International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-2 Issue-3, February 2013

Generating More Reusable Components while Development: A Technique

Navneet Kaur, Ashima Singh

Abstract*: The Component Based Software Development (CBSD) is increasingly being adopted for software development. This approach uses reusable components as building blocks for constructing software systems. The main advantages of CBSD are reduced development time, cost and efforts along with many others. The advantages are mainly provided by the reuse of already built-in software components. But there are many factors that affect the reusability of a component across many applications. Some factors can be resolved during the development of the component to make component more reusable. In this paper, some factors that affect the component reusability have been discussed with the techniques to resolve those factors. Thus a technique has been proposed for Software Component Development Organizations (SCDO) to be used while developing the components in order to generate more reusable component.

***Index Terms*: CBSD, Component, Reusability, SDCO.**

I. INTRODUCTION

The component based software development is increasingly being adopted for software development. In this approach reusable components are used as building blocks for constructing software and thus provide fast-paced delivery of software systems. This approach also provides the improvement of software quality and productivity. The main advantages of CBSD are reduced development time, cost and efforts with many others. These advantages are mainly provided by the reuse of already developed software components.

Now a day black box components are mainly developed for reuse. But due to the black-box nature of components where the source codes of these components are not available, it is difficult to measure the reusability. This may result in low component reusability.

There are many other factors that affect the component reusability. But some factors can be resolved during the development of component in order to make it more reusable.

In this paper a case study for Student Fee Submission Management System has been discussed to illustrate the some factors affecting the component reusability and solution to resolve them. The black box components have been considered in the case study. By considering these factors and using their solution during the component development, the software component development organization can generate more reusable components.

Manuscript published on 28 February 2013.
***Correspondence Author(s)**
Navneet Kaur, Computer Science & Engineering Department, Thapar University, Patiala, India.
Ashima Singh, Computer Science & Engineering Department, Thapar University, Patiala, India.

II. REUSABILITY

The term Reusability may be defined as the ability of a component that allows it to be used repeatedly. We can reduce the implementation time by using the reusable components. In the case of component based development we can build applications from existing components by assembling and replacing interoperable parts. Thus a single component can be reused in many applications that will help in providing faster development of applications with reduced cost and high quality.

There are mainly two types of reuse of a component : White Box and Black Box reuse. In the case of white box reuse the component consumer usually have the access to the source code that can be modified to include the new demands of the application. Thus the component consumer can modify the component to fit in the target system. So this type of reuse results in maximizing the reuse opportunities. But this approach has one limitation also. Because for code modification there is a need of high level of familiarity with the implementation details. On the other hand in the case of black-box reuse, the component is used as it is and in most cases the source code is not provided with the component. Thus component consumer does not have any access to the source code. In this category component must be flexible enough for better reusability. So with the black box component reuse it is difficult to determine the reusability of the component. The component consumer can rely only on the specifications. In this paper the technique for generating more reusable components has been discussed for black box components. This technique can be used by a software component development organization while generating the black box component in order to increase their reusability.

III. THE FACTORS AFFECTING COMPONENT REUSABILITY

This section describes some factors that affect the reusability of a black box component and techniques to resolve these factors.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

A. *Parameter Incompatibility*

When a component is integrated with another components then exchange of data may occur between them. But in some cases there may be some problems in the exchange of data between components. Because in the case of component based software development the components may be from different vendors and in most of the cases the source code is not provided with the Black Box components. So it may be difficult to predict the functionality of Black Box components. It may not be possible or very difficult to modify the black box component (if customization interfaces are provided). This results in reducing the component reusability.

For example when return value of one component's function is passed to the another component's function as an argument to perform its function but their data types are different then there will be parameter incompatibility problem . In this case an error may arise that may affect the other component's functionality that interact with the affected component. This may also results in wrong output or the system may hang.

This problem can be reduced by using the independent component or the component having low coupling with another components.

B. *Less Ease of Modification and Replacement*

Sometimes a need may arise for replacing a component with another component or making some modifications in the existing component, which belongs to software system, for maintenance purposes. But this may cause some problems .This may arise the requirement of modification in the components that are interacting with the modified or replaced component. Thus the component that is less replaceable and modifiable is less reusable. So this may decrease component reusability . This problem can be solved by using independent component or by using component that has less coupling with the another components.

C. *Interface Complexity*

Controlling and minimizing software complexity is one of the most important objective of each software development paradigm because it affects all other software quality attributes like reusability, testability, maintainability etc. Interface complexity is an important factor to be considered while developing a component to make it more reusable.

The interface complexity can be defined by considering its interactions (interfaces) with other components. So the component should have low no. of incoming and outgoing interactions with another components i.e the component should have low coupling with another components. Reduction in interface complexity helps in minimizing both integration and maintenance efforts. Thus the components which have less complex interfaces can be easily integrated with the other components to provide the required functionality and are more reusable . By generating the independent components or by generating the components that have less number incoming and outgoing interactions , interface complexity can be reduced. This will help in increasing the reusability of the component.

D. *Test Cases*

It is very difficult for the component consumer to trust the functionality of a black box component because in most of the cases the source code is not provided. In that case it may be

difficult for the component consumer to understand the functionality and generate the appropriate test cases to test the functionality of a black box component. Thus this will affect the reusability of the component .So the component developer should generate the appropriate test cases and provide it to the component consumer with the testing reports . This will help the component consumer in testing the component functionality. Thus it will improve the component reusability. By generating the independent component or the component with low coupling the effort, cost and time for generating the appropriate test cases can be reduced. Because the developer will have to consider the low number of combination of inputs and their expected results. Which will result in generating the simple and less number of test cases. Thus it will be convenient for the developer as well as for the component consumer.

E. *Documentation*

The documentation plays an important role in increasing component reusability. Because without it, users can not be trained and they virtually cannot use the software. The incomplete documentation reduces component reusability. Because in case of black box component , where source code is not provided by the component developer with component ,it is very difficult to trust the functionality of the component. Then Component consumer have to rely only on the specifications. So the appropriate documentation should be provided with component which must include all the functions performed by the component, their formal parameters, return types, component's dependency on the another components or software elements etc .The documentation should be simple and understandable so that the component consumer can easily understand the use of component and the way to integrate it with another components. We can also provide simple and understandable documentation by generating the independent component or the component with low coupling .Because the developer will not have to discuss many dependencies of the component on another component, software element or any other factors. That will help in reducing the complexity of the documentation and it will be easy for the component consumer to understand the component's use and its functionality . Thus it will help in increasing component reusability.

IV. SFSM SYSTEM: A CASE STUDY

In this paper, a case study of SFSM (Student Fee Submission Management) system has been considered to illustrate the impact of some factors on the reusability of a component and the techniques to resolve those factors. This system stores the student fee submission details in different files for different departments. In this system first of all the user have to log in by providing the correct password .

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Then a list of different departments will be displayed from which the user can select the department for which the user wants to record the student fee submission details in a file for that department or check the fee submission status for that department. Whole system can be represented in the form of components. Like SFSM system has been divided in the form of components as shown in fig.1 . In this case study the components in C++ language has been considered .

Login : This component is responsible for authenticating the user while the user provide the password to log in the system.

Dept_list: This component represents the list of different departments for which user want to keep the record of student fee submission details.

CSE_DEPT : This component is responsible for keeping the record of student fee submission details for CSE department. .

Similarly the components ECE_DEPT and MEC_DEPT are responsible for keeping the record of student fee submission details for ECE and MEC departments respectively.

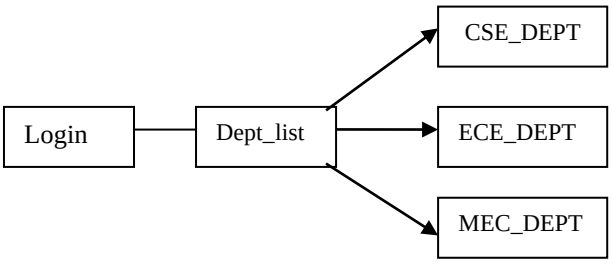

Fig.1 Representation of SFSM system in form of components

Description of the working of different components of SFSM system

Login Component

This component receives the password from the user and compare it with the password stored in a file. If match is found then it return 1 otherwise 0.

The interface of Login component has been shown below:

```
class Login
   { public:
     virtual int check_password()=0;
    };
```

The implementation for the interface of Login component has been described below:

```
class pswrd : public Login
 {   public:
   int check_password()
    {
        ifstream f("password.txt");
        char b[5];
        char ch;
        char c[5]
        int i=0;
        while(!f.eof())
       { f.get(ch);
         c[i]=ch;
                i++;
    }
   cout<<"\nEnter Password";
   cin>>b;
   int count=0;
   for(i=0;i<=4;i++)
   {
      if(c[i]!=b[i])
      count=count+1;
    }
     if(count==0)
    {  cout<<"Correct Password";
       return 1;
     }
     else
      {  cout<<"Try Again";
        return 0;
       }
  }
};
```

Dept_list Component

If the password is correct then the component named Dept_list is enabled. This component display the list of various departments for which the user want to keep the record of student fee submission details .The user can select any department for which he/she want to record the student fee submission details or check student fee submission status.

The interface of the Dept_list component has been shown below :

```
class Dept_list
 {   public:
      virtual int dept_select()=0;
  };
```

The implementation detail for Dept_list component has been shown below:

```
class choice: public Dept_list
     {   public:
         int dept_select()
        {  int b;
          clrscr();
          cout<<"\nEnter 1 For CSE Student Fee-
              Submission Regarding Operations";
          cout<<"\nEnter 2 For ECE Student Fee-
                Submission Regarding Operations";
          cout<<"\nEnter 3 For MEC Student Fee-
              Submission Regarding Operations";
          cin>>b;
          if(b==1||b==2||b==3)
             return b;
          else
             { cout<<"Enter valid option";
               return 0;
              }
           clrscr();
 }
};
```


Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

This component is connected with three components as shown in the Fig.1. It passes the return value to the connected components CSE_DEPT, ECE_DEPT and MEC_DEPT to conform the selected department. .

CSE_DEPT Component

This component performs two functions , one for recording the student fee submission details in a file named "cse_fee.txt" and second for checking the student fee submission status for CSE department.

The interface of CSE_DEPT component has been shown below:

```
class CSE_DEPT
   {  public:
      virtual void submit_csefee(char)=0;
      virtual void check_feestatus(char)=0;
    };
```

The implementation for CSE_DEPT component has been described below:

```
class fee : public CSE_DEPT
   {   public:
       void submit_csefee (char d)
        {   clrscr();
            if d=='c';
            {
              // Fee amount is 50000;
              // If submitted fee amount=50000 ,write 'Fee
                  submitted' in "cse_fee.text" file.
             // If submitted fee amount<50000, write
               'Amount Due' with amount in "cse_fee.txt"
               file.
            // If submitted fee amount>50000, write
               'Balance Amount' with amount of balance
               in "cse_fee.txt"file.
            }
         }

       void check_feestatus(char d)
        {  clrscr();
           if d=='c';
           {
               // Display the student fee submission status
                by reading file "cse_fee.txt";
            }

         }
     };
```

It has been assumed that ECE_DEPT and MEC_DEPT have similar kind of interface description and implementation.

V. POINTS TO DISCUSS

1) In the case of Login component , it has been designed in such a way that it acts like an independent component. Because it does not pass any information to the another component named Dept_list . The value returned by this component is used only to write a glue code to connect this component with the Dept_list component. Thus if we replace this component or update this component there is no need to modify the another components. We will have to change glue code only. Thus this component can be reused across another applications also (which has been implemented using components written in c++) easily by writing appropriate glue code.
2) When Dept_list component executes it displays a list of departments. When the department is selected from the list of departments to perform student fee submission management regarding operations then 1,2,3 integers are return for CSE,ECE and MEC departments respectively. When this component is integrated with CSE_DEPT,ECE_DEPT,MEC_DEPT this may cause compatibility problems. Because, the value return by this component is passed to the CSE_DEPT, ECE_DEPT,MEC_DEPT components to conform the selected department , then there will be parameter incompatibility problem. Because this component's function return integer value for selected department but the CSE_DEPT component use char value 'C' to conform the selected department. Similarly ECE_DEPT and MEC_DEPT components use 'E' and 'M' char values to conform the selected department.

   This problem may be solved by writing appropriate glue code for connecting the components. In which the value return by one component will be converted in required form before passing as an argument to another component's function. But this may require the detailed description of the working of connected components which may not be available. This may seem easy here but in the case of components with complex interfaces this may require lot of effort and cost. Thus this factor can reduce the reusability of the component.

   This problem can be solved by generating the independent components or by generating the components that will have the less coupling with another components . In that case either the component will perform its all operations by itself or it will have minimum incoming and outgoing interactions with the another components. So it will reduce the effort, cost and time to integrate the components.
3) As it has been discussed in section III, less ease of modification and replacement of component in the system also reduce the reusability because it may result in increasing the effort and cost for replacing or modifying a component. Suppose in the above case if the Dept_list component functionality has been increased to pass the department number and department name to conform the selected department. But the CSE_DEPT,ECE_DEPT and MEC_DEPT components can receive only department name. Then it will affect the functionality of these components that are connected with component Dept_list . In this case there will be need to make modifications in the CSE_DEPT,ECE_DEPT and MEC_DEPT components which may not be possible or very difficult in case of black box components. Thus this factor will affect the reusability of the components. Because it may be necessary to replace or modify a component for maintenance purpose.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

This problem can also be solved by generating the independent components or the component having low coupling with another component. Because in this case if the component is replaced or modified , it will not affect the functionality of another components that are connected to it or it will affect as less as possible. So the component can be replaced or modified with low cost and effort. Thus this will help in increasing the component reusability.

For example As shown in Fig. 1, the Dept_list component has 3 outgoing interactions with CSE_DEPT,ECE_DEPT and MEC_DEPT components because it passes the return value to these components to conform the selected department .

But the CSE_DEPT component can be generated as following to make it independent from other components .This new form of CSE_DEPT component has been given the name CSE_DEPT1.

```
class CSE_DEPT1
 {  public:
     virtual void submit_csefee()=0;
     virtual void check_feestatus()=0;
  };
```

The component CSE_DEPT1 is independent from Dept_list component. Similarly we can generate ECE_DEPT1 and MEC_DEPT1 components . Now no return value will be passed from the Dept_list component to CSE_DEPT1, ECE_DEPT1 and MEC_DEPT1 components. The returned value will be used in glue code to connect the CSE_DEPT1,ECE_DEPT1 and MEC_DEPT1 components to Dept_list component and to conform the selected department.

Thus by generating the CSE_DEPT1 component as an independent component it can be easily modified or replaced with another component without affecting any another components like Dept_list . Now the component Dept_list has no outgoing interaction with another components, so if the component Dept_list is replaced or modified it will not affect the functionality of another components . There will be need to modify the glue code only Thus this will help in increasing the reusability of the component.

4) The interface complexity also affects the component reusability. The interface complexity can be defined by considering its interactions (interfaces) with other components. So the interface should specify the less number of services with less number of incoming and outgoing interactions. If a component has many incoming and outgoing interaction then it will depend upon another components to provide its functionality. So it may reduce the ease of replacement and modification of a component. Thus it affects the component reusability. The development of independent component or the component that has less coupling with another components helps in reducing number of incoming and outgoing interaction. Thus interface complexity will be reduced . It will also help the component consumer to easily understand the services and use of services of the component. Thus this will help in increasing the component reusability.

5) As discussed in section III ,Test Cases also affect the reusability of the black box components .So the component developer should provide the appropriate test cases suite with the component to the component consumer to test the component functionality. But while generating a component that is assumed to be highly coupled with the another components , developer may have to assume so many combinations of the inputs and expected outputs. So it may be difficult for the developer to generate the appropriate test cases for component to test it's functionality.

But by generating the components that are independent from one another or are less likely to be coupled with the another components , this problem can be solved. Because in this case the component developer will have to consider the less number of combinations of inputs and expected outputs. So developer may have to generate the less number of and simple test cases . This will help in reducing the effort, time and cost for generating the appropriate test cases . It will be useful for the component consumer for testing the component functionality by using less number of and simple test cases. Thus it will help in increasing the reusability.

For example, to test the functionality of CSE_DEPT component, which is coupled with Dept_list component, the following Table I has been generated. Table I represent the various combinations of inputs and expected outputs to test the functionality of CSE_DEPT component. Thus in case of CSE_DEPT component 10 different test cases will have to be considered to test its functionality .

But if we generate the component that are independent or have less incoming and outgoing interactions we may have to generate lesser number of and simple test cases .For example in the case of CSE_DEPT1 component ,which does not accept any argument, only 5 different combinations of inputs and expected outputs have been considered to test its functionality as shown below in the form of Table II. that are less than in the cases of Table I as shown above.

As discussed above, a component's coupling with another components affects the reusability of that component and some of the various issues that decrease the reusability of the component can be resolved by reducing the coupling. Thus during the development by generating independent component or by generating the component in such a way that it will have less coupling with another components like by reducing the number of incoming inputs and outputs, the reusability of a component can be enhanced.

Table I

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Different combinations of inputs and expected outputs for test cases for CSE_DEPT Component

Seq No.	Function Name	Input	Expected Output
1	Submit_csefee(Char d)	‘ C’	Display the page for recoding fee submission details for CSE department in “cse_fee.txt” file.
2	Submit_csefee(Char d)	Any other character	Does not respond
3	Submit_csefee(Char d)	Any variable of other data type	Does not respond
4	Submit_csefee(Char d)	Submitted_Fee_Amount=50000	Write ‘**Fee Submitted**’ in file
5	Submit_csefee(Char d)	Submitted_Fee _amount<50000	Write ‘**Amount Due**’ with amount in file.
6	Submit_csefee(Char d)	Submitted_Fee _amount>50000	Write ‘ **Balance Amount**’ with balance amount in file .
7	Submit_csefee(Char d)	Any variable of other type for Submitted_Fee _Amount	Print message “enter correct information”.
8	Check _feestatus(char d)	‘ C’	Fee submission status can be viewed.
9	Check _feestatus(char d)	Any other character	Does not respond
10	Check _feestatus(char d)	Any variable of other data type	Does not respond
11	Check _feestatus(char d)		Stored information is represented in required for m.

Table II

Different combinations of inputs and expected outputs for test cases for CSE_DEPT1 Component

Seq No.	Function Name	Input	Expected Output
1	submit _csefee ( )	Submitted_Fee_ amount=50000	Write ‘**Fee Submitted’** in file.
2	submit _csefee ( )	Submitted_Fee_ amount<50000	Write ‘**Amount Due**’ with deu amount in file.
3	submit _csefee ( )	Submitted_ Fee_amount>50000	Write ‘**Balance Amount**’ with balance amount in file.
4	submit _csefee ( )	Any variable of other type	Does not respond
5	Check _feestatus( )		Stored information is represented in required form.

VI. PROPOSED APPROACH

From the above discussion a new approach has been proposed for Software Component Development Organizations to generate more reusable components. Thus by considering the following points during the component development an SDCO can generate more reusable components:

- Developing independent component or reducing component coupling with another components, by reducing number of incoming and outgoing interactions, as much as possible by using some techniques as shown in case study.
- Generation of appropriate and simple test cases to provide with the component to the component consumer to test the component functionality.
- Generate appropriate and understandable documentation about component functionality and its various other features to provide it with the component to component consumer .

VII. CONCLUSION

Although the CBSD is increasing being adopted for software development, because of its advantages like reduction in development effort, time and cost , increase in quality with many others. But there are many factors that affect the reusability of the component across many applications.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

The coupling of the component with another components also affect the component reusability. As discussed in this paper some factors that decrease the reusability of the component can be resolved by reducing the coupling during the development of component . Thus as discussed in the case study, the software component development organization should generate the independent component or the component with low coupling to improve the reusability of the component. Appropriate test case suite and documentation should also be provided with component in order to increase component reusability.

REFERENCES

1. W. B. Frakes and K. C. Kang, "Software reuse research: status and future," IEEE Transactions on Software Engineering, vol. 31, pp. 529-536, 2005.
2. M. Morisio , "Success and Failure Factors in Software Reuse," IEEE Transactions on Software Engineering, vol. 28, pp. 340-357, 2002.
3. B. Weide , "Reusable software components," Advances in computers, vol. 33, pp. 1-65, 1991.
4. Gui Gui and Paul. D. Scott , "Measuring Software Component Reusability by Coupling and Cohesion Metrics", JOURNAL OF COMPUTERS, VOL. 4, NO. 9, SEPTEMBER 2009.
5. Nasib S. Gill , "Reusability Issues in Component-Based Development", Department of Computer Science & Applications,M.D. University, Rohtak , Haryana.
6. SONG Cui-Ye, DU Cheng-Lie , "Formal Interface-Component Based SoftwareAnalysis and Design", School of Computer Science and Technology, Northwestern Polytechnical University,China,2010.
7. Usha Kumari and Shuchita Upadhyaya, "An Interface Complexity Measure for Component-based Software Systems", International Journal of Computer Applications , Volume 36– No.1, December 2011.
8. Reghu Anguswamy , "A Study of Factors Affecting the Design and Use of Reusable Component", Software Reuse Lab, Virginia Tech.,USA .
9. Gianluigi Caldiera and Victor R.Basili University of Maryland, " Identifying and qualifying Reusable Software Components", University of Maryland, February 1991.

AUTHOR PROFILE

Navneet Kaur received her B.Tech Degree In Computer Science & Engineering from University College Of Engineering, Punjabi University, Patiala, India (2011). Currently she is pursuing her degree for Master of Engineering (ME) In Computer Science & Engineering from Thapar University ,Patiala, India. Her research interests include, Software Components, Software Reuse, Software Architecture.
. (Email – navirathour27@gmail.com)

Ashima is Assistant Professor in Computer Science and Engineering Department, Thapar University, Patiala, India; and pursuing Ph.D. in Computer Science from Faculty of Engineering, UCOE, Punjabi University, Patiala, India. She holds a Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering from GZSCET, Bathinda, India (2001). She obtained her Master of Technology degree in Computer Science from Department of Computer Science and Engineering, Punjabi University, Patiala India (2005). Her research interests include Software Process Reengineering, Software Engineering, Agile Software Development, Software Quality Improvement in Small Scale Enterprises, Software Reuse, Software Process Customization and Automation, and Software Process Metrics.
(Email- ashima@thapar.edu)

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)