
*/
.LOGON ${LOGON};
/*
################################################## ############################
# MLOAD Tables
################################################## ############################
# MultiLoad Work Tables
# The data is loaded into worktable before it is sent to the target
# table.
################################################## ############################
# MultiLoad Error Tables
# The tables with ERR suffix contains error results
# The tables with UV suffix contains duplicate records
################################################## ############################
# Error Limit at which to stop Multiload
# ERRLIMIT : The number of rejected records before failure
# This excludes the Duplicate records
# CHECKPOINT : Multiload checkpoint after X number of records
# SESSIONS : Maximum sessions should be equal to the number
# of AMPs on the Teradata Server
################################################## ############################
*/
.BEGIN IMPORT MLOAD TABLES
${DX_TAB}.TAB1
,${DX_DUP}.TAB1
WORKTABLES
${DX_WRK}.TAB1_WT1
,${DX_WRK}.TAB1_WT2
ERRORTABLES
${DX_UTL}.TAB1_ERR1
${DX_UTL}.TAB1_UV1
,${DX_UTL}.TAB1_ERR2
${DX_UTL}.TAB1_UV2
ERRLIMIT 1000
CHECKPOINT 100000
SESSIONS 4
;

/*
################################################## ############################
# Layout of the File from where the data will be feed to the database
################################################## ############################
*/
.LAYOUT LAYOUT_NAME INDICATORS;
.FIELD x * INTEGER;
.FIELD y * VARCHAR(20);
.FIELD z * BYTEINT;
.FIELD RECORD_STAT_IND * CHAR(1);
```

```
/*
################################################## ##########################
# Business Rule Defined File :
# Telenor DW -Design Document- V1.4 - 04-Aug-2003
################################################## ##########################
# Section X.X.X
# Description :
################################################## ##########################
*/
.DML LABEL NEW_RECORD;
INSERT INTO ${DX_TAB}.TAB1
(
x
,y
,z
)
VALUES
(
:x,
:y,
:z
);
/*
################################################## ##########################
# Business Rule Defined File :
# Telenor DW -Design Document- V1.4 - 04-Aug-2003
################################################## ##########################
# Section X.X.X
# Description :
################################################## ##########################
*/
.DML LABEL DUP_RECORD;
INSERT INTO ${DX_DUP}.TAB1
(
x
,y
,z
)
VALUES
(
:x,
:y,
:z
);
/*
################################################## ##########################
# Business Rule Defined File :
# Telenor DW -Design Document- V1.4 - 04-Aug-2003
################################################## ##########################
# Section X.X.X
# Description :
################################################## ##########################
*/
.DML LABEL UPD_RECORD;
UPDATE ${DX_TAB}.TAB1
SET
x = :x
```

```
,y = :y
WHERE
z = :z
;
/*
##################################################### ############################
# Business Rule Applied
# Telenor DW -Design Document- V1.4 - 04-Aug-2003
##################################################### ############################
# Section X.X.X
##################################################### ############################
# RECORD STATUS CODE
# Below is an example - modify according to your script
# I - Insert Record
# U - Record needs to be updated
# D - Records already exist in the target table
##################################################### ############################
# INSERTION , UPDATION & DUPLICATES AS DEFIN
```

********** **MultiLoad Script End** **********

iii) FastExport:

FastExport utility export data from database to flat file or some other small database. FastExport is the reverse of FastLoad utility. FastExport utility can export data to network attached or channel attached in same way like FastLoad [24].

Sample script given below has been taken from wikipidia.org [24].

****

A Sample Script

********** **FastExpor Script Start** **********

```
.LOGTABLE utillog;                    /*define restart log */

.LOGON tdpz/user,pswd;                /*DBC logon string */

.BEGIN EXPORT                         /*specify export function */
SESSIONS 20;                          /*number of sessions to be used */

.LAYOUT UsingData;                    /*define the input data */
.FIELD ProjId * Char(8);              /*values for the SELECT */
.FIELD WkEnd * Date;                  /*constraint clause. */

.IMPORT INFILE ddname1                /*identify the file that */
LAYOUT UsingData;                     /*contains the input data */

.EXPORT OUTFILE ddname2;              /*identify the destination */
                                      /*file for exported data */

SELECT EmpNo, Hours FROM CHARGES      /*provide the SQL SELECT */
WHERE WkEnd = :WkEnd                  /*statement with values */
AND Proj_ID = :ProjId                 /*provided by the IMPORT */
ORDER BY EmpNo;                       /*command */

.END EXPORT;                          /*terminate the export */
                                      /* operation */

.LOGOFF;                              /*disconnect from the DBS */
```

----------**FastExport Script End**----------

1.5.1.2 ETL Tools

ETL is the first and important step in data warehousing. Different tools have been provided in the market for the designing and populating databases as shown in the fig.1 [20].

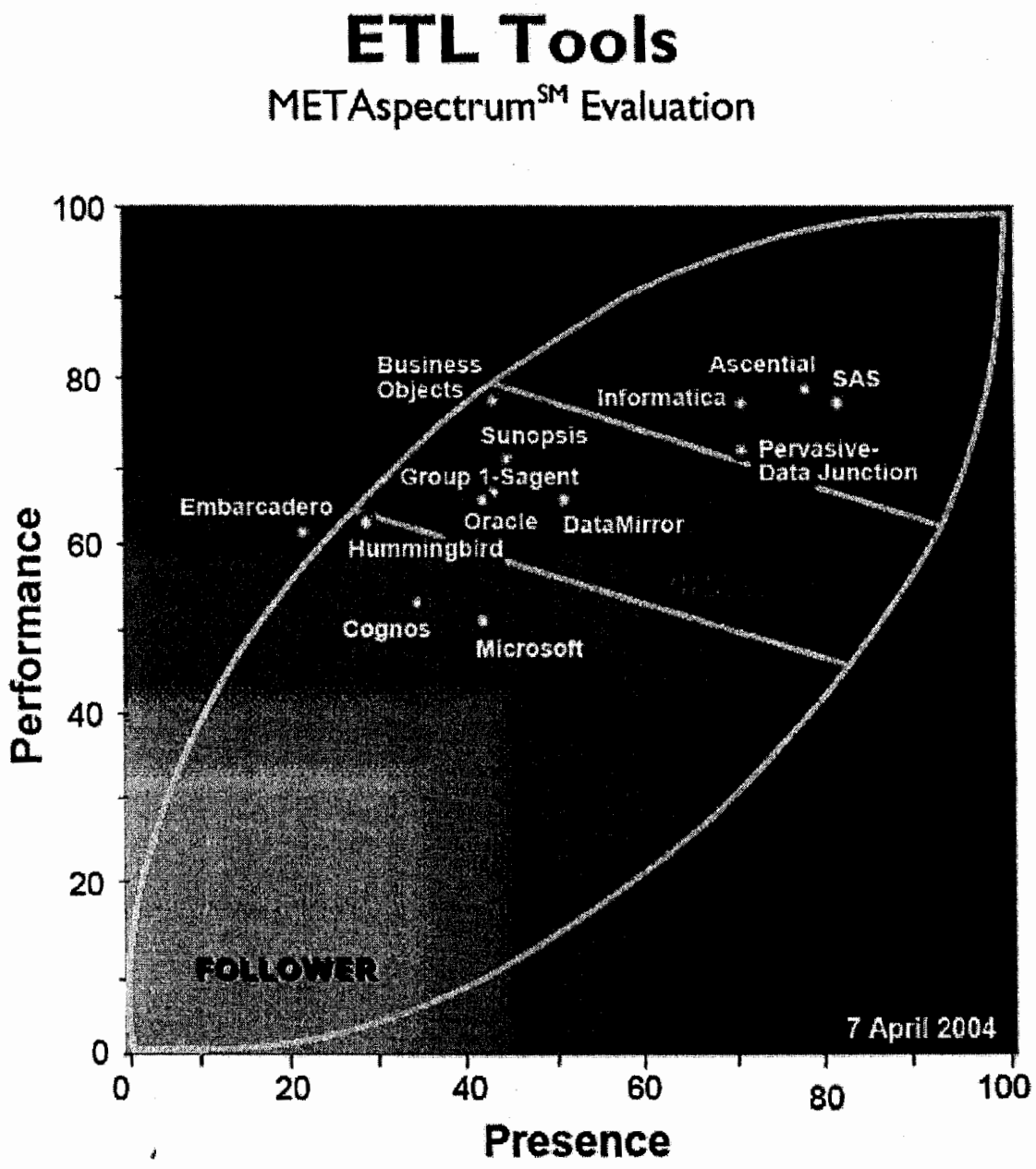


Fig. 1 ETL Tools [20]

1.5.1.2.1 ETL Tool Selection:

To decide that whether to buy a third party tool or to develop an ETL routine from the scratch depend on the following conditions [21].

Data complexity: For more complex data it is recommended to by ETL tool.
Data cleansing: If a thorough cleansing is required before loading the buying ETL tool will be the best choice otherwise it's better to build ETL routines for from the scratch.
Data Volume: For large amount of data it will be the best choice to buy an ETL tool. As ETL tools supports fast movement of data.

1.6 *Thesis Outline*

The remainder of this is structured as follows:

Chapter 2 describes the literature survey, which shows already present approaches for the reusable component selection and their limitations. The literature survey is to support our problem, and to justify that why MSR approach is required.

Chapter 3 describes the problem on which the whole thesis is based. This chapter gives full detail that why MSR approach is required, and also shows that through MSR approach what kind of issues and difficulties during component selection can be solved which is still facing by the current available approaches.

Chapter 4 presents the proposed solution i.e. MSR approach. The MSR process has been explained in detail. In this chapter it has been shown that how to follow MSR process to make a single MSR repository. It has also been shown that how the Customer can interact with the MSR repository and how he can select his best required component on the basis of functional requirements.

Chapter 5 validates the MSR approach by implementing two case studies through MSR approach and shows the result on the basis of which Customer can get his best requirement component among the available many components available for the same purpose on the basis of functional requirements.

Chapter 6 discusses the conclusion of the whole work.

Chapter 7 discusses the future work for the MSR approach

Chapter 2

Literature Survey

2 Literature Survey

Maxym Sjachyn and Ljerka Beus-Dukic have presented a "Semantic Component Selection technique (SemaCS)". In SemaCS component's description is used to select and classify the reusable component, but the problem is that this approach has is a huge amount of description and redundant information and there no way clearly defined that how the user will select the best required component among the components available for the same purpose.[10]

Vijayan Sugumaran and Veda C. Storey have presented "A Semantic-Based Approach to Component Retrieval" for reusable components retrieval [15]. The approach is based on a reuse repository where the components have been stored and a natural language interface has been used to interact with the repository and to select the required components. The system architecture presented in their work [15] is shown below in Fig. 2.

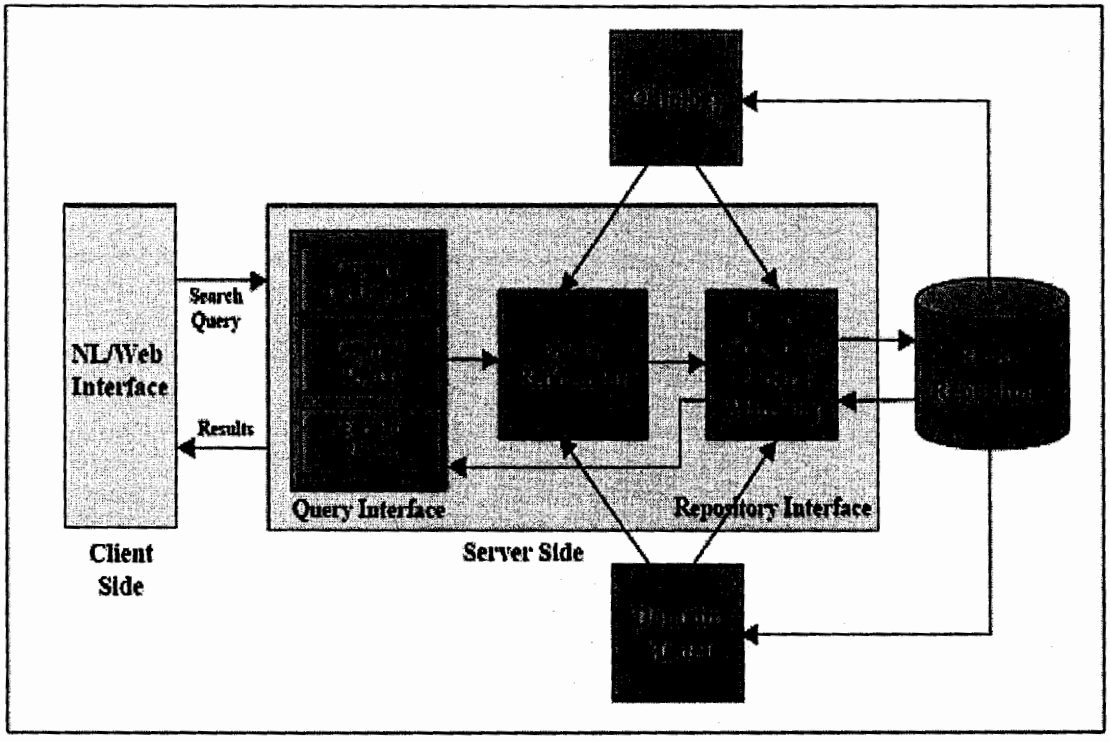


Fig. 2 System Architecture [15]

As shown in the Fig. 2 above the approach is based on the following steps.

Initial Query Generation

Query Refinement

Component Retrieval and feedback

User will write query in natural language and a natural language processing will be applied to that query to transform it to structured query language. After transforming the user's query to structured query language, Query Refinement step is applied on it to check that query has been written correctly or not and at end on the basis User's query the reusable component is provided to the User.

Haining Yao and Letha Etzkorn presented *"Towards A Semantic-based Approach for Software Reusable Component Classification and Retrieval"* approach for classification and retrieval of the software component [6]. In this approach again the user communicate with the reuse repository in natural language. The component has been retrieved by semantic matching Components Semantic description and user query semantic representation against the Domain Ontology. The following figure shows the system data flow diagram [6].

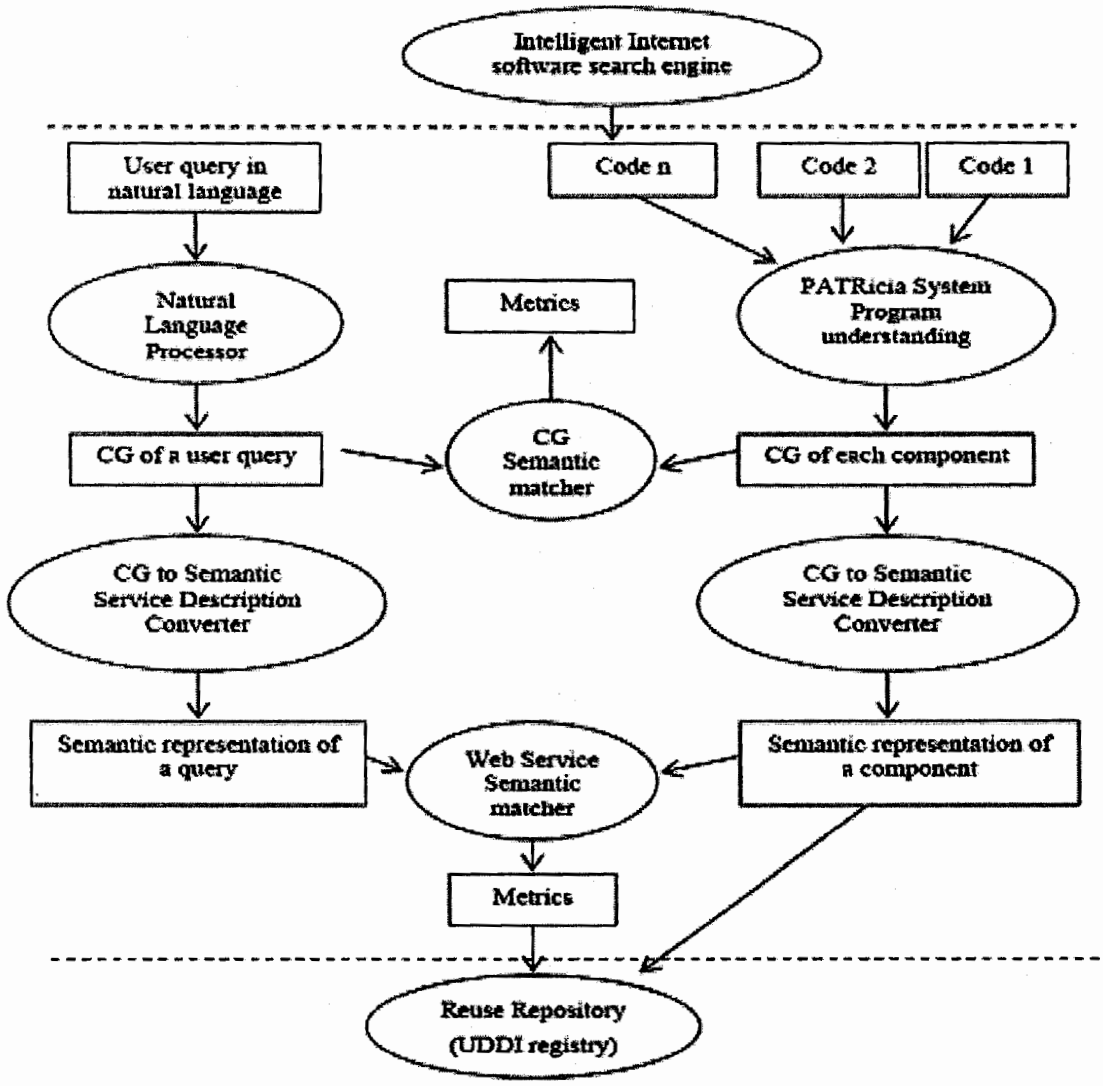


Fig.3 System Data Flow Diagram [6]

The user will enter his requirements in an unrestricted natural language. An intelligent natural language interface will transform this query into conceptual graph semantic representation in within a knowledge base and will be translated into semantic web based representation. The software component functionality will be identified by an analysis and annotation tool and will be translated into semantic web based representation. Finally semantic matchmaker will compare the component description and the user query in the conceptual graph and will retrieve the required component.

Jeffrey S. Poulin and Keith .J. Wcrkman presented "Melding Structured Abstracts anti the Worlcl Wide Web :for Retrieval of Reusable Components" for retrieval of reusable components [8]. In this approach World Wide Web browser "mosaic" has been used for Reusable Software Libraries (RSL) and also showed the way to quickly assed the component and to submit component to RSL using the "Structure Abstract" of reusable components. Fig. 4 shows us the Structure Abstract Search Facility present in this paper [8].

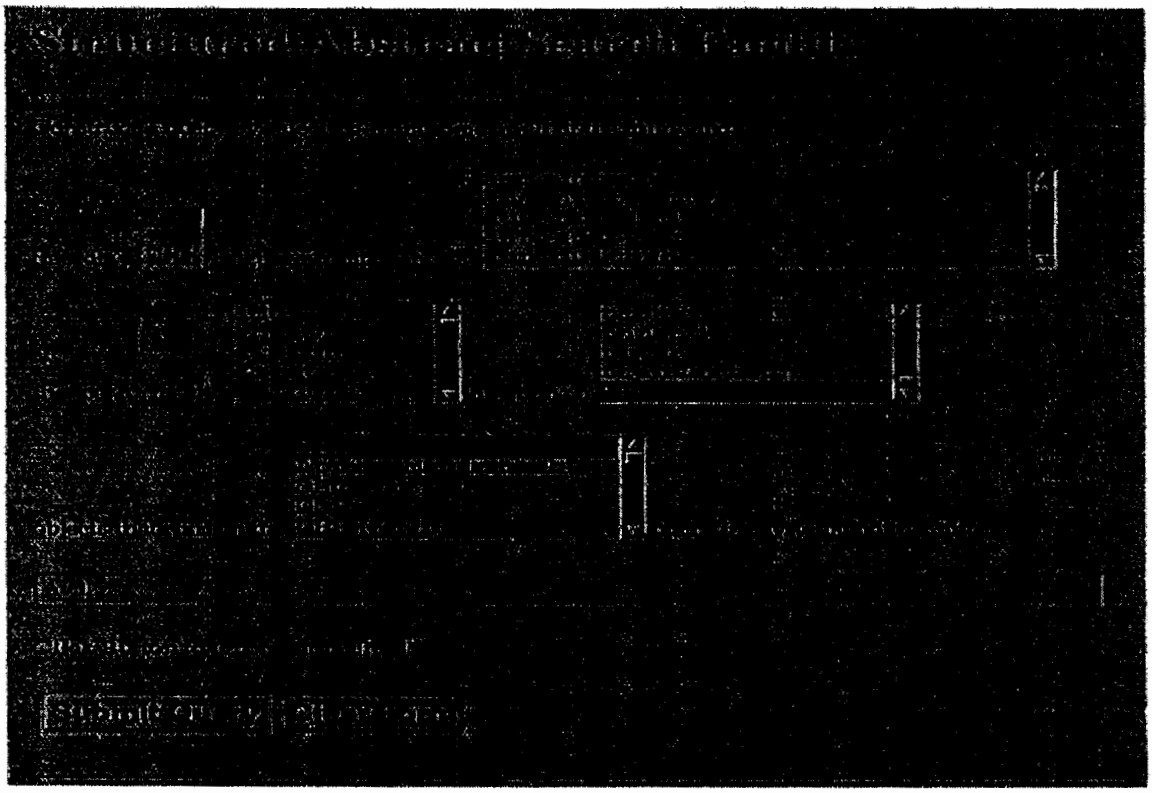


Fig. 4 Structured Abstract Search Facility [8]

Young Park and Ping Bai presented "Retrieving Software Components by Execution" approach for retrieval of reusable component from reuse library [13]. The approach is based on the executing the components by providing inputs generated systematically and on the basis of these inputs one should decide the either he need the component or not.

The following Fig. 5 shows the *Component retrieval by execution* taken from Young Park and Ping Bai [13] work.

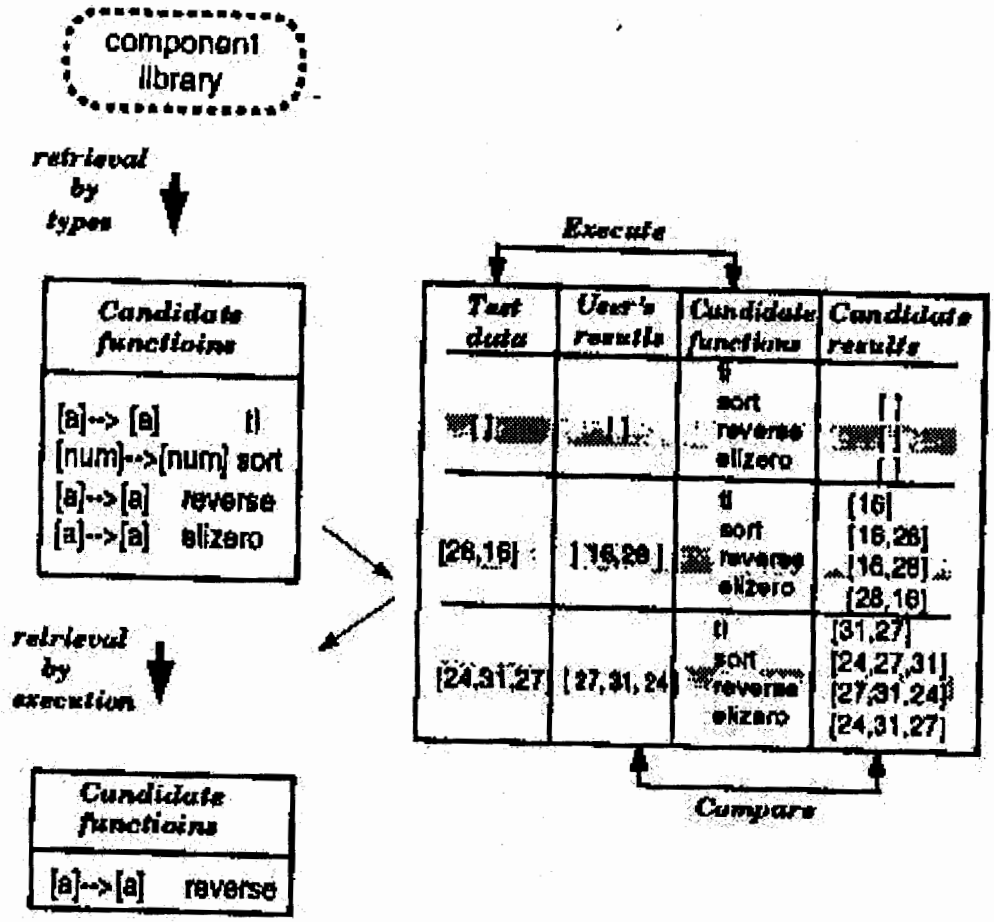


Fig. 5Component retrieval by execution [13]

We know that with Component Based Software Engineering we can develop a high quality product with in less time and low cost but on the other side we are facing many challenges in this field. Component Selection is one of the big challenges in CBSE and different authors have tried but still failed to find result with a desired performance, accuracy and user friendliness. *A. Mili et al* tried to solve this problem and presented a refinement based system where the repository is based on formal representation and a binary relation is used to present program specification [1]. But the technique is based on assumptions and requires Formal Method skills to be followed practically. Different authors have tried to solve the challenges that are facing by the CBSE as *Bernd Fischer* presented specification based retrieval for the selection of the reusable component [2], *Yonghao Chen and Betty H. C. Cheng* presented a way that how to Formalize and automate software reuse by using generality relation [16] and *Gerald C. Gannod et al* presented an automated approach for supporting software reuse through reverse engineering [5] but

somewhere we are just assuming the things and some where we are even not involving the User during component's selection. Vijayan *Sugumaran et al.* have given some different touch to component selection and they used domain model and object libraries to identify software components [14]. The domain Analyst will perform analysis to build a domain model. But in the same domain different components for the same purpose will be developed by different organization. To select the best required component the user must know about the functionalities provided by the components, but unfortunately this method does not provide any functional information to the User on the basis of which he can make decision about his best required component. Here a keyword based approach is presented. The keyword will be retrieved from the user's query written in natural language. The keywords will be mapped against the repository and the required component will be selected.

As discussed above all the approaches have some limitations. All organizations have their own development standards [3] and after developing the reusable component, they store it in their reuse library. The above approaches do not provide us the solution that how to select the best qualifying component among the available different repositories and how to compare component with each other if there are more than one component developed for the same purpose by different organization. The main focus of the MSR approach proposed in this thesis is to make a single repository and to transform the components that have been developed for same purpose into just one standard i.e. Customer's standard. Instead of visiting all the available repositories individually, MSR makes the Customers able to select their component from common MSR repository and makes it possible to compare components developed for the same purpose with each other as they all will be according to one standard and comparing them will not be a difficult task.

Chapter 3

Problem

3 Problem

In Component Based Software Engineering there are two types of organizations, the first type we have referred in our work as Owner Organization, who will be responsible for making reusable component, and the second type is Customer who will select the reusable component and will reuse it.

Different Owner Organizations are busy in making the reusable component, and after making it they keep it in a reuse library from where the Customers can access it. Every Owner Organization has his own development standard [3] through which they develop the reusable components and store it in their own library by describing the component's functionality in their own way. Different components for the same purpose can be developed by different organizations and every organization describes their component's description in different way. To select the one best required component on the basis of functional requirements among the available components, Customers are facing the problems as the components have been described in different way and also stored in different locations. Following are the few questions which can come to the mind of the Customer during the reusable component selection.

Q1. How many components are available for this purpose?

Q2. If more than one component available for this functionality then should I check all of them individually?

Q3. How will I compare all available components to select the one that fully qualifies my functional requirements?

Q4. Is there a simple way to select one best required component in just a single go?

MSR's main purpose is to make a single repository, and to Transform the components functionalities for the purpose to remove redundant information and to classify it for the purpose to be easily stored in a single repository. Due to MSR customers will no more visits all the repositories one by one as they will get all their required information on a single place.

Chapter 4

MSR Approach

4 MSR approach

With the passage of time there is more and more demand for high quality software in less time. To provide a high quality product with in the required time is a big challenge. Customers are demanding for a high quality product with in a very short deadline which is big challenge for developer organization. If the developer organization tries to do so by hiring more resources or using existing resources for overtime then it's difficult to control the project budget and also there is a big chance of poor quality product. The customers are not ready to compromise on quality and development organization are not ready to let the budget overrun. To solve problem faced by both of the parties i.e. customers and developers, Component Base Software Engineering (CBSE) is a best solution for it.

Component Based Software Engineering is based on integrating the already build and available components. With Component Based Software Engineering we can provide a high quality product with in less time. Using already build component does not need any special skill and also there is no need of more resources, so with limited available resources we can provide a high quality product with in less time and obviously with high quality.

In CBSE, there are actually two types of organizations, one that develops the component, and hence become the owner of that component, another is that selects that reusable component and reuse it in their own product. In this document we will refer to the component developing organization as *Owner Organization* and the second who will reuse the already build component as *Customer*.

Different *Owner Organizations* are busy in making the reusable components to make them available for *Customers* so that they can reuse it. After developing a reusable component; *Owner Organization* of that component place it in their reuse library so that to make it available for *Customers*. *Customers* from different location access that library to select their required component that fully satisfies their requirements.

To select the best required component, different approaches are already present, but all of them have some limitation as discussed in chapter 2 of this document. In MSR approach reusable component will be selected on the basis of the Customer's Functional requirements. To select his best required component *Customer* will put *Priority Values* in front of all functionalities that the component provides; and on the basis of that *Priority Values* components will be filtered out in the MSR repository and will be finally selected on the basis of Customer's Input & Output Information.

There are many *Owner Organizations* that are busy in making the reusable component and keeping them in the reuse libraries. Every *Owner Organization* has his own way of describing the reusable component's functionality. Different reuse libraries available from where the *Customer* can select his best required component and reuse it. Following four questions which can come in the mind of the *Customer* while selecting the reusable component have been explained in detail.

Q1. *How many components are available for this purpose?*

As shown in Fig. 6, as there are many *Owner Organizations* busy in making Reusable components, and after making, it they place it in their reuse repositories. *Customer* as shown in Fig. 6 is facing the problem that how to know about that components for his purpose are available or not, and if yes then how many are they.

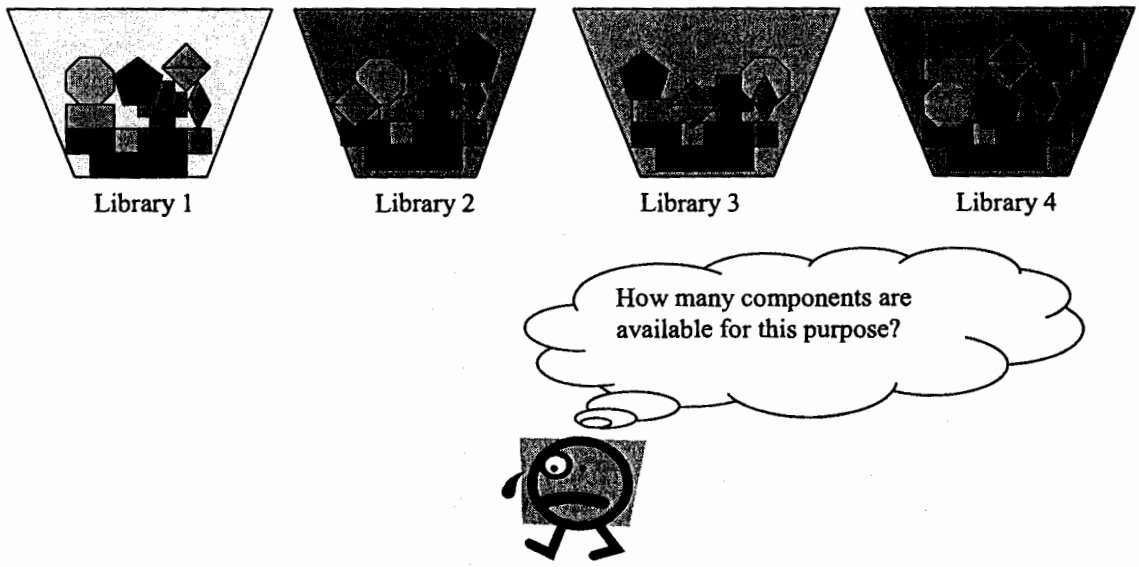


Fig. 6 Customer and Reuse Libraries

Q2. *If more than one component available for this functionality then should I check all of them individually?*

As discussed above, different owner organizations can develop different components for the same purpose. During the selection of the reusable component *Customer* is facing the problem that how to visit all the available Reuse Libraries and to check the individual component as this is a tedious and time consuming job.

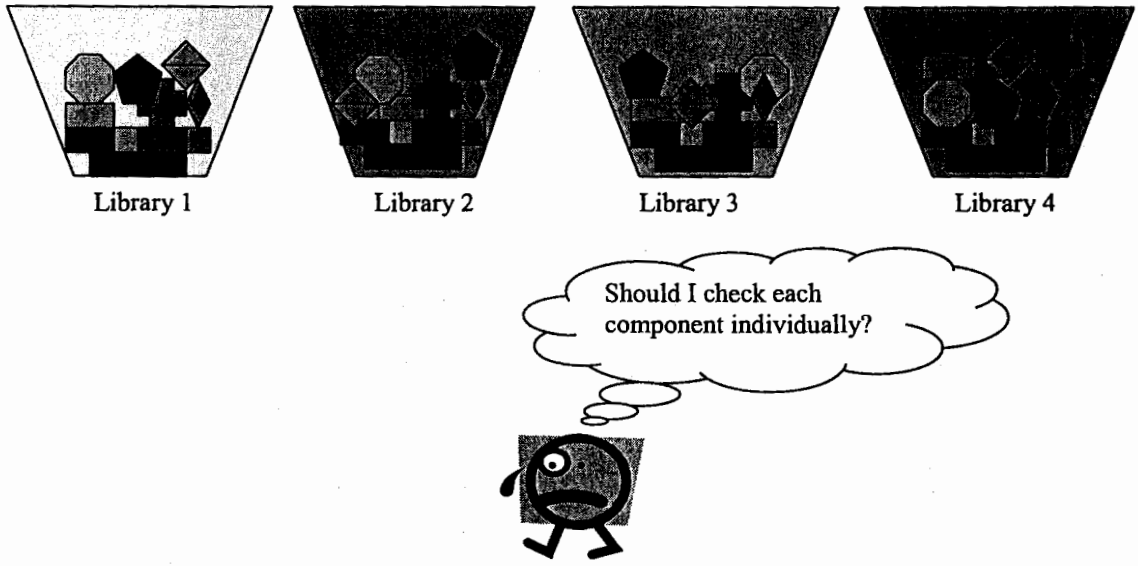


Fig. 7 Customer and Reuse Libraries

Q3. *How will I compare all available components to select the one that fully qualifies my functional requirements?*

Different *Owner Organizations* have described their component in their own way. To select the best component among the available components, it is important to compare them with each other, but if they will have different way of description from each other then how will we compare it. Here the *Customer* is facing the problem component's different description Fig. 8.

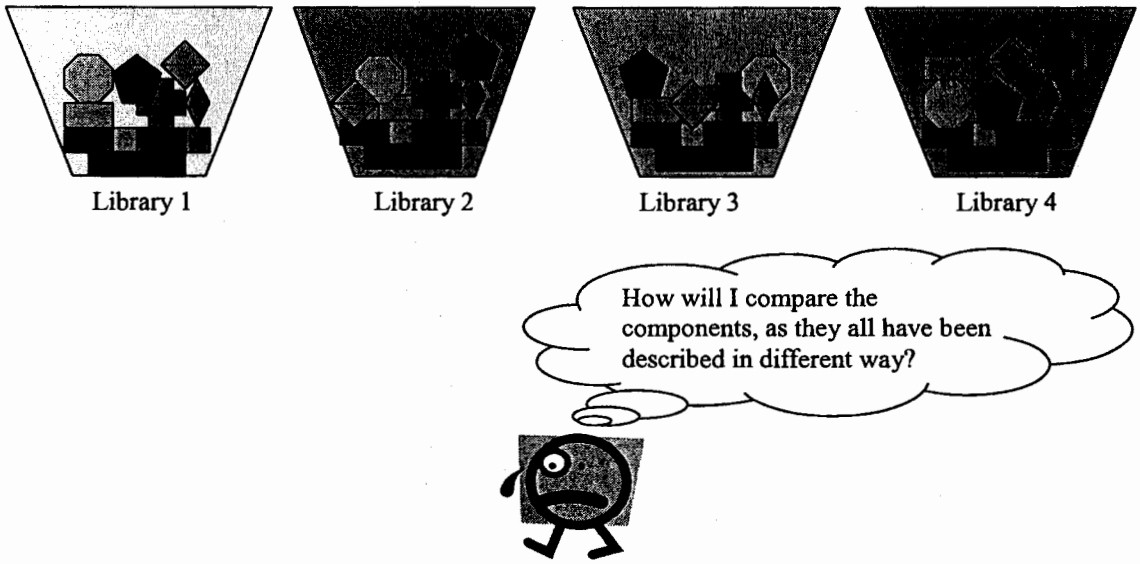


Fig. 8 Customer and Reuse Libraries

Q4. *Is there a simple way to select one best qualifying component in just a single go?*

As shown below in Fig. 9, there can be more than one Reuse Library and to visit them all is a tedious and time consuming job, also there is no simple way to compare the component of two repositories in between and to select the best one. Customer is demanding for single point of access instead of searching and looking at each individual library and its components.

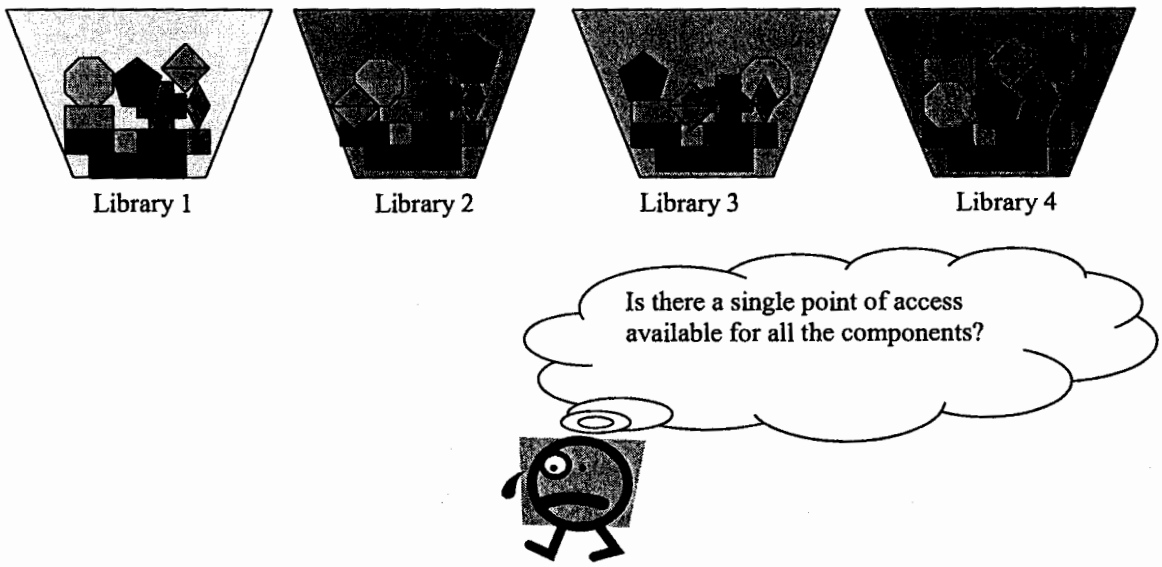


Fig. 9 Customer and Ruse Libraries

MSR approach is an answer to all the above mentioned questions. Instead of visiting all the reuse libraries, MSR provides a single point of access as shown in Fig. 10 from where a Customer can select his best required component on the basis of his requirements. In the coming section of this document the MSR process has been explained in detail.

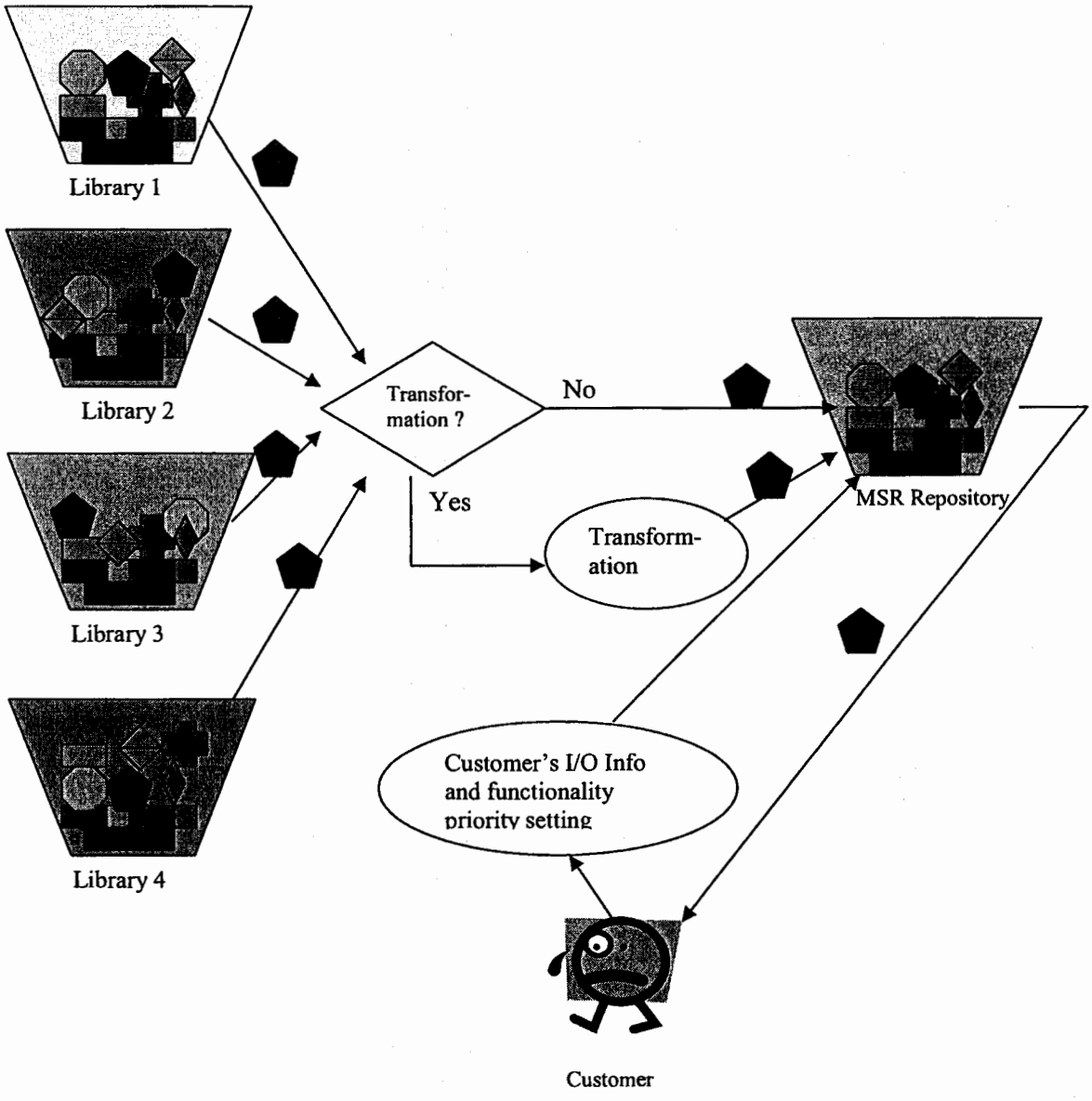


Fig. 10 MSR single Library

4.1 *MSR Process*

The main purpose of MSR is to provide a single point of access from where Customers can select their required component on the basis of Functional requirements. Different reusable component are available in different repositories developed by different organizations. As shown in Fig. 11 in MSR we extract information about the reusable components developed by different Owner Organization and bring it into one standard by performing transformation on it and then we load it into MSR repository. MSR repository contains information about the Owner Organization of the reusable component, functionalities provided by that component and information about the input and out put of that component. All the information that will be loaded into MSR repository will be provided by the Owner Organization of that component. The MSR approach mainly consists of the following four steps as shown in Fig. 11.

i) Extraction
ii) Transformation
iii) Loading
iv) Component Selection process

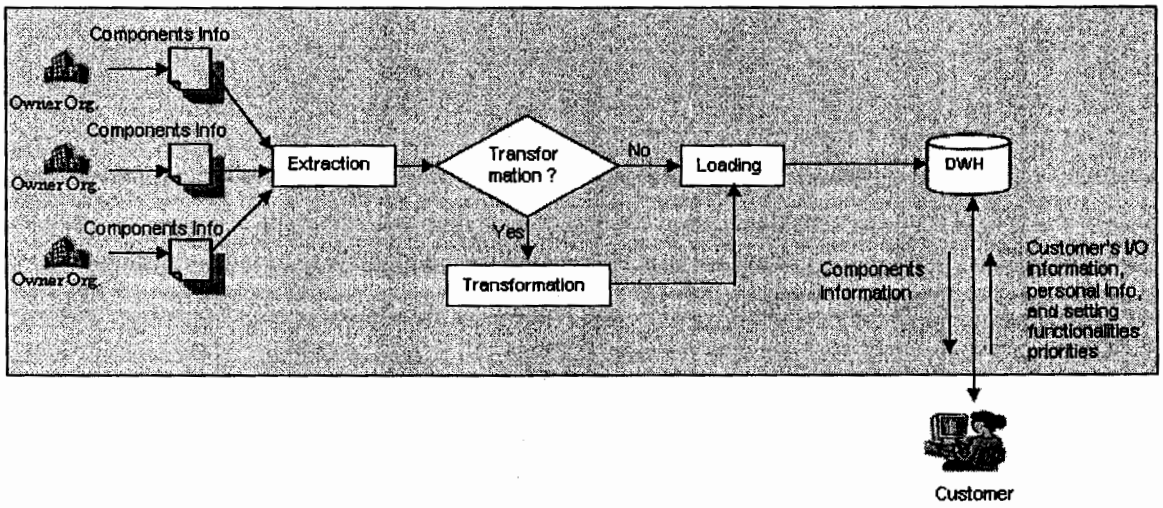


Fig. 11 MSR Process

4.1.1 Extraction

As discussed above, the main focus of the MSR approach is to make a single repository and a single point of access from where *Customers* can easily select their required component on the basis of Functional Requirements. To make the repository, we need information about the functionalities and I/O information of the component. Information

can be obtained from the *Owner Organizations* in form of heterogeneous sources like Flat File, Database Table, Excel Sheet etc and through Extraction which is data warehousing process that information will be retrieved.

For extraction we have different types of tools and utilities as discussed in chapter 2. Utilities like FastLoad, MultiLaod provided by *Teradata* RDBMS can be used to load information to data warehouse from heterogeneous sources and *FastExport* utility provided by *Teradata* RDBMS can export data from *Teradata* DBMS to any other source like flat files or small databases.

4.1.1.1 Transformation

In MSR process information from different organizations are retrieved and stored in MSR repository. As information about the reusable components are coming from different sources and from different organizations, so every Organization will describe their component in their own way. To make a single point of access and to store information about all the components in a single repository, only one standard way of describing the component functionality will be used and the description of all the components build for the same purpose will be transformed into that standard way of description by making consensus with the Owner Organization.

In MSR Transformation is mainly required in two (not restricted to) situations.

The first case in which Transformation is required is that different Components for the same purpose can be developed by different *Owner Organizations*. The problem can arise here that same functionality can be described in different way by different *Owner Organization.*

The I/O information about different components will be described by different organizations in different way. To make a single source of information all the I/O information will be transformed and will be presented in a same way.

Other cases like difference in naming conventions etc. can be handled through Transformation. Transformation as shown in MSR process is one of the most important step for making a single repository from which a *Customer* can easily select his required components.

Different Tools and Utilities are available to Extract and Transform the data as discussed in Chapter 2. Some tools supports parallel processing due to which transforming a large amount of data is not a big deal. Tools discussed in Chapter 2 also support built-in function for transforming data to any standard.

4.1.1.2 Loading

As shown in MSR process, after performing Extraction and Transformation the next step is to Load the data into a repository from were the Customer can easily access the data.

Loading in MSR means to make a single point of access for all the Customers from where they could select their required component on the basis of functional requirements. Now Customer will only visit MSR repository instead of visiting all the available repositories of different Owner Organization.

4.1.2 Component Selection Process

In MSR process as shown in Fig.11, the last step is the *Component Selection Process. Component Selection Process* is mainly consists of the following steps.

- Customer will set the priorities of all functionalities according to his requirements.
- Customer will set priorities of IO (Input and Output) for every functionality
- Customer will provide personal information, through which he can be tracked easily.

One of the benefits of MSR approach is that it fully involves *Customer* during Component selection and the reusable component will be selected on the basis of the priorities that will be assigned by the *Customer*. *Component Selection Process* has been explained in detail in the next coming section i.e. Detailed MSR Process.

4.2 *Detailed MSR Process*

In MSR process we have discussed different steps that MSR consists of, now in detailed process as shown in Fig.12 we are going to explain some technical detail of the MSR process that how we will build a Common Reusable Repository and how we will store the data in it and how we will provide reusable component to the *Customer*.

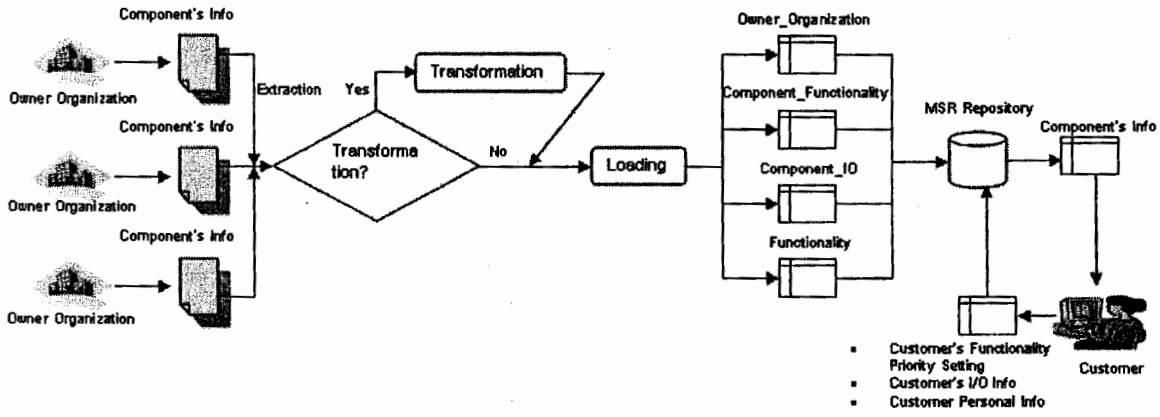


Fig. 12 MSR Detailed Process

As shown in Fig.12 the MSR process consists of the following steps

Extraction

Transformation

Loading

Component Selection Process

4.2.1 Extraction

For making the repository, we will extract the important information about the reusable components for the purpose to transform and load it into the repository. For making the reuse library we will demand the following information from the Owner Organization about the component.

- Information about the Owner Organization
  Owner Organization name
  Contact Detail
  Component they are providing

- Information about the Component's Functionality
  Functionality provided by the component
  Purpose

- Information abut Component's Input Output
  IO description of all functionalities

4.2.2 Transformation

After getting the above information we will check our reuse library if we already have any component for the same purpose. If we have already any component for the same purpose then we will apply transformation by keeping those components functionality in mind, and if we don't have any component for the same purpose in advance then we will only apply transformation on it without keeping any stored component in mind.

4.2.3 Loading

As shown in Fig. 12, after performing the two steps i.e. Extraction and Transformation the next step is Loading. In Loading we will load all the Extracted and Transformed information into the repository to make them available for the *Customers*. In Loading we will perform the following steps.

4.2.3.1 Loading information about the Owner Organization

After Extraction and Transformation we will load the following information into Owner Organization Table.

Component_id: To uniquely identify the component
Component_name: name of the reusable component
Org_name: Owner Organization's name who has developed this component
Phn_no: Phone Number of the Owner Organization
Address: Address of the Owner Organization
Mail_add: Mail address of the Owner Organization
Purpose: Specifies that for what purpose this component has been developed

Following figure shows the Owner_Organization Table in our CBSE database.

	ColumnName	Type
1	Component_id	VARCHAR
2	Component_name	VARCHAR
3	Org_name	VARCHAR
4	Phone_No	VARCHAR
5	Address	VARCHAR
6	Mail_Add	VARCHAR
7	Purpose	VARCHAR

Fig. 13 Owner Organization Table

4.2.3.2 Loading Information about component's Functionality

In our CBSE database we have the Functionality Table where will store information about the functionalities of the components. Functionality Table will keep the following information about the reusable components.

Funct_id: *To uniquely identify each functionality.*

F_Description: Description about the functionality of the component.

F_Priority: *Assign priority to each functionality*

	ColumnName	Type
1	Funct_id	VARCHAR
2	F_Description	VARCHAR
3	F_Priority	VARCHAR

Fig.14 Functionality Table

4.2.3.3 Component's Functionality Result (I/O) Information

To select the best required component on the basis of functional requirements, information about the Input and Output of all the functionalities that it performs is also required. In MSR process we will extract information about the IO and will store it in the repository as shown below in Fig. 15.

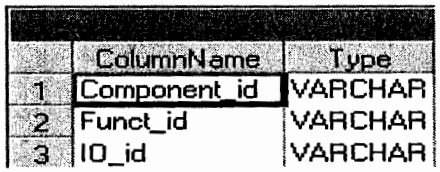
	ColumnName	Type
1	IO_id	VARCHAR
2	IO_Description	VARCHAR

Fig. 15 Component_IO Table

4.2.3.4 Relating Component's, Functionality and Result (I/O) information

MSR repository provides information about the reusable components to Customers, so that they can easily select their required component. MSR will keep information about the Component, Functionality and I/O of that functionality as shown in Figure.16

Component_id: To uniquely identify the component
Funct_id: To uniquely identify each functionality.
IO_Id: To uniquely identify IO informaition



	ColumnName	Type
1	Component_id	VARCHAR
2	Funct_id	VARCHAR
3	IO_id	VARCHAR

Fig. 16 Component_IO table

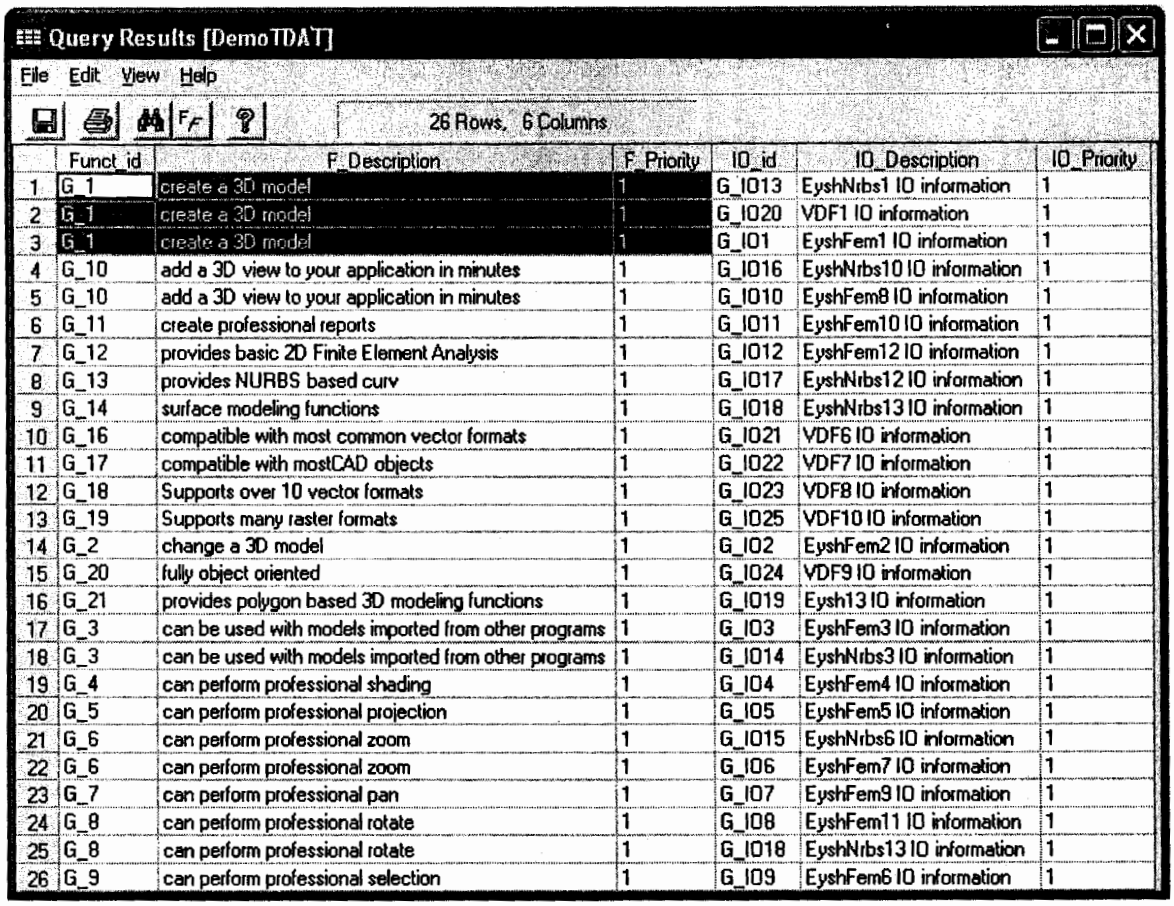
4.2.4 Component Selection Process

Component selection process is mainly consists of the following steps.

i) Information about the Customer

ii) Changing Functionality and I/O Priority Values

MSR provides detailed information about the components to make the Customers able to select their best required component on the basis of their requirements. MSR provides information about the Component's Functionality and also the IO information all functionalities that that component performs. Fig.17 shows the information that MSR will provide to the Customers for Component selection.



Query Results [DemoTDAT]

File Edit View Help

26 Rows, 6 Columns

	Funct_id	F_Description	F_Priority	IO_id	IO_Description	IO_Priority
1	G_1	create a 3D model	1	G_IO13	EyshNrbs1 IO information	1
2	G_1	create a 3D model	1	G_IO20	VDF1 IO information	1
3	G_1	create a 3D model	1	G_IO1	EyshFem1 IO information	1
4	G_10	add a 3D view to your application in minutes	1	G_IO16	EyshNrbs10 IO information	1
5	G_10	add a 3D view to your application in minutes	1	G_IO10	EyshFem8 IO information	1
6	G_11	create professional reports	1	G_IO11	EyshFem10 IO information	1
7	G_12	provides basic 2D Finite Element Analysis	1	G_IO12	EyshFem12 IO information	1
8	G_13	provides NURBS based curv	1	G_IO17	EyshNrbs12 IO information	1
9	G_14	surface modeling functions	1	G_IO18	EyshNrbs13 IO information	1
10	G_16	compatible with most common vector formats	1	G_IO21	VDF6 IO information	1
11	G_17	compatible with mostCAD objects	1	G_IO22	VDF7 IO information	1
12	G_18	Supports over 10 vector formats	1	G_IO23	VDF8 IO information	1
13	G_19	Supports many raster formats	1	G_IO25	VDF10 IO information	1
14	G_2	change a 3D model	1	G_IO2	EyshFem2 IO information	1
15	G_20	fully object oriented	1	G_IO24	VDF9 IO information	1
16	G_21	provides polygon based 3D modeling functions	1	G_IO19	Eysh13 IO information	1
17	G_3	can be used with models imported from other programs	1	G_IO3	EyshFem3 IO information	1
18	G_3	can be used with models imported from other programs	1	G_IO14	EyshNrbs3 IO information	1
19	G_4	can perform professional shading	1	G_IO4	EyshFem4 IO information	1
20	G_5	can perform professional projection	1	G_IO5	EyshFem5 IO information	1
21	G_6	can perform professional zoom	1	G_IO15	EyshNrbs6 IO information	1
22	G_6	can perform professional zoom	1	G_IO6	EyshFem7 IO information	1
23	G_7	can perform professional pan	1	G_IO7	EyshFem9 IO information	1
24	G_8	can perform professional rotate	1	G_IO8	EyshFem11 IO information	1
25	G_8	can perform professional rotate	1	G_IO18	EyshNrbs13 IO information	1
26	G_9	can perform professional selection	1	G_IO9	EyshFem6 IO information	1

Fig. 17 MSR Customer's Information Table

4.2.4.1 Information about the Customer

While selecting the component, Customer will provide the following information about himself.

Cust_id — *to uniquely identify the Customer*

Cust_nam — *Shows the name of the Customer*

Cust_phone_number — *Shows the Phone number of the Customer*

Cust_Address — *Shows the address of the Customer*

Cust_Mail_Address — *Shows the mail address of the Customer*

4.2.4.2 Changing Functionality and IO Priority Values

As shown in the MSR process after loading all the information about the reusable components, the information will be available in a single location, and now the customers can access and select their required component from that single location. As shown in Fig. 17 all the functionalities and the IO information will have priorities on the basis of which Customer will be able to select his required component.

Component Selection process is mainly based on the priorities of the Functionalities and its associate IO information, which the Customer will assign during the Component Selection process. Functionality and IO can have any one of the following Priority Values.

H: High priority
M: Medium Priority
L: Low Priority
N: Not Required

For calculation purpose priority values have assigned the following numeric values which the Customer will assign during the Component's selection process.

H= 0.6.....1.0
M= 0.3.....0.5
L= 0.1.....0.2
N=0

The following formula will be used to determine that how much each component qualifies Customer's requirements.

$$Q_c = \sum_{i=1...N} (FPi * IOi)/ T * 100 \quad \text{..... (1)}$$

Where **FP** is Functionality Priority, **N** is the number of functionalities offered by this component, **T** is Total Number of unique transformed Functionalities for the same purpose and Q_c shows percent qualification of component.

By default in the Functionality as shown in Fig. 17 all the Functionalities and their associated IO information will have 1 i.e. H priority value. Then to select the Component, the Customer will change Priority Value to any one of the above mentioned values on the basis of which his best required component will be selected.

Chapter 5

Implementation

5 Implementation

To implement the MSR approach we have taken the following two cases

Case 1: Selection of the Graphics component

Case 2: Selection of the Barcode Component

5.1 *Case 1, Selection of the Graphics Component*

For implementing the MSR approach, in the first case ware going to select one best qualifying component among the available four graphics components for the same purpose.

Eyeshot Fem

create a 3D model|
change a 3D model|
can be used with models imported from other programs|
can perform professional shading|
can perform professional projection|
can perform professional zoom|
can perform professional pan|
can perform professional rotate|
can perform professional selection|
add a 3D view to your application in minutes|
create professional reports|
provides basic 2D Finite Element Analysis|

Eyeshot Nurbs

create a 3D model|
change a 3D model|
can be used with models imported from other programs|
can perform professional shading|
can perform professional projection|
can perform professional zoom|
can perform professional pan|
can perform professional rotate|
can perform professional selection|
add a 3D view to your application in minutes|
create professional reports|
provides NURBS based curv|
surface modeling functions|

Eyeshot

create a 3D model|
change a 3D model|
can be used with models imported from other programs|
can perform professional shading|
can perform professional projection|
can perform professional zoom|
can perform professional pan|
can perform professional rotate|
can perform professional selection|
add a 3D view to your application in minutes|
create professional reports|
provides polygon based 3D modeling functions|

VectorDraw Developer Framework (VDF)

create 2D drawings|
create 3D drawings|
manage 2D drawings|
manage 3D drawings|
print 2D and 3D drawings|
compatible with most common vector formats|
compatible with mostCAD objects|
Supports over 10 vector formats|
Supports many raster formats|
fully object oriented|

To select the best qualifying component among the available four graphics on the basis of Customer choice, we will implement MSR approach step by step.

5.1.1 Extraction

As shown in MSR process we will extract the following information about the components

Component's Name

Owner Organization Name

Phone #

Address

Mail Add

Purpose

Component's Functionalities

In the present case we have four graphics component, and we will extract information about all the four as follows.

Component No. 1

Owner Organization Name: USoft

Component's Name: Eyeshot Fem

Phone #: +92-51-9332624

Address: Street No. 152, H#59, G-9/4 Islamabad Pakistan

Mail Add: Info@usof.com

Purpose: Graphics

Component's Functionalities:

F#	Functionalities	Input and Output Information
EyshFem1	create a 3D model|	EyshFem1 IO information
EyshFem2	change a 3D model|	EyshFem2 IO information
EyshFem3	can be used with models imported from other programs|	EyshFem3 IO information
EyshFem4	can perform professional shading|	EyshFem4 IO information
EyshFem5	can perform professional projection|	EyshFem5 IO information
EyshFem6	can perform professional zoom|	EyshFem6 IO information
EyshFem7	can perform professional pan|	EyshFem7 IO information
EyshFem8	can perform professional rotate|	EyshFem8 IO information
EyshFem9	can perform professional selection|	EyshFem9 IO information
EyshFem10	add a 3D view to your application in minutes|	EyshFem10 IO information
EyshFem11	create professional reports|	EyshFem11 IO information
EyshFem12	provides basic 2D Finite Element Analysis|	EyshFem12 IO information

Component No. 2

Component's Name: Eyeshot Nurbs

Owner Organization Name: Inova

Phone #: +92-51-9332655

Address: Street No. 48, H#595, F-6/4 Islamabad Pakistan

Mail Add: Info@inova.com

Purpose: Graphics

Component's Functionalities:

F#	Functionalities	Input and Output Information
EyshNrbs1	Create a 3D model|	EyshNrbs1 IO information
EyshNrbs2	Change a 3D model|	EyshNrbs2 IO information
EyshNrbs3	can be used with models imported from other programs|	EyshNrbs3 IO information
EyshNrbs4	can perform professional shading|	EyshNrbs4 IO information
EyshNrbs5	can perform professional projection|	EyshNrbs5 IO information
EyshNrbs6	can perform professional zoom|	EyshNrbs6 IO information
EyshNrbs7	can perform professional pan|	EyshNrbs7 IO information
EyshNrbs8	can perform professional rotate|	EyshNrbs8 IO information
EyshNrbs9	can perform professional selection|	EyshNrbs9 IO information
EyshNrbs10	add a 3D view to your application in minutes|	EyshNrbs10 IO information
EyshNrbs11	Create professional reports|	EyshNrbs11 IO information
EyshNrbs12	provides NURBS based curv|	EyshNrbs12 IO information
EyshNrbs13	Surface modeling functions|	EyshNrbs13 IO information

Component No. 3

Component's Name: Eyeshot

Owner Organization Name: Techlogix

Phone #: +92-51-9232655

Address: Street No. 105, H#678, H-10/4 Islamabad Pakistan

Mail Add: Info@techlogix.com

Purpose: Graphics

Component's Functionalities:

F#	Functionalities	Input and Output Information
Eysh1	create a 3D model|	Eysh1 IO information
Eysh2	change a 3D model|	Eysh2 IO information
Eysh3	can be used with models imported from other programs|	Eysh3 IO information
Eysh4	can perform professional shading|	Eysh4 IO information
Eysh5	can perform professional projection|	Eysh5 IO information
Eysh6	can perform professional zoom|	Eysh6 IO information
Eysh7	can perform professional pan|	Eysh7 IO information
Eysh8	can perform professional rotate|	Eysh8 IO information
Eysh9	can perform professional selection|	Eysh9 IO information
Eysh10	can perform professional selection|	Eysh10 IO information
Eysh11	add a 3D view to your application in minutes|	Eysh11 IO information
Eysh12	create professional reports|	Eysh12 IO information
Eysh13	Provides polygon based 3D modeling functions|	Eysh13 IO information

Component No. 4

Component's Name: VectorDraw Developer Framework (VDF)

Owner Organization Name: ISoft

Phone #: +92-51-9112655

Address: Street No. 48, H#595, F-6/3 Islamabad Pakistan

Mail Add: Info@isoft.com

Purpose: Graphics

Component's Functionalities:

F#	Functionalities	Input and Output Information
VDF1	Create 2D drawings|	VDF1 IO information
VDF2	Create 3D drawings|	VDF2 IO information
VDF3	Manage 2D drawings|	VDF3 IO information
VDF4	Manage 3D drawings|	VDF4 IO information
VDF5	print 2D and 3D drawings|	VDF5 IO information
VDF6	compatible with most common vector formats|	VDF6 IO information
VDF7	compatible with mostCAD objects|	VDF7 IO information
VDF8	Supports over 10 vector formats|	VDF8 IO information
VDF9	Supports many raster formats|	VDF9 IO information
VDF10	fully object oriented|	VDF10 IO information

After performing Extraction, we will have all the required information about the component. As shown above we have four different graphics component and have extracted our all required information about every component.

As shown above the *Component's Functionalities* portion of every component consists of two colors text. The *Green* color shows common functionalities which may have different description but will perform the same functionality, and the *other* color shows the unique functionality of that respective component.

5.1.2 Transformation

The information are coming from different sources i.e. form different *Owner Organizations,* and each of them will describe their component in their own way. As the components are for the same purpose but developed by different organization, so there is a maximum possibility that same functionalities and IO information will be described in

different way. In transformation we will transform those functionalities which perform same function but described in different way, into one function to remove redundant information and the same process we will also do with the IO information. After transformation each component's functionalities and IO have been presented by two different colors. Why two different colors have been used has been explained below in detail.

Green Color Description:

The Green color shows that the functionalities or IO information are common with other graphics component i.e. these functionalities or IO information are also provided by other components as shown in Fig. 18.

Other Color Description:

The color other than Green shows different functionalities or IO of that component.

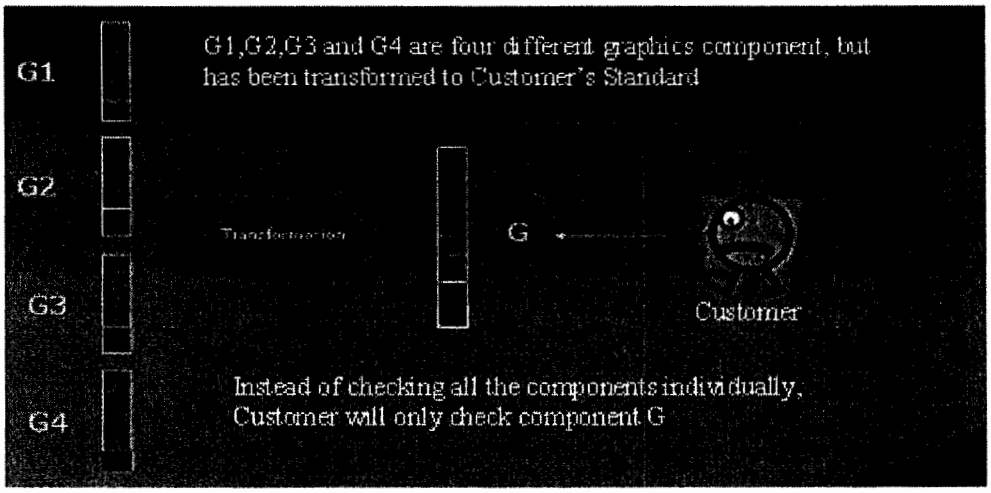


Fig. 18 Transformation

In the above figure the G1 represents Graphics Component No 1 i.e. *Eyeshot Fem*, G2 represents Graphics Component No 2 i.e. *Eyeshot Nurbs*, G3 represents Graphics Component No 3 i.e. *Eyeshot* and G4 represents Graphics Component No 4 i.e. *VectorDraw Developer Framework (VDF).*

After implementing transformation on G1, G2, G3 and G4, we will get component G with the following *Component Functionalities*.

Graphics Component G (after transformation)

Component's Functionalities:

F_id	F_Description
G_1	create a 3D model|
G_2	change a 3D model|
G_3	can be used with models imported from other programs|
G_4	can perform professional shading|
G_5	can perform professional projection|
G_6	can perform professional zoom|
G_7	can perform professional pan|
G_8	can perform professional rotate|
G_9	can perform professional selection|
G_10	add a 3D view to your application in minutes|
G_11	create professional reports|
G_12	Provides basic 2D Finite Element Analysis|
G_13	Provides NURBS based curv|
G_14	surface modeling functions|
G_15	Provides polygon based 3D modeling functions|
G_16	print 2D and 3D drawings|
G_17	compatible with most common vector formats|
G_18	compatible with mostCAD objects|
G_19	Supports over 10 vector formats|
G_20	Supports many raster formats|
G_21	fully object oriented|

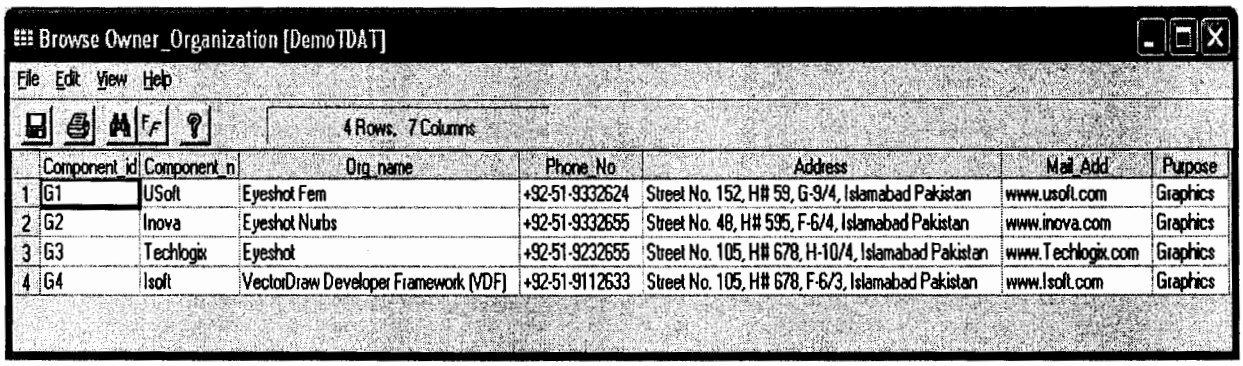
Component's IO:

IO_id	IO_Description
G_IO1	EyshFem1 IO information
G_IO2	EyshFem2 IO information
G_IO3	EyshFem3 IO information
G_IO4	EyshFem4 IO information
G_IO5	EyshFem5 IO information
G_IO6	EyshFem7 IO information
G_IO7	EyshFem9 IO information
G_IO8	EyshFem11 IO information
G_IO9	EyshFem6 IO information
G_IO10	EyshFem8 IO information
G_IO11	EyshFem10 IO information
G_IO12	EyshFem12 IO information
G_IO13	EyshNrbs1 IO information
G_IO14	EyshNrbs3 IO information
G_IO15	EyshNrbs6 IO information
G_IO16	EyshNrbs10 IO information
G_IO17	EyshNrbs12 IO information

IO_id	IO_Description
G_IO18	EyshNrbs13 IO information
G_IO19	Eysh13 IO information
G_IO20	VDF1 IO information
G_IO21	VDF6 IO information
G_IO22	VDF7 IO information
G_IO23	VDF8 IO information
G_IO24	VDF9 IO information
G_IO25	VDF10 IO information

5.1.3 Loading

In MSR process, after Extraction and Transformation the next step is to Load data to the repository. We will load information about the component i.e. Owner Organization name, Contact Detail, Component they are providing and Purpose into Owner_Organizaiton Table as shown below in the Fig. 25.

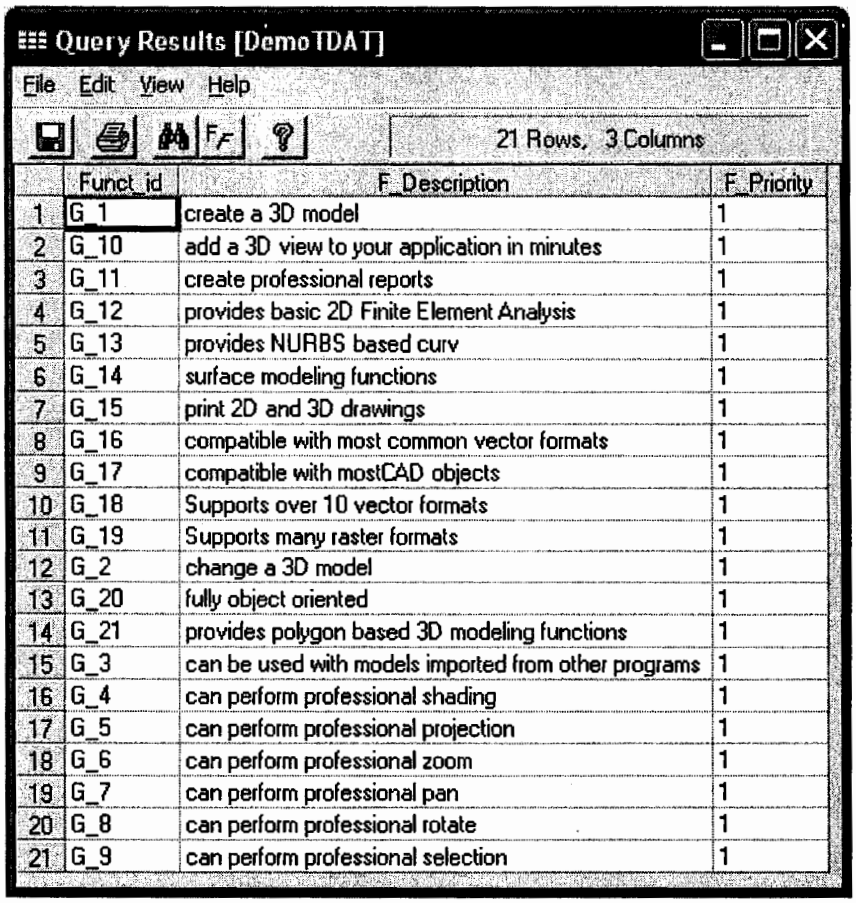


	Component_id	Component_n	Org_name	Phone_No	Address	Mail_Add	Purpose
1	G1	USoft	Eyeshot Fem	+92-51-9332624	Street No. 152, H# 59, G-9/4, Islamabad Pakistan	www.usoft.com	Graphics
2	G2	Inova	Eyeshot Nurbs	+92-51-9332655	Street No. 48, H# 595, F-6/4, Islamabad Pakistan	www.inova.com	Graphics
3	G3	Techlogix	Eyeshot	+92-51-9232655	Street No. 105, H# 678, H-10/4, Islamabad Pakistan	www.Techlogix.com	Graphics
4	G4	Isoft	VectorDraw Developer Framework (VDF)	+92-51-9112633	Street No. 105, H# 678, F-6/3, Islamabad Pakistan	www.Isoft.com	Graphics

Fig. 19 Owner_Organization Table

After loading information about Component's Owner organization, the next step is to load data about the component's functionalities and IO information. We will load the transformed functionalities and IO information i.e. of component G into a Functionality Table in as shown below in the Fig. 20 and Fig. 21.

The Functionality Table as shown below will consist of the Funct_id, F_Description and Priority fields. The Priority field shows priority of functionality in the Table. During loading we will keep priority value High i.e. '1' of all functionalities.



Query Results [DemoTDAT]

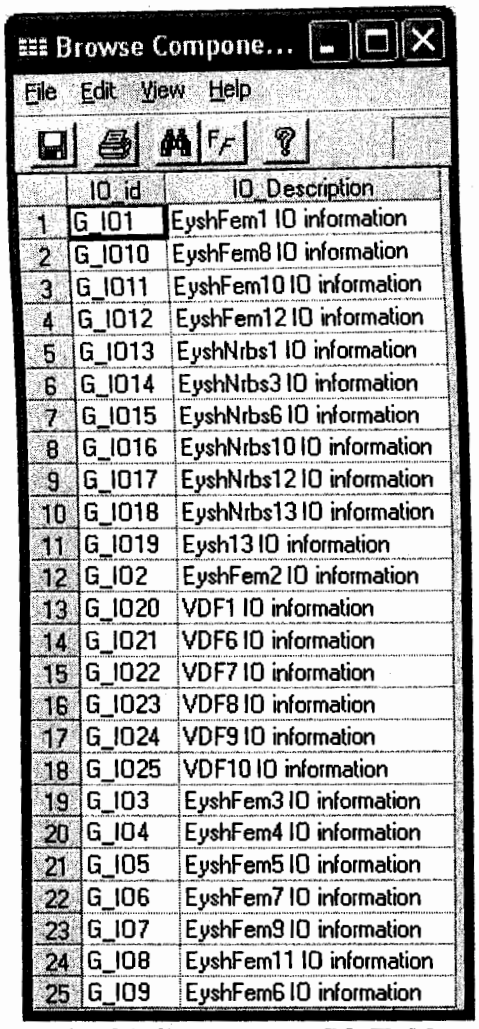
File Edit View Help

21 Rows, 3 Columns

	Funct_id	F_Description	F_Priority
1	G_1	create a 3D model	1
2	G_10	add a 3D view to your application in minutes	1
3	G_11	create professional reports	1
4	G_12	provides basic 2D Finite Element Analysis	1
5	G_13	provides NURBS based curv	1
6	G_14	surface modeling functions	1
7	G_15	print 2D and 3D drawings	1
8	G_16	compatible with most common vector formats	1
9	G_17	compatible with mostCAD objects	1
10	G_18	Supports over 10 vector formats	1
11	G_19	Supports many raster formats	1
12	G_2	change a 3D model	1
13	G_20	fully object oriented	1
14	G_21	provides polygon based 3D modeling functions	1
15	G_3	can be used with models imported from other programs	1
16	G_4	can perform professional shading	1
17	G_5	can perform professional projection	1
18	G_6	can perform professional zoom	1
19	G_7	can perform professional pan	1
20	G_8	can perform professional rotate	1
21	G_9	can perform professional selection	1

Fig.20 Functionality Table

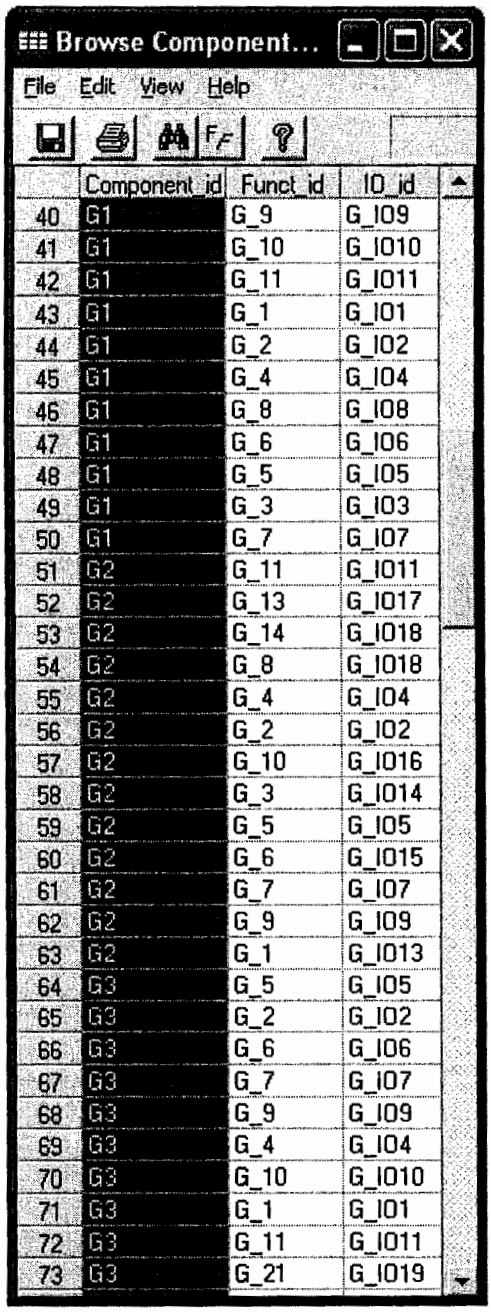
As shown above that all the functionalities have Input and Output information. After extracting and transforming information about the functionalities IO, we will load it into Component_IO table as shown below in Fig. 21.



	IO_id	IO_Description
1	G_IO1	EyshFem1 IO information
2	G_IO10	EyshFem8 IO information
3	G_IO11	EyshFem10 IO information
4	G_IO12	EyshFem12 IO information
5	G_IO13	EyshNrbs1 IO information
6	G_IO14	EyshNrbs3 IO information
7	G_IO15	EyshNrbs6 IO information
8	G_IO16	EyshNrbs10 IO information
9	G_IO17	EyshNrbs12 IO information
10	G_IO18	EyshNrbs13 IO information
11	G_IO19	Eysh13 IO information
12	G_IO2	EyshFem2 IO information
13	G_IO20	VDF1 IO information
14	G_IO21	VDF6 IO information
15	G_IO22	VDF7 IO information
16	G_IO23	VDF8 IO information
17	G_IO24	VDF9 IO information
18	G_IO25	VDF10 IO information
19	G_IO3	EyshFem3 IO information
20	G_IO4	EyshFem4 IO information
21	G_IO5	EyshFem5 IO information
22	G_IO6	EyshFem7 IO information
23	G_IO7	EyshFem9 IO information
24	G_IO8	EyshFem11 IO information
25	G_IO9	EyshFem6 IO information

Fig. 21 Component_IO Table

As shown above in the Owner_Organization Table and Functionality Table and Compnent_IO Table we have information about the Organization, Functionalities and the IO information of the all the components, but we don't know the which functionality is performed by which component and which IO information belongs to which functionality. Table Component_Fucnt_IO as shown in Fig. 22 shows the relation between the component, functionality and IO information from which we can easily determine that which functionality is belong to which component and have which IO information.



	Component_id	Funct_id	IO_id
40	G1	G_9	G_IO9
41	G1	G_10	G_IO10
42	G1	G_11	G_IO11
43	G1	G_1	G_IO1
44	G1	G_2	G_IO2
45	G1	G_4	G_IO4
46	G1	G_8	G_IO8
47	G1	G_6	G_IO6
48	G1	G_5	G_IO5
49	G1	G_3	G_IO3
50	G1	G_7	G_IO7
51	G2	G_11	G_IO11
52	G2	G_13	G_IO17
53	G2	G_14	G_IO18
54	G2	G_8	G_IO18
55	G2	G_4	G_IO4
56	G2	G_2	G_IO2
57	G2	G_10	G_IO16
58	G2	G_3	G_IO14
59	G2	G_5	G_IO5
60	G2	G_6	G_IO15
61	G2	G_7	G_IO7
62	G2	G_9	G_IO9
63	G2	G_1	G_IO13
64	G3	G_5	G_IO5
65	G3	G_2	G_IO2
66	G3	G_6	G_IO6
67	G3	G_7	G_IO7
68	G3	G_9	G_IO9
69	G3	G_4	G_IO4
70	G3	G_10	G_IO10
71	G3	G_1	G_IO1
72	G3	G_11	G_IO11
73	G3	G_21	G_IO19

Fig.22 Component_Funct_IO Table

5.1.4 Component Selection Process

In this step of the MSR process the Customer will input information about himself and also will set the priorities of the Functionalities and IO of that component.

5.1.4.1 Customer's Information

Customer will input the following information about him self and will be stored in Customer Table as shown below in Fig 30.

Cust_id: *C1*
Cust_name: *ASAD KHAN*
Cust_phone_number: 0334-5331364
Cust_Address: G-9/4 ISLAMABAD
Cust_Mail_Address: *asad.khan@gmail.com*

Browse Component_User [DemoTDAT]

File Edit View Help

4 Rows, 5 Columns

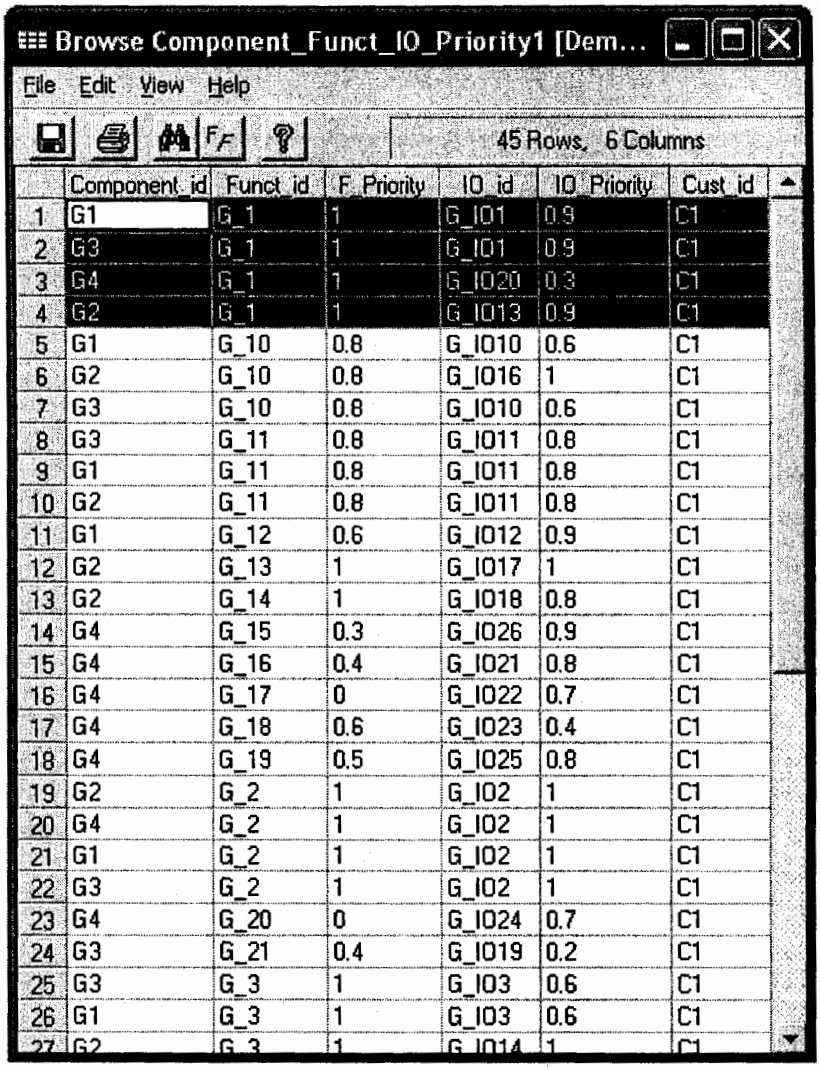
	User_id	User_name	User_phone_nu	User_Address	User_Mail_Address
1	C1	ASAD KHAN	0334-5331364	G-9/4 ISLAMABAD	asad.khan@gmail.com
2	U1	ALI AHMAD	0333765482	ISLAMABAD, PAKISTAN	ali.ahmad@gmail.com
3	U2	KASHIF SAEED	0333765477	ISLAMABAD, PAKISTAN	kashif.saeed@gmail.com
4	U3	Naveed Ashraf	03468002883	Thana Malakand Agency, Pakistan	Naveed.Ashraf@gmail.com

Fig. 23: Customer Table

Changing Priorities:

Component selection is the last process of the MSR approach. After performing Extraction, Transformation and loading we will have all the required information about the reusable component is MSR repository. To select the best required component, the Customer will set priorities of the two fields, i.e. F_Priority and IO_Priority. F_Priority will show that how much that functionality is required by the Customer, and IO_Priority which IO information how much qualifies Customer's requirements. The following

figure shows information about the graphics component which has been filled by Customer C1.



	Component_id	Funct_id	F_Priority	IO_id	IO_Priority	Cust_id
1	G1	G_1	1	G_IO1	0.9	C1
2	G3	G_1	1	G_IO1	0.9	C1
3	G4	G_1	1	G_IO20	0.3	C1
4	G2	G_1	1	G_IO13	0.9	C1
5	G1	G_10	0.8	G_IO10	0.6	C1
6	G2	G_10	0.8	G_IO16	1	C1
7	G3	G_10	0.8	G_IO10	0.6	C1
8	G3	G_11	0.8	G_IO11	0.8	C1
9	G1	G_11	0.8	G_IO11	0.8	C1
10	G2	G_11	0.8	G_IO11	0.8	C1
11	G1	G_12	0.6	G_IO12	0.9	C1
12	G2	G_13	1	G_IO17	1	C1
13	G2	G_14	1	G_IO18	0.8	C1
14	G4	G_15	0.3	G_IO26	0.9	C1
15	G4	G_16	0.4	G_IO21	0.8	C1
16	G4	G_17	0	G_IO22	0.7	C1
17	G4	G_18	0.6	G_IO23	0.4	C1
18	G4	G_19	0.5	G_IO25	0.8	C1
19	G2	G_2	1	G_IO2	1	C1
20	G4	G_2	1	G_IO2	1	C1
21	G1	G_2	1	G_IO2	1	C1
22	G3	G_2	1	G_IO2	1	C1
23	G4	G_20	0	G_IO24	0.7	C1
24	G3	G_21	0.4	G_IO19	0.2	C1
25	G3	G_3	1	G_IO3	0.6	C1
26	G1	G_3	1	G_IO3	0.6	C1
27	G2	G_3	1	G_IO14	1	C1

Fig. 24 Component_Funct_IO_Priority Table

After changing the values of the priorities and submitting it. To determine that how much each component qualifies the functional requirement of the Customer, we will apply eq. 1 on it and will retrieve qualification values for every component as shown below in Table. 1.

Table 1: Component's Qualification information

Component_id	Funct_id	F_Priority	IO_id	IO_Priority	Customer_id	Qualification of Each Functionality FPi * IOi	$\sum_{i=1...N}$ (FPi * IOi)	$Q_c = \sum_{i=1...N}$ (FPi * IOi)/ T * 100	
G1	G_1	1	G_IO1	0.9	C1	0.9			
G1	G_10	0.8	G_IO10	0.6	C1	0.48			
G1	G_11	0.8	G_IO11	0.8	C1	0.64			N=12 T=13
G1	G_12	0.6	G_IO12	0.9	C1	0.54			
G1	G_2	1	G_IO2	1	C1	1			
G1	G_3	1	G_IO3	0.6	C1	0.6			
G1	G_4	0.8	G_IO4	0.7	C1	0.56			
G1	G_5	0.6	G_IO5	0.9	C1	0.54			
G1	G_6	1	G_IO6	0.8	C1	0.8			
G1	G_7	0.6	G_IO7	0.9	C1	0.54			
G1	G_8	0.9	G_IO8	0.8	C1	0.72			
G1	G_9	0.7	G_IO9	0.3	C1	0.21	7.74	59.53846154	
G2	G_1	1	G_IO13	0.9	C1	0.9			
G2	G_10	0.8	G_IO16	1	C1	0.8			N=13 T=13
G2	G_11	0.8	G_IO11	0.8	C1	0.64			
G2	G_13	1	G_IO17	1	C1	1			
G2	G_14	1	G_IO18	0.8	C1	0.8			
G2	G_2	1	G_IO2	1	C1	1			
G2	G_3	1	G_IO14	1	C1	1			
G2	G_4	0.8	G_IO4	0.7	C1	0.56			
G2	G_5	0.6	G_IO5	0.9	C1	0.54			
G2	G_6	1	G_IO15	0.9	C1	0.9			
G2	G_7	0.6	G_IO7	0.9	C1	0.54			
G2	G_8	0.9	G_IO18	0.8	C1	0.72			
G2	G_9	0.7	G_IO9	0.3	C1	0.21	9.82	75.53846154	
G3	G_1	1	G_IO1	0.9	C1	0.9			
G3	G_10	0.8	G_IO10	0.6	C1	0.48			N=12 T=13
G3	G_11	0.8	G_IO11	0.8	C1	0.64			
G3	G_2	1	G_IO2	1	C1	1			
G3	G_21	0.4	G_IO19	0.2	C1	0.08			
G3	G_3	1	G_IO3	0.6	C1	0.6			
G3	G_4	0.8	G_IO4	0.7	C1	0.56			
G3	G_5	0.6	G_IO5	0.9	C1	0.54			
G3	G_6	1	G_IO6	0.8	C1	0.8			
G3	G_7	0.6	G_IO7	0.9	C1	0.54			
G3	G_8	0.9	G_IO8	0.8	C1	0.72	7.79	59.92307692	
G3	G_9	0.7	G_IO9	0.3	C1	0.21			
G4	G_1	1	G_IO20	0.3	C1	0.3			
G4	G_15	0.3	G_IO26	0.9	C1	0.27			N=8 T=13
G4	G_16	0.4	G_IO21	0.8	C1	0.32			
G4	G_17	0	G_IO22	0.7	C1	0			
G4	G_18	0.6	G_IO23	0.4	C1	0.24			
G4	G_19	0.5	G_IO25	0.8	C1	0.4			

G4	G_2	1	G_I02	1	C1	1		
G4	G_20	0	G_I024	0.7	C1	0	2.53	19.46153846

G2 Qualifies Customer's Requirements up to **75.53846154 %**

G3 Qualifies Customer's Requirements up to **59.92307692 %**

G1 Qualifies Customer's Requirements up to **59.53846154 %**

G4 Qualifies Customer's Requirements up to **19.46153846 %**

So the bold highlighted **75.53846154 %** shows that G2 is the best qualifying component for Customer 'C1'.

5.2 *Case 2, Selection of Barcode Component on the basis of Functional Requirements*

In Case 2, we are going to select the Barcode component among the different available components. The Customer will select his best required component among the available three different barcode components i.e. Barcode Professional [25], Barcode for Java 3.0 [26] and Aspose.Barcode [27]. In Case 2 we have omitted some details as they are already given in Case 1.

To implement the MSR approach we have taken information about the three Barcode components from different libraries.

Barcode Professional [25]

1. Barcode image generation and printing support for 53 Linear/1D & Postal Barcode Simbologies
2. Barcode image generation and printing support for 9 2D Barcode Symbologies
3. Automatically computes checksum or check digits for all Symbologies
4. Automatically validates the value to encode
5. Written entirely in managed C#
6. Strong named assemblies for .NET Framework 1.0, 1.1 & 2.0
7. Can be used in any .NET Language such as C#, VB.NET, Managed C++, etc.
8. Save barcodes in image files as well as byte stream objects
9. GIF, BMP, PNG, JPEG/JPG & TIFF barcode image formats support
10. Monochrome (Black and White 1 bit per pixel) barcode image generation support
11. Barcode rotation support (0, 90, 180, 270 degrees)
12. Binary, SOAP, and XML serialization support
13. ASP.NET Web Services support
14. Windows Forms Data-Binding support
15. Crystal Reports for .NET support
16. Visual Studio 2005/2008 ReportViewer Local Reports (RDLC) support
17. Full and flexible printing support through PrintDocument class

Barcode for Java 3.0 [26]

1. Support 24 linear (1D) & 2D Barcode Symbologies.
2. Can save the barcode in an **image file** as well as in a **stream object**.
3. GIF, and JPEG/JPG image formats support.
4. Automatically computes the checksum or check digits.
5. **Jasper Reports** support
6. Support JDK 1.4.2 and later version

Aspose.BarCode [27]

Aspose.BarCode is a Java component for generation & recognition of 1D & 2D barcodes to support Java and web applications. Main features include: barcode size control, color settings, rotation angle & captions etc. It supports 20 barcode symbologies like MSI, QR, PDF417, Aztec, Code128, Planet, Code11, EAN128, Codabar, Postnet, UPCA, UPCE etc and also supports Image output in JPEG, GIF, PNG, BMP & JPG formats. Render barcodes to Images, printers, HTTP servlet response & graphical objects too.

If we look at the above three components, they are all for the same purpose i.e. Barcode components, but are developed by different *Owner Organizations*. Every Owner Organization have their own standard and have explained there components in different ways. Also if we look at the above components all of them have some functionality in common, but have been written in different way.

To make available all the components in a single repository i.e. MSR repository, MSR process consists of the following steps.

5.2.1 Extraction

We will extract the following information about the reusable component.

Component's Name
Owner Organization Name
Purpose
Component's Functionalities

In case 2 we have the following three Barcode components from different Owner Organizations.

Component No 1

Owner Organization Name:	Neodynamic
Component's Name:	**Barcode Professional**
Phone #:	+92 51 432324
Address:	Neodynamics blue area Islamabad
Mail Add:	http://www.neodynamic.com
Purpose:	Barcode Component

Component's Functionalities:

BrcP1	Barcode image generation and printing support for 53 Linear/1D & Postal Barcode Simbologies	BrcP1 IO information
BrcP2	Barcode image generation and printing support for 9 2D Barcode Symbologies	BrcP2 IO information
BrcP3	Automatically computes checksum or check digits for all Symbologies	BrcP3 IO information
BrcP4	Automatically validates the value to encode	BrcP4 IO information
BrcP5	Written entirely in managed C#	BrcP5 IO information
BrcP6	Strong named assemblies for .NET Framework 1.0, 1.1 & 2.0	BrcP6 IO information
BrcP7	Can be used in any .NET Language such as C#, VB.NET, Managed C++, etc.	BrcP7 IO information
BrcP8	Save barcodes in image files as well as byte stream objects	BrcP8 IO information
BrcP9	GIF, BMP, PNG, JPEG/JPG & TIFF barcode image formats support	BrcP9 IO information
BrcP10	Monochrome (Black and White 1 bit per pixel) barcode image generation support	BrcP10 IO information
BrcP11	Barcode rotation support (0, 90, 180, 270 degrees)	BrcP11 IO information
BrcP12	Binary, SOAP, and XML serialization support	BrcP12 IO information
BrcP13	ASP.NET Web Services support	BrcP13 IO information
BrcP14	Windows Forms Data-Binding support	BrcP14 IO information
BrcP15	Crystal Reports for .NET support	BrcP15 IO information
BrcP16	Visual Studio 2005/2008 ReportViewer Local Reports (RDLC) support	BrcP16 IO information
BrcP17	Full and flexible printing support through PrintDocument class	BrcP17 IO information

Component 2

Owner Organization Name: BarcodeLib

Component's Name: **Barcode for Jave 3.0**

Phone #: +9251332456

Address: Blue Area Islamabad

Mail Add: www.Barcodelib.com

Purpose: Barcode Component

Component's Functionalities:

BrcJ1	Support 24 linear (1D) & 2D Barcode Symbologies.	CBrcJ1 IO information
BrcJ2	Can save the barcode in an **image file** as well as in a **stream object**.	BrcJ2 IO information
BrcJ3	GIF, and JPEG/JPG image formats support.	BrcJ3 IO information
BrcJ4	Automatically computes the checksum or check digits.	BrcJ4 IO information
BrcJ5	**Jasper Reports** support	BrcJ5 IO information
BrcJ6	Support JDK 1.4.2 and later version	BrcJ6 IO information

Component 3

Owner Organization Name: ASPOSE

Component's Name: **AsposeBarcode**

Phone #: +9251443678

Address: Blue Area Islamabad

Mail Add: http://www.devdirect.com/java/BARCO

Purpose: Barcode Component

Component's Functionalities:

BrcA1	generation & recognition of 1D & 2D barcodes to support Java and web applications	BrcA1IO information
BrcA2	barcode size control	BrcA2IO information
BrcA3	barcode color settings	BrcA3IO information
BrcA4	barcode rotation angle	BrcA4IO information
BrcA5	barcode captions Control	BrcA5IO information
BrcA6	supports 20 barcode symbologies like MSI, QR, PDF417, Aztec, Code128, Planet, Code11, EAN128, Codabar,	BrcA6IO information

	Postnet, UPCA, UPCE etc	
BrcA7	supports Image output in JPEG, GIF, PNG, BMP & JPG formats	BrcA7IO information
BrcA8	Render barcodes to Images, printers, HTTP servlet response & graphical objects too.	BrcA8IO information

5.2.2 Transformation

If we look at the three different barcode components build for the same purpose by different owner organization then we will find that all the components have some functionalities in common. But every Owner Organization has stated the features and functionalities of their components in their own way. In MSR instead of loading the common functionalities again and again, we will load them just once accompanied with all the different functionalities.

After transformation and Loading all the three different barcode components will be available in a single location and instead of searching them in three different repositories the Customer will just visit a single MSR repository. After performing transformation we will load the following information about the reusable component into the repository.

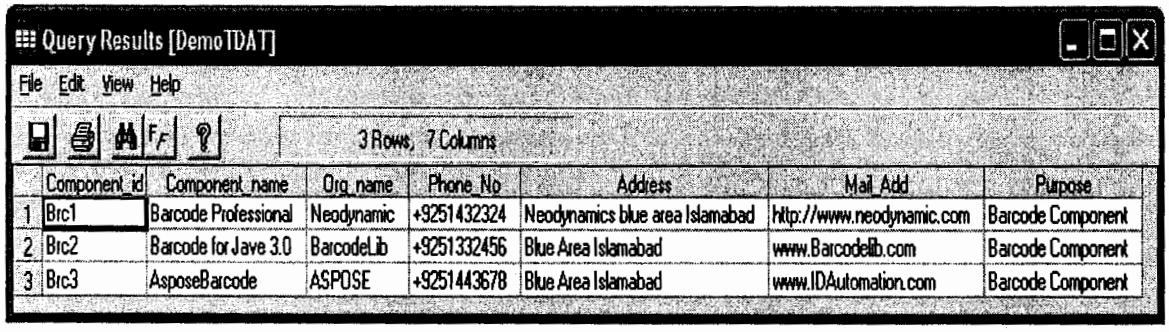
FBrc1	Barcode image generation and printing support for 53 Linear/1D & Postal Barcode Simbologies
FBrc2	Support 24 linear (1D) & 2D Barcode Symbologies
FBrc3	Automatically computes checksum or check digits for all Symbologies
FBrc4	Save barcodes in image files as well as byte stream objects
FBrc5	GIF, BMP, PNG, JPEG/JPG & TIFF barcode image formats support
FBrc6	Printing support for 2D barcode Symboligies
FBrc7	Automatically validates the value to encode
FBrc8	Written entirely in managed C#
FBrc9	Strong named assemblies for .NET Framework 1.0, 1.1 & 2.0
FBrc10	Save barcodes in image files as well as byte stream objects
FBrc11	GIF, BMP, PNG, JPEG/JPG & TIFF barcode image formats support
FBrc12	Monochrome (Black and White 1 bit per pixel) barcode image generation support
FBrc13	Barcode rotation support (0, 90, 180, 270 degrees)
FBrc14	Binary, SOAP, and XML serialization support
FBrc15	ASP.NET Web Services support
FBrc16	Windows Forms Data-Binding support

FBrc17	Crystal Reports for .NET support
FBrc18	Visual Studio 2005/2008 ReportViewer Local Reports (RDLC) support
FBrc19	Full and flexible printing support through PrintDocument class
FBrc20	Jasper Reports support
FBrc21	Support JDK 1.4.2 and later version
FBrc22	barcode size control
FBrc23	barcode color settings
FBrc24	barcode rotation angle
FBrc25	barcode captions Control
FBrc26	Render barcodes to Images, printers, HTTP servlet response & graphical objects too.

5.2.3 Loading

After Extraction and Transformation the next step is to load the data to the repository so that to make it available for the Customers.

Initially we will load information about the Owner Organizations as shown below in Fig 25.



	Component_id	Component_name	Org_name	Phone_No	Address	Mail_Add	Purpose
1	Brc1	Barcode Professional	Neodynamic	+9251432324	Neodynamics blue area Islamabad	http://www.neodynamic.com	Barcode Component
2	Brc2	Barcode for Jave 3.0	BarcodeLib	+9251332456	Blue Area Islamabad	www.Barcodelib.com	Barcode Component
3	Brc3	AsposeBarcode	ASPOSE	+9251443678	Blue Area Islamabad	www.IDAutomation.com	Barcode Component

Fig. 25 Owner_Organization Table

After loading information about Component's Owner organization, the next step is to load data about the component's functionalities and IO information. We will load the transformed functionalities and IO information into a Functionality Table in as shown below in the Fig. 26 and Fig. 27.

The Functionality Table as shown below will consist of the Funct_id, F_Description and Priority fields. The Priority field shows priority of functionality in the Table. During loading we will keep priority value High i.e. '1' of all functionalities.

Query Results [DemoTDAT]

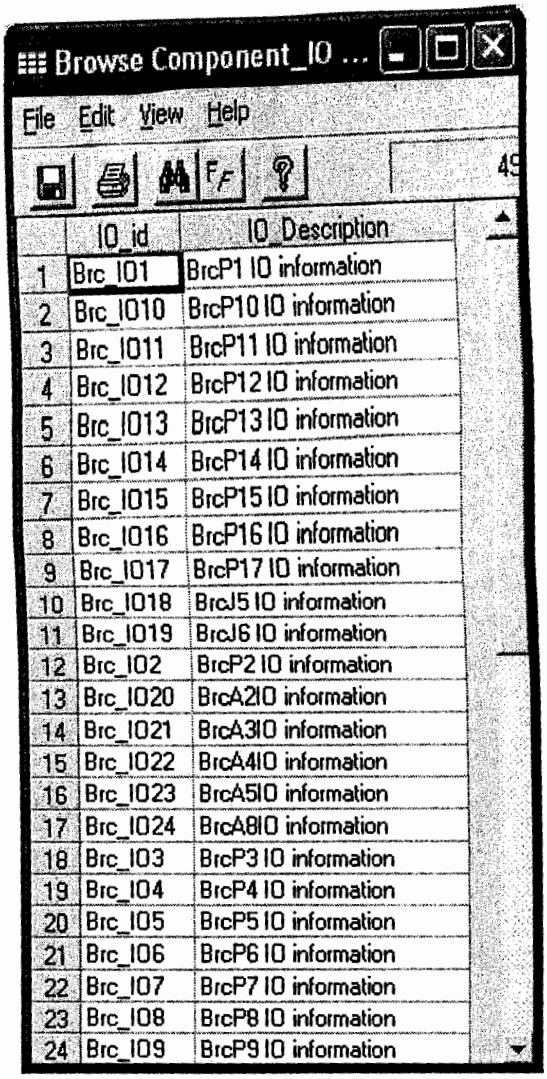
File Edit View Help

26 Rows, 3 Columns

	Funct_id	F_Description	F_Priority
1	FBrc1	Barcode image generation and printing support for 53 Linear/1D & Postal Barcode Simbologies	1
2	FBrc10	Save barcodes in image files as well as byte stream objects	1
3	FBrc11	GIF, BMP, PNG, JPEG/JPG & TIFF barcode image formats support	1
4	FBrc12	Monochrome (Black and White 1 bit per pixel) barcode image generation support	1
5	FBrc13	Barcode rotation support (0, 90, 180, 270 degrees)	1
6	FBrc14	Binary, SOAP, and XML serialization support	1
7	FBrc15	ASP.NET Web Services support	1
8	FBrc16	Windows Forms Data-Binding support	1
9	FBrc17	Crystal Reports for .NET support	1
10	FBrc18	Visual Studio 2005/2008 ReportViewer Local Reports (RDLC) support	1
11	FBrc19	Full and flexible printing support through PrintDocument class	1
12	FBrc2	Support 24 linear (1D) & 2D Barcode Symbologies	1
13	FBrc20	Jasper Reports support	1
14	FBrc21	Support JDK 1.4.2 and later version	1
15	FBrc22	barcode size control	1
16	FBrc23	barcode color settings	1
17	FBrc24	barcode rotation angle	1
18	FBrc25	barcode captions Control	1
19	FBrc26	Render barcodes to Images, printers, HTTP servlet response & graphical objects too	1
20	FBrc3	Automatically computes checksum or check digits for all Symbologies	1
21	FBrc4	Save barcodes in image files as well as byte stream objects	1
22	FBrc5	GIF, BMP, PNG, JPEG/JPG & TIFF barcode image formats support	1
23	FBrc6	Printing support for 2D barcode Symboligies	1
24	FBrc7	Automatically validates the value to encode	1
25	FBrc8	Written entirely in managed C#	1
26	FBrc9	Strong named assemblies for .NET Framework 1.0, 1.1 & 2.0	1

Fig. 26 Functionality Table

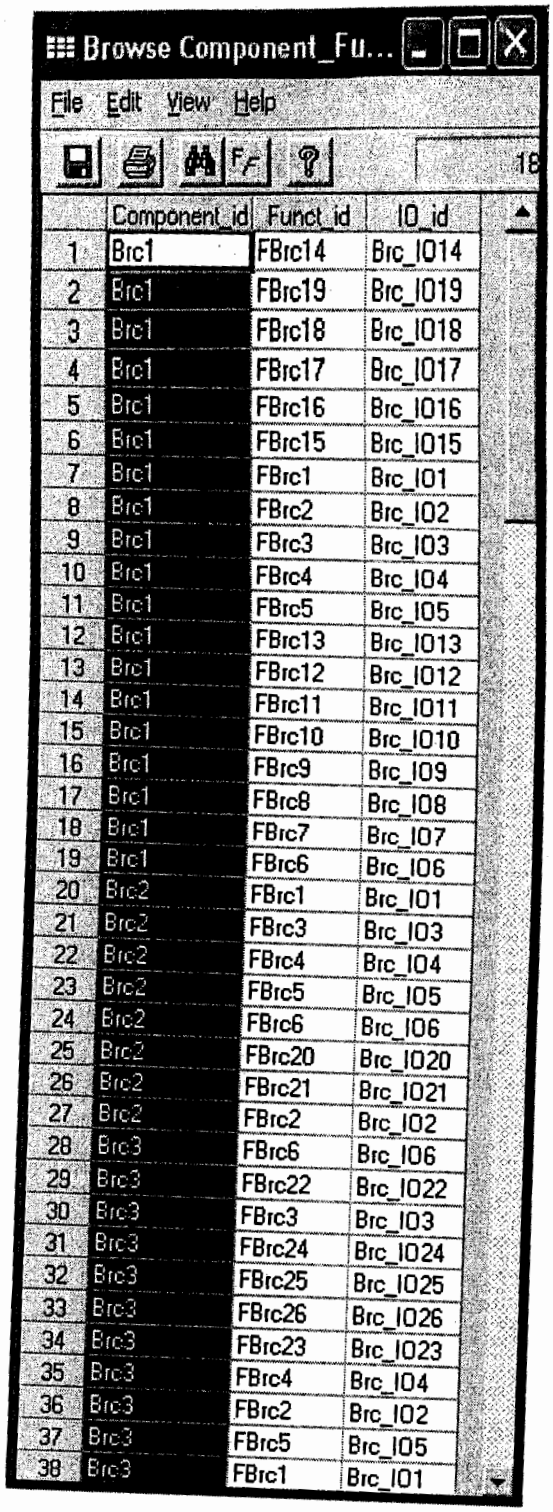
As shown above that all the functionalities have Input and Output information. After extracting and transforming information about the functionalities IO, we will load it into Component_IO table as shown below in Fig. 27.



	IO_id	IO_Description
1	Brc_IO1	BrcP1 IO information
2	Brc_IO10	BrcP10 IO information
3	Brc_IO11	BrcP11 IO information
4	Brc_IO12	BrcP12 IO information
5	Brc_IO13	BrcP13 IO information
6	Brc_IO14	BrcP14 IO information
7	Brc_IO15	BrcP15 IO information
8	Brc_IO16	BrcP16 IO information
9	Brc_IO17	BrcP17 IO information
10	Brc_IO18	BrcJ5 IO information
11	Brc_IO19	BrcJ6 IO information
12	Brc_IO2	BrcP2 IO information
13	Brc_IO20	BrcA2IO information
14	Brc_IO21	BrcA3IO information
15	Brc_IO22	BrcA4IO information
16	Brc_IO23	BrcA5IO information
17	Brc_IO24	BrcA8IO information
18	Brc_IO3	BrcP3 IO information
19	Brc_IO4	BrcP4 IO information
20	Brc_IO5	BrcP5 IO information
21	Brc_IO6	BrcP6 IO information
22	Brc_IO7	BrcP7 IO information
23	Brc_IO8	BrcP8 IO information
24	Brc_IO9	BrcP9 IO information

Fig. 27 Component_IO Table

After loading information to Owner_Organization, Functionality and Component_IO Table, we have information about the Component's functionality, IO and Organization, but we don't know that which functionality is performed by which component and which IO information is associated to which functionality. To solve this problem we will load information into Component_Funct_IO table which shows us the relationship among the Component, functionality and IO information as shown below in Fig. 28.



	Component_id	Funct_id	IO_id
1	Brc1	FBrc14	Brc_IO14
2	Brc1	FBrc19	Brc_IO19
3	Brc1	FBrc18	Brc_IO18
4	Brc1	FBrc17	Brc_IO17
5	Brc1	FBrc16	Brc_IO16
6	Brc1	FBrc15	Brc_IO15
7	Brc1	FBrc1	Brc_IO1
8	Brc1	FBrc2	Brc_IO2
9	Brc1	FBrc3	Brc_IO3
10	Brc1	FBrc4	Brc_IO4
11	Brc1	FBrc5	Brc_IO5
12	Brc1	FBrc13	Brc_IO13
13	Brc1	FBrc12	Brc_IO12
14	Brc1	FBrc11	Brc_IO11
15	Brc1	FBrc10	Brc_IO10
16	Brc1	FBrc9	Brc_IO9
17	Brc1	FBrc8	Brc_IO8
18	Brc1	FBrc7	Brc_IO7
19	Brc1	FBrc6	Brc_IO6
20	Brc2	FBrc1	Brc_IO1
21	Brc2	FBrc3	Brc_IO3
22	Brc2	FBrc4	Brc_IO4
23	Brc2	FBrc5	Brc_IO5
24	Brc2	FBrc6	Brc_IO6
25	Brc2	FBrc20	Brc_IO20
26	Brc2	FBrc21	Brc_IO21
27	Brc2	FBrc2	Brc_IO2
28	Brc3	FBrc6	Brc_IO6
29	Brc3	FBrc22	Brc_IO22
30	Brc3	FBrc3	Brc_IO3
31	Brc3	FBrc24	Brc_IO24
32	Brc3	FBrc25	Brc_IO25
33	Brc3	FBrc26	Brc_IO26
34	Brc3	FBrc23	Brc_IO23
35	Brc3	FBrc4	Brc_IO4
36	Brc3	FBrc2	Brc_IO2
37	Brc3	FBrc5	Brc_IO5
38	Brc3	FBrc1	Brc_IO1

Fig. 28 Component_Funct_IO Table

5.2.4 Component Selection Process

In this step of the MSR process the Customer will input information about himself and also will set the priorities of the Functionalities and IO of that component.

5.2.4.1 Customer's Information

Customer will input the following information about him self and will be stored in Customer Table as shown below in Fig 29.

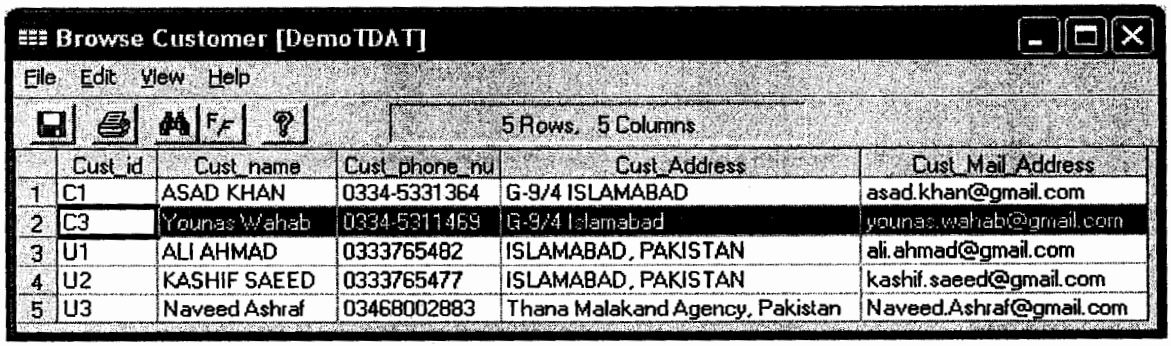
Cust_id: *C3*

Cust_name: *YOUNAS WAHAB*

Cust_phone_number: 0334-5311469

Cust_Address: G-9/4 ISLAMABAD

Cust_Mail_Address: *younas.wahab@gmail.com*

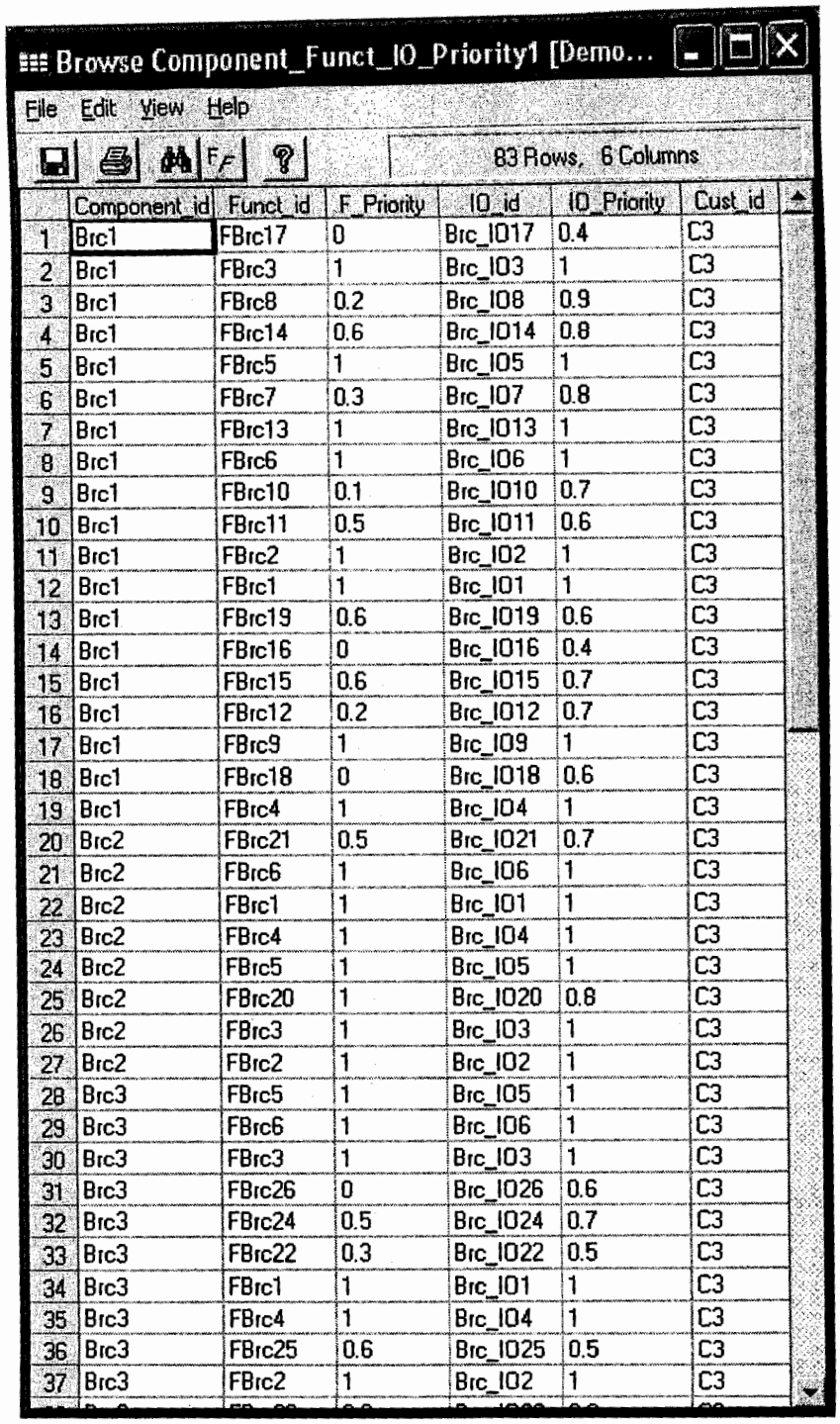


	Cust_id	Cust_name	Cust_phone_nu	Cust_Address	Cust_Mail_Address
1	C1	ASAD KHAN	0334-5331364	G-9/4 ISLAMABAD	asad.khan@gmail.com
2	C3	Younas Wahab	0334-5311469	G-9/4 Islamabad	younas.wahab@gmail.com
3	U1	ALI AHMAD	0333765482	ISLAMABAD, PAKISTAN	ali.ahmad@gmail.com
4	U2	KASHIF SAEED	0333765477	ISLAMABAD, PAKISTAN	kashif.saeed@gmail.com
5	U3	Naveed Ashraf	03468002883	Thana Malakand Agency, Pakistan	Naveed.Ashraf@gmail.com

Fig. 29 Customer Table

Changing Priorities:

Component selection is the last process of the MSR approach. After performing Extraction, Transformation and loading we will have all the required information about the reusable component is MSR repository. To select the best required component, the Customer will set priorities of the two fields, i.e. F_Priority and IO_Priority. F_Priority will show that how much that functionality is required by the Customer, and IO_Priority which IO information how much qualifies Customer's requirements. The following Fig. 30 shows information about the graphics component which has been filled by Customer C3.



	Component_id	Funct_id	F_Priority	IO_id	IO_Priority	Cust_id
1	Brc1	FBrc17	0	Brc_IO17	0.4	C3
2	Brc1	FBrc3	1	Brc_IO3	1	C3
3	Brc1	FBrc8	0.2	Brc_IO8	0.9	C3
4	Brc1	FBrc14	0.6	Brc_IO14	0.8	C3
5	Brc1	FBrc5	1	Brc_IO5	1	C3
6	Brc1	FBrc7	0.3	Brc_IO7	0.8	C3
7	Brc1	FBrc13	1	Brc_IO13	1	C3
8	Brc1	FBrc6	1	Brc_IO6	1	C3
9	Brc1	FBrc10	0.1	Brc_IO10	0.7	C3
10	Brc1	FBrc11	0.5	Brc_IO11	0.6	C3
11	Brc1	FBrc2	1	Brc_IO2	1	C3
12	Brc1	FBrc1	1	Brc_IO1	1	C3
13	Brc1	FBrc19	0.6	Brc_IO19	0.6	C3
14	Brc1	FBrc16	0	Brc_IO16	0.4	C3
15	Brc1	FBrc15	0.6	Brc_IO15	0.7	C3
16	Brc1	FBrc12	0.2	Brc_IO12	0.7	C3
17	Brc1	FBrc9	1	Brc_IO9	1	C3
18	Brc1	FBrc18	0	Brc_IO18	0.6	C3
19	Brc1	FBrc4	1	Brc_IO4	1	C3
20	Brc2	FBrc21	0.5	Brc_IO21	0.7	C3
21	Brc2	FBrc6	1	Brc_IO6	1	C3
22	Brc2	FBrc1	1	Brc_IO1	1	C3
23	Brc2	FBrc4	1	Brc_IO4	1	C3
24	Brc2	FBrc5	1	Brc_IO5	1	C3
25	Brc2	FBrc20	1	Brc_IO20	0.8	C3
26	Brc2	FBrc3	1	Brc_IO3	1	C3
27	Brc2	FBrc2	1	Brc_IO2	1	C3
28	Brc3	FBrc5	1	Brc_IO5	1	C3
29	Brc3	FBrc6	1	Brc_IO6	1	C3
30	Brc3	FBrc3	1	Brc_IO3	1	C3
31	Brc3	FBrc26	0	Brc_IO26	0.6	C3
32	Brc3	FBrc24	0.5	Brc_IO24	0.7	C3
33	Brc3	FBrc22	0.3	Brc_IO22	0.5	C3
34	Brc3	FBrc1	1	Brc_IO1	1	C3
35	Brc3	FBrc4	1	Brc_IO4	1	C3
36	Brc3	FBrc25	0.6	Brc_IO25	0.5	C3
37	Brc3	FBrc2	1	Brc_IO2	1	C3

Fig 30 Component_Funct_IO_Priority Table

After changing the values of the priorities and submitting it. To determine that how much each component qualifies the functional requirement of the Customer, we will apply eq. 1 on it and will retrieve qualification values for every component as shown below in Table. 2.

Table 2: Component's Qualification information

Component_id	Funct_id	F_Priority	IO_id	IO_Priority	Customer_id	Qualification of Each Functionality FPi * IOi	$\sum_{i=1...N}$ (FPi * IOi)	$Q_c = \sum_{i=1...N}$ (FPi * IOi)/ T * 100	
Brc1	FBrc17	0	Brc_IO17	0.4	C3	0			
Brc1	FBrc3	1	Brc_IO3	1	C3	1			N=19
Brc1	FBrc8	0.2	Brc_IO8	0.9	C3	0.18			T=26
Brc1	FBrc14	0.6	Brc_IO14	0.8	C3	0.48			
Brc1	FBrc5	1	Brc_IO5	1	C3	1			
Brc1	FBrc7	0.3	Brc_IO7	0.8	C3	0.24			
Brc1	FBrc13	1	Brc_IO13	1	C3	1			
Brc1	FBrc6	1	Brc_IO6	1	C3	1			
Brc1	FBrc10	0.1	Brc_IO10	0.7	C3	0.07			
Brc1	FBrc11	0.5	Brc_IO11	0.6	C3	0.3			
Brc1	FBrc2	1	Brc_IO2	1	C3	1			
Brc1	FBrc1	1	Brc_IO1	1	C3	1			
Brc1	FBrc19	0.6	Brc_IO19	0.6	C3	0.36			
Brc1	FBrc16	0	Brc_IO16	0.4	C3	0			
Brc1	FBrc15	0.6	Brc_IO15	0.7	C3	0.42			
Brc1	FBrc12	0.2	Brc_IO12	0.7	C3	0.14			
Brc1	FBrc9	1	Brc_IO9	1	C3	1			
Brc1	FBrc18	0	Brc_IO18	0.6	C3	0			
Brc1	FBrc4	1	Brc_IO4	1	C3	1	11.19	43.03846154	
Brc2	FBrc2	1	Brc_IO2	1	C3	1			
Brc2	FBrc21	0.5	Brc_IO21	0.7	C3	0.35			N=8
Brc2	FBrc6	1	Brc_IO6	1	C3	1			T=26
Brc2	FBrc1	1	Brc_IO1	1	C3	1			
Brc2	FBrc4	1	Brc_IO4	1	C3	1			
Brc2	FBrc5	1	Brc_IO5	1	C3	1			
Brc2	FBrc20	1	Brc_IO20	0.8	C3	0.8			
Brc2	FBrc3	1	Brc_IO3	1	C3	1	8.15	31.34615385	
Brc3	FBrc5	1	Brc_IO5	1	C3	1			
Brc3	FBrc6	1	Brc_IO6	1	C3	1			N=11
Brc3	FBrc3	1	Brc_IO3	1	C3	1			T=26
Brc3	FBrc26	0	Brc_IO26	0.6	C3	0			
Brc3	FBrc24	0.5	Brc_IO24	0.7	C3	0.35			
Brc3	FBrc22	0.3	Brc_IO22	0.5	C3	0.15			
Brc3	FBrc4	1	Brc_IO4	1	C3	1			
Brc3	FBrc25	0.6	Brc_IO25	0.5	C3	0.3			
Brc3	FBrc2	1	Brc_IO2	1	C3	1			
Brc3	FBrc23	0.6	Brc_IO23	0.6	C1	0.36			
Brc3	FBrc1	1	Brc_IO1	1	C3	1	8.16	31.38461538	

Brc1 Qualifies Customer's Requirements up to	**43.03846154 %**
Brc3 Qualifies Customer's Requirements up to	**31.38461538 %**
Brc2G1 Qualifies Customer's Requirements up to	**31.34615385 %**

So the bold highlighted **43.03846154** % shows that Brc1 is the best qualifying component for Customer 'C3'.

Chapter 6

Conclusion

6 Conclusion

MSR approach provides a single repository where component from all other reuse libraries have been Extracted, Transformed and Loaded. Components in the MSR library are from different *Owner Organizations* and every Owner Organization has their own way of describing the reusable components. If *Customers* want to select their best qualifying component based on functional requirements of the component then it is required to compare the components with each other in the MSR library, but the problem is that components are described in different way as developed by different organization and is not possible to compare with each other. To solve this problem MSR provides us Transformation techniques through which functional requirements of the components will be transformed to remove redundant information and to classify the components on the basis of the type of the components. MSR not only saves time by providing all the components in a single repository, but also fully involve the Customer during the Components selection and provide an accurate result based on Customer's choice.

Chapter 7

Future Work

7. Limitations & Future Work

MSR approach has some limitations like how to bring up to date information to the Repository. Different components information will be stored in the MSR repository, but when the Owner Organization will update their component, then how will they communicate with the repository so that to place the updated information. A lot of manual work is required to Extract and Load information about the thousands of reusable component.

In future we are trying to make this process automatic and to make the MSR process so intelligent that will require a minimum manual work and will automatically communicate with the Owner Organization and load information to the single MSR repository.

8. References

[1] A. Mili , R. Mili and R. Mittermeir , "Storing and Retrieving Software Components: A Refinement Based System", IEEE Trans. Software Eng., vol. 23, no. 7, pp. 445–460, July 1997

[2] Bernd Fischer, "Specification Based Browsing Of Software Component Libraries", Proceedings of Technical University of Braunschweig, Germany, 1999

[3] Chapter 1. Introduction to IEEE Std. 1517—Software Reuse Processes, 25-06-2008

[4] Edward A. Boyno , "Extraction, Transformation, and Loading in a Data Warehouse Course" Proceeding ISECON, 2003

[5] Gerald C. Gannod_, Yonghao Chen, and Betty H. C. Cheng , "An Automated Approach for Supporting Software Reuse via Reverse Engineering", 13th IEEE International Conference on Automated Software Engineering (ASE'98), 1998

[6] Haining Yao , Letha Etzkorn, "Towards A Semantic-based Approach for Software Reusable Component Classification and Retrieval", ACM Southeast Regional Conference, 2004

[7] Hyon J. Lee, "Software Reusability", 1998

[8] Jeffrey S. Poulin, Keith .J. Werkman, "Melding Structured Abstracts and the World Wide Web: for Retrieval of Reusable Components", Symposium on Software Reusability, 1995

[9] Jo Woodison , Mandarin Consulting, "Managing Software Reuse with Perforce", 2003

[10] Maxym Sjachyn, Ljerka Beus-Dukic, "Semantic Component Selection – SemaCS", 2006

[11] Roger S. Pressman , "Software Engineering", Fifth Edition

[12] Stephen White, Michael Edwards, "DomainEngineering: The challenge, Status, and Trends", 1996

[13] Thomas Jell, "Component Based Software Engineering", CUC 96

[14] Vijayan Sugumaran, Mohan Tanniru and Veda C.Storey, "Identifying software components from process requirements using domain model and object libraries", 1999

[15] Vijayan Sugumaran, Veda C. Storey, "A Semantic-Based Approach to Component Retrieval", 2003

[16] Yonghao Chen, Betty H. C. Cheng, "Formalizing and Automating Component Reuse", 1997

[17] http://www.wright.edu/~luo.3/ncrpdf/MiniHowTo MultiLoad.pdf, 23-10-2007

[18] http://www.teradataforum.com/teradata_pdf/b035-2410- 062a.pdf, 23-10-2007

[19] http://en.wikipedia.org/wiki/Extract,_transform,_load, 21-04-2008

[20] http://www.sas.com/offices/europe/czech/technologies/enterprise_intelligence_platform /Metagroup_ETL_market.pdf, 25-04-2008

[21] http://www.1keydata.com/datawarehousing/tooletl.html, 25-04-2008

[22] http://en.wikipedia.org/wiki/FastLoad, 05-06-2008

[23] http://en.wikipedia.org/wiki/MultiLoad, 05-06-2008

[24] http://en.wikipedia.org/wiki/FastExport, 05-06-2008

[25] http://www.neodynamic.com/Products/BCWinC/BarcodeWinControl.aspx?tabid=23&prodid=3&gclid=CMjU_qGUj5QCFSEbagodsVIEfw, 25-06-2008

[26] http://www.barcodelib.com/java_barcode/main.html?gclid=CMKNoYeUj5QCFR4vagodGhsMeQ, 25-06-2008

[27] http://www.devdirect.com/all/asposebarcodeforjava_PROD_00017222.aspx, 25-06-2006