Integrate the Components with the Help of Glue Code Using .Net And Java Platform

ManbirKaur[1], Er.Prabhdeep Singh[2]

[#]*Department of Computer Science & Engineering,Global Institute of Engg.& Tech Amritsar*

Abstract-Modern software systems become more and more large-scale, complex and uneasily controlled, resulting in high development cost, low productivity, unmanageable software quality and high risk to move to new technology . Consequently, there is a growing demand of searching for a new, efficient, and cost-effective software development paradigm. One of the most promising solutions today is the component-based software development approach. This approach is based on the idea that software systems can be developed by selecting appropriate off-the-shelf components and then assembling them with well-defined software architecture.

Keywords- Component, Component-based software Development, Component Integration, Glue code

1. Introduction

Software development approach is very different from the traditional approach in which software systems can only be implemented from scratch. Commercial off-the-shelf (COTS) components can be developed by different developers using different languages and different platforms. This can be shown in Figure 1.1, where COTS components can be checked out from a component repository, and assembled into a target software system. Component-based software development (CBSD)can significantly reduce development cost and time-to-market, and improve maintainability, reliability and overall quality of software systems. This approach has raised a tremendous amount of interests both in the research community and in the software industry. The life cycle and software engineering model of CBSD is much different from that of the traditional ones. This is what the Component-Based Software Engineering (CBSE) is focused.

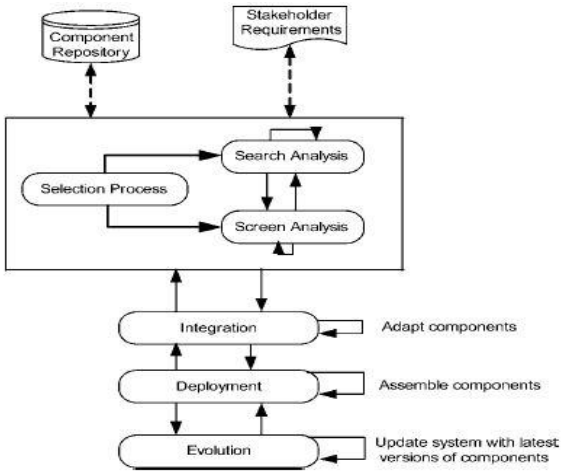

Fig1. Overview of component-based software development (CBD) process

2. RELATED WORK

Component Based Software Engineering is used to develop high quality and reliable products in less time domain and low cost. Component Selection is one of the big challenges in CBSE and different authors have presented different techniques but lacking in desired performance, accuracy and user friendliness.

The success of CBSD depends upon the ability to identify and select suitable components (Maiden and Ncube, 1998), (Alves and Finkelstein, 2002, 2003). Different methodologies like off-the-shelf-option method (OSTO) (Kontio*et al.*, 1995), COTS-based integration system development model (Tran *et al.*, 1997) based on hierarchical evaluation of component are used for component identification and selection process. Component development methodology, COMO (Lee *et al.*, (1999)] which extends unified modelling language (UML) and Rational's Unified Process, is based on clustering technique for identifying components. Jain (Jain *et al.*, 2001) proposed an approach which assists in identifying and selecting components from an object model representing a business domain by using a clustering algorithm based on a set of predefined rule and heuristics.

ISSN: 2278 – 1323

Lee (Lee *et al.*, 2001) used component identification algorithm with the focus on high cohesion and low coupling values in the process of component selection.The knowledge-based approaches include COTS-aware requirement engineering (CARE) (Chung and Cooper, 2002), which focuses on keeping the requirements flexible as they have to be constrained by the capabilities of available components and COTS usage risk evaluation (CURE) (Carney *et al.*, 2003), which is a tool that predicts the areas where the use of COTS products will have the great impact on the program. Lee and Shirani (Lee and Shirani, 2004) proposed a component-based methodology for requirement analysis and high-level design for web applications. It uses web pages as the fundamental building block of web applications and introduces pages and component classifications, and a set of tags to implement link semantics.

Wrappers (Dietrich *et al.*, 2006) were introduced to adapt components. The wrappers are automatically generated as enterprise java bean components and as proxy objects. These proxy objects intercept method calls and provide functionality required by the overall component-based system. The concepts of aspect-oriented software development have also been incorporated into CBD integration process.

The emergence of CBSE has introduced various component models such as Component Object Model (COM) from Microsoft (Rogerson, 1997), Enterprise Java Beans (EJB) from Sun Microsystems [JAVA], Common Object Request Broker Architecture (CORBA) from OMG [COR] and .NET from Microsoft [NET]. All these models define standard forms and interfaces for component deployment. Moreover, these models allow components to be freely exchanged between and across application domains and development contexts. The adoption of these component technologies has rapidly changed the world of software development.

3. Component Integration Approach

Component-Based Software Engineering (CBSE) is concerned with composing, selecting and designing components. As the popularity of this approach and hence number of commercially available software components grows, selecting a set of components to satisfy a set of requirements while minimizing cost is becoming more difficult.

i. Adaptation

Adaptation has always been an important challenge for software engineering. Systems have to be continuously revised to address new functional or non functional requirements, changing environment. The need for adaptation may appear at any time in the software lifecycle: development, deployment, supervision and maintenance (evaluative, corrective).[4]

ii. Testing and validation

Validating adapted components is a major task for the CBD process. Software component testing and validation techniques focus on the expected behaviour of the component to ensure that the exhibited behaviour is correct. The internal structure of the components is usually unknown, the most appropriate technique for component testing and validation is black box testing. The black box testing is for customized parts to uncover the functional and behaviour errors of the new and altered parts in components based on given specification and its strategies include test case generation by equivalence classes, error guessing and random tests. These techniques rely on some notion of the input space and expected functionality of the component being tested.[3]

iii. Inspections

Systematic software testing requires a large number of tests to be developed, executed and examined and this means that testing is both time-consuming and expensive. Inspections have proved to be an inexpensive and effective technique to identify defects for two reasons. Onereason is that many defects can be found during one inspection session while a test normallyonly identifies one or a few defect at the time. The other reason is that inspections reuse domainand implementation knowledge among the reviewers in a more efficient way than tests. Inspections can be done on different levels of formality and can either be done manually or by means of some analysis tools.[3]

4. Experimental Work

A. Introduction

Glue code is the code used to provide the functionality to integrate different components. It deals with control flow, Component Bridge and exception handling.
Glue code can be used to transfer information between computer languages; it is not required to do so. Generally, it allows one piece of code to call functions in the other, or allows small data values to be passed between code blocks. Generated glue code, particularly when it connects distinct computer languages, often contains code pieces specific for each connected code module.

ISSN: 2278 – 1323

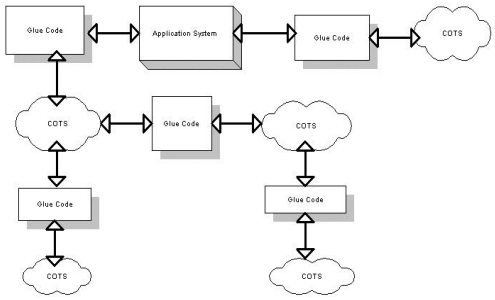

Fig2. Overview of Glue code

B. Actual System

In purposed system we will create web service that act as a Glue code between website filecloud.io and programming languages like java, PHP and .Net. In this system user of the various programming language can access the website filecloud.io through this web service.

In the existing system (filecloud.io website) support API in PHP language only. If the user of java, .Net and cold fusion want to direct access this website it not work and create error. So to solve this crisis situation we will implement web service that work as a glue code b/w website and various languages. It shall provide interface b/w website and various languages.

This web service makes the API language independent. This work creates a frame work through which we can access the website by various languages. It increases flexibility of the website. The definition of Glue code will be improved to include the word web service which it currently does not.

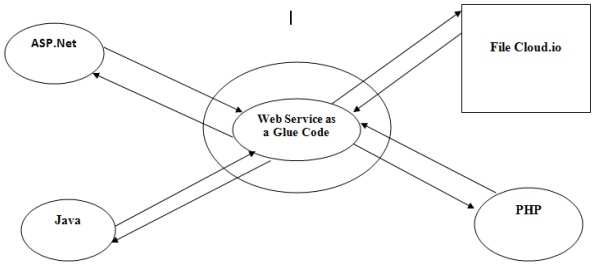

Fig3. Actual System

C. Glue Code using Dot Net

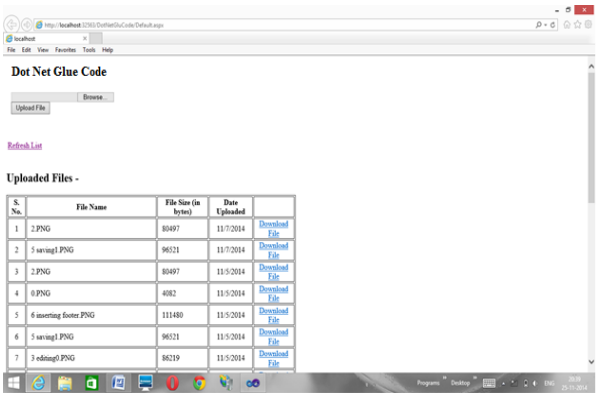

Fig4 This table shows how to upload the file.

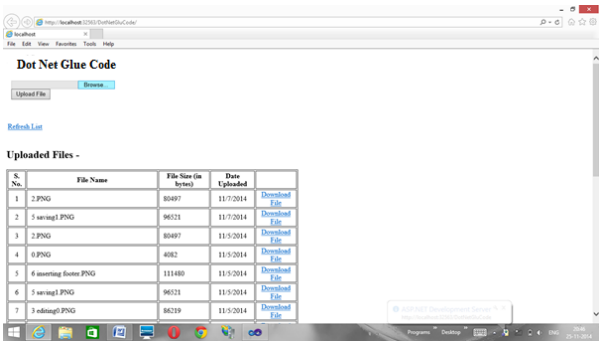

Fig5 This table shows the how to browse the file.

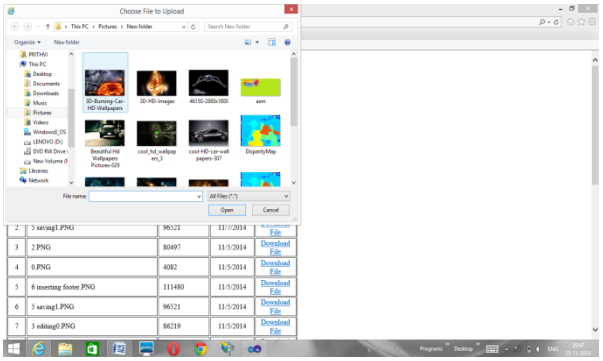

Fig6 Here, we choose any file.

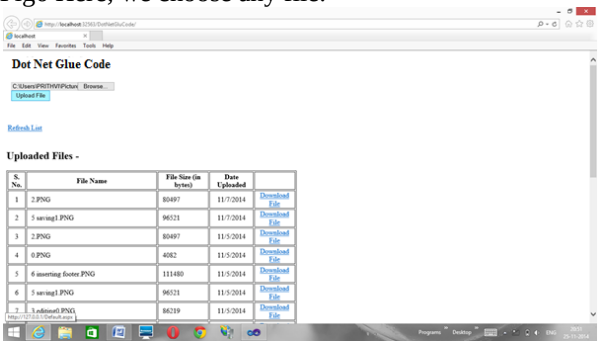

Fig7 Then upload that file.

ISSN: 2278 – 1323

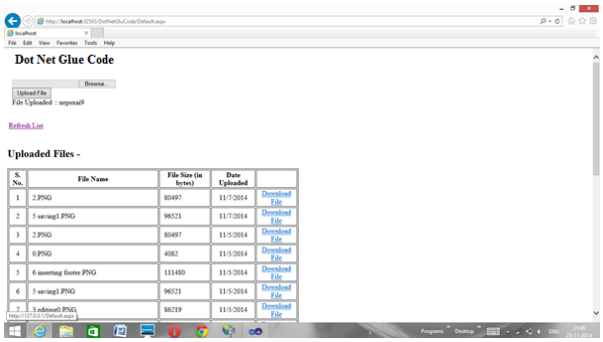

Fig8 file uploaded.

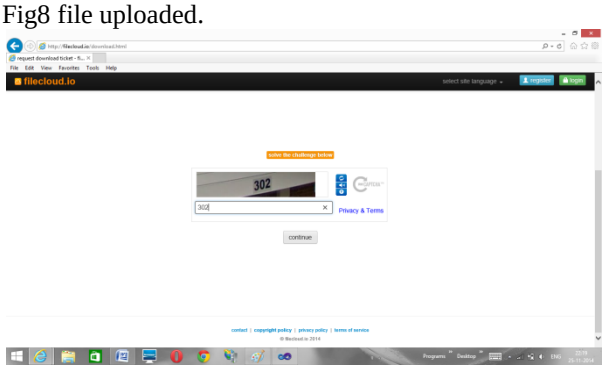

Fig9If we want to download that file, then dialog box will appear and here we enter the above shown characters.

D. Glue code using Java

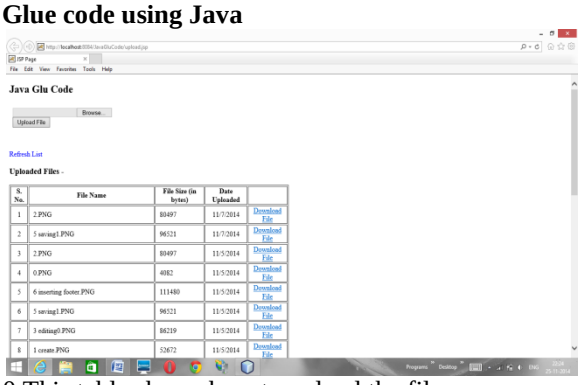

Fig10 This table shows how to upload the file.

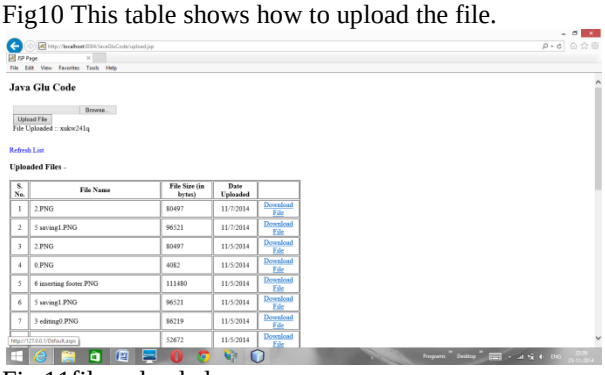

Fig 11file uploaded

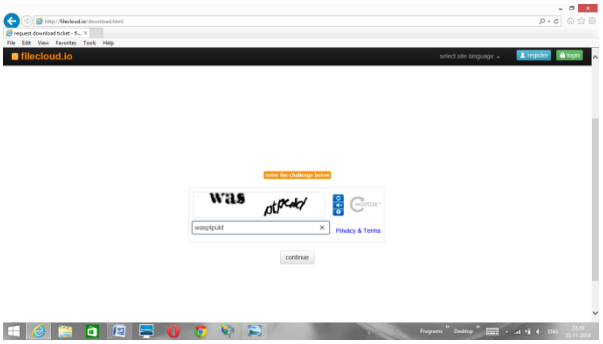

Fig11If we want to download that file, then dialog box will appear and here we enter the above shown characters.

5. Conclusion

CBSD approach is based on the idea to develop software systems by selecting appropriate commercial off-the-shelf components and then to assemble them with a well-defined software architecture. COTS-based software development mainly involves COTS identification, COTS selection and COTS integration activities. COTS component selection is regarded as the most crucial phase. Component Integration gradually puts the pieces together – COTS, glueware and traditionally developed software – to assemble the final application.

In our research work we use glue code technique to integrate various components. In future work we can integrate the various components by using various levels of adaptation like Architecture level and component level adaptation.

REFERENCES

[1] G.Pour, "Software Component Technologies: JavaBeans and ActiveX", Proceedings of Technology of Object-Oriented Languages and systems, 1999, pp. 398-402.

[2] Szyperski, C., Gruntz, D. and Murer, S. (2008), *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley Professional, Boston, First Edition 1997. ISBN 0-201-17888-5.

[3] Mahmood, S., Lai, R. and Kim, Y.S. (2007), "Survey of Component-based Software Development", IET Software, Volume 1, No. 2, pp. 57-66.

[4] Alves, C. and Finkelstein, A. (2002), "Challenges in COTS decision-making: a goal-driven requirements engineering perspective", In *Proceedings of the 14th international Conference on Software Engineering and Knowledge Engineering* (Ischia, Italy, July 15 - 19, 2002). SEKE '02, vol. 27. ACM, New York, NY, pp 789-794.

[5] Maiden, N.A., and Ncube, C. (1998), "Acquiring COTS software selection requirements", IEEE Software, 15, (2), pp. 46–56.

[6] Kontio, J., Chen, S.-F., Limperos, K., Tesoriero, R., Caldiera, G., and Deutsch, M. (1995), "A COTS selection method and experience of its use", *Proc. 20th Annual Software Engineering Workshop*, Greenbelt, Maryland, 1995.

ISSN: 2278 – 1323

[7] Tran, V., Liu, D.-B., and Hummel, B. (1997), "Component based systems development: challenges and lessons learned", *Proc. Eighth IEEE Int. Workshop Software Technol. Eng. Practice*, pp. 452–462
[8] Lee, S.D., Yang, Y.J., Cho, F.S., Kim, S.D., and Rhew, S.Y. (1999), "COMO: a UML-based component development methodology", In *Proc. Sixth Asia Pacific Software Eng. Conf.*, pp. 54–61.
[9] Lee, J.K., Jung, S.J., Kim, S.D., Jang, W.H., and Ham, D.H. (2001), "Component identification method with coupling and cohesion", In *Proc.Eighth Asia-Pacific Software Eng. Conf.*, pp. 79–86.
[10] Jain, H., Chalimeda, N., Ivaturi, N., and Reddy, B. (2001), "Business component identification – a formal approach", *Proc. Fifth IEEE Int. Enterprise Distributed Object Computing Conf.*, EDOC '01, pp. 183–187.
[11] Chung, L. and Cooper, K. (2002), "Knowledge based COTS aware requirements engineering approach", *Proc. 14th Int. Conf. Software Eng. Knowledge Eng., (ACM Press)*, pp. 175–182.
[12] Carney, D.J., Morris, E.J. and Place, P.R.H. (2003), "Identifying commercial off-the-shelf (COTS) product risks: the COTS usage risk evaluation", Carnegie Mellon Software Engineering Institute (SEI), CMU/ SEI-2003-TR-023, Sept. 2003.
[13] Lee, S.C., and Shirani, A.I. (2004), "A component based methodology for web application development", Journal of System Software, 71, pp. 177 - 187.
[14] Dietrich, S.W., Patil, R., Sundermier, A. and Urban, S.D. (2006), "Component adaptation for event-based application integration using active rules", J. Syst. Softw., 79, (12), pp. 1725–1734.
[15] Rogerson, D. (1997), *Inside COM*, Microsoft Press, ISBN 1-57231-349-8.
[16] [COR] OMG, CORBA, http://www.omg.org/corba
[17] [JAVA] Sun Microsystems, "JavaBeans 1.01 Specification", http://java.sun.com/beans
[18] [NET] Microsoft .NET Framework (2009), http://en.wikipedia.org/wiki/NET_Framework, preview release 19-10-2009.
[19] Zhuge, H.: 'A Problem oriented and rule based component repository', J. Syst. Softw., 2000, pp. 201–208
[20] Pressman, R.S. (2000), *Software Engineering-A Practitioner's Approach*, McGraw-Hill International Ltd., New York.
[21] Cechich, A., Piattini, M. and Vallecillo, A. (2003), "Assessing component based systems, Component based software quality", LNCS 2693, pp. 1–20.
[22] Boehm, B., Abts, C., "COTS Integration: Plug and Pray", *IEEE Computer*, 32(1), Jan. 1999, pp. 135-138.

ISSN: 2278 – 1323