

Case Study of Glue Code using .Net Platform

Manbir Kaur¹;Gurpreet kaur²

Department of Computer Science & Engineering, Global Institute of Engg.& Tech¹

Amritsar

manbirkhurana90@gmail.com¹;gurpreet.8891@gmail.com²

Abstract-Component based software engineering is achieving more popularity in today's software development community due to the increasing complexity of software systems, increasing costs of maintenance, and decreasing costs of underlying hardware. This causes software community to follow Brook's famous "buy versus build" colloquy. Business organizations therefore prefer to build their systems from previously built components which means they should concentrate more on selecting and composing components than manually adopt software systems. As the popularity of such approaches grows, commercial software vendors tend to develop more commercial software components and connectors.

Keywords-COTS, GlueCode, Web Service, ComponentIntegration,File cloud

1. Introduction

Component based software engineering is achieving more popularity in today's software development community due to the increasing complexity of software systems, increasing costs of maintenance, and decreasing costs of underlying hardware. CBD not only requires focus on system specification and development, but also requires additional consideration for overall system context, individual components properties and component acquisition and integration process.

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.

For example, to connect C++ with Java, the generated code may include both a C++ file and a Java file.

Glue code for COTS components is the new code needed to get a COTS product integrated into a

larger system. It is usually clearly defined as connected to the COTS component itself, acting more as a "bridge" between the COTS component and the system into which it is being integrated [18].

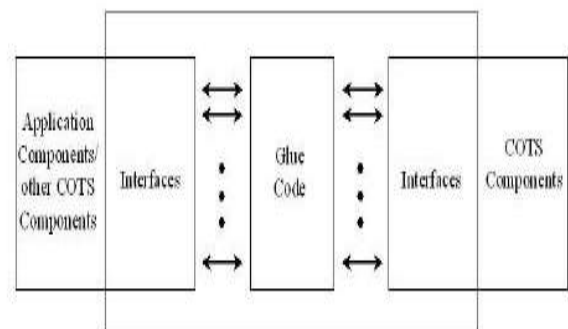


Fig1 Glue Code Configuration

2. Related Work

CBD not only requires focus on system specification and development, but also requires additional consideration for overall system context, individual components properties and component acquisition and integration process.(Fox et al, 1997) in their paper "A Software Development Process for COTS-based Information System Infrastructure" has Proposed IIDA (The Infrastructure Incremental Development Approach).(George and Helgo, 2000) in the paper "An

experiment in Component Adaptation” has compared various approaches to adapting software components. They believe this area of research needs much investigation since current state-of-the-practice of component-based software engineering is unable to achieve its promised goals.(Wilhelm Hasselbring,2001) in the paper “Component-Based Software Engineering” has discussed that with component-based software engineering, software systems can be created and maintained at lower costs and with increased stability through reuse of approved components in flexible software architectures.(Wilhelm Hasselbring, 2003) in the paper “Managerial Goals for Component-Based Software Engineering” has discussed various factors like Cost Reduction, Ease of Assembly, Reusability, Customization, Flexibility and Maintainability.(Xia et al, 2007) in the paper "CBSE: Technology, Development Frameworks and QA schemes ", has discussed current component-based software technologies, describe their advantages and disadvantages, and discuss the features they inherit. In the paper they also address QA issues for component-based software.(S. Mahmood and Y.S. Kim 2007) in the paper "Survey Of Component-Based Software Development" has discussed various techniques for component identification and selection, integration, deployment and evolution.(Amandeep and Shivani, 2011) in the paper,"COTS Components usage Risks in Component Based Software Development" has discussed Risk identification approach for component-based development, a number of risks in various component-based development stages and these risks arise due to the widespread belief that it is a low risk development strategy.(Sajjad Mahmood and Mohammed A, 2012) in the paper “Towards a Glue-Code Specification Framework for Component-Based Systems” they present an initial glue-code framework to specify the glue-code required in integrating potentially mismatched components and the missing functionalities required to meet the requirements of a CBS. They define glue-code specification framework consists of two phases, namely, use case conceptual mapping and component based sequence mapping.(Jongmoon et al, 2001) in paper “Empirical Software Simulation for COTS Glue Code Development and Integration” they define how the glue code development process and the COTS component integration process affect each other and how they affect development effort and schedule throughout the development life cycle, based on given

parameters via software simulation that provides a method for checking the understanding of the real world process and thus help people produce better results in the future.(Jennifer et al,2002) in the paper "An Approach for Understanding and Testing Third Party Software Components" has discussed an approach to mitigating software risk by understanding and testing third party, or commercial off-the-shelf (COTS), software components. They use the approach, based on the notion of software wrapping, gives system integrators an improved understanding of how a COTS component behaves within a particular system.

3. Methodology

i. Component Based Software development (CBSD)

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. This new 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.

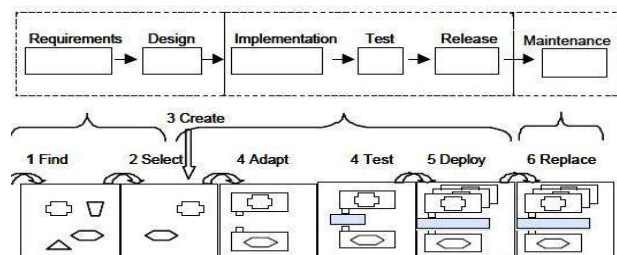


Fig.1 Development cycle compare with the Waterfall Model

ii. Component Selection

Component-Based Software Engineering (CBSE)[22] 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. In the selection process, we should assign each component a set of requirements it satisfies. Each component is assigned a cost which is the overall cost of acquisition and adaptation of that component. Many organizations have supported their reuse with component-based technologies. Many organizations are spending much time in reusable component selection since the choice of the appropriate components has a major impact on the project and resulting product.

iii. **Component Integration**

The integration process includes integration of standard infrastructure components that build a component framework and the application components. The integration of a particular component into a system is called a component deployment. In difference to the entire system integration a component deployment is a mechanism for integration of particular components it includes download and registering of the component. Components need to be integrated to constitute usable software systems. Various forms of component integration, such as integrating legacy systems and enterprise/office application integration are discussed in the following subsections.

- Integrating Legacy Systems into Component-Based Systems
- Enterprise Application Integration
- Office Application Integration

iv. **Development**

The selected components are integrated through well defined infrastructure and this infrastructure provides the binding that forms a system from the disparate components. The operation of infrastructure consists of three main levels, namely highest abstract level that defines how the different components will interact to carry out the required functionality, lower level that will be used by components to interact and carryout common task; and software component level that will implement the necessary coordination services.

4. **Experimental Work**

A. **Introduction**

Glue is the code used to provide the functionality to integrate different components. It deals with control flow, Component Bridge and exception handling. Glue code is computer code that unites programs or software components that would not be compatible otherwise. This code usually does not serve a purpose such as computation or calculation, but serves exclusively as a proxy between two incompatible pieces of software. Object-oriented programming languages can be connected to scripting languages, two object-oriented languages can be connected together, or large pieces in the same language can be united by this code.[20] Although 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. Some glue code generators allow user-specified data structures to be passed between code modules, but not all of them do. Moving large pieces of data through the connecting code may not always be reliable.

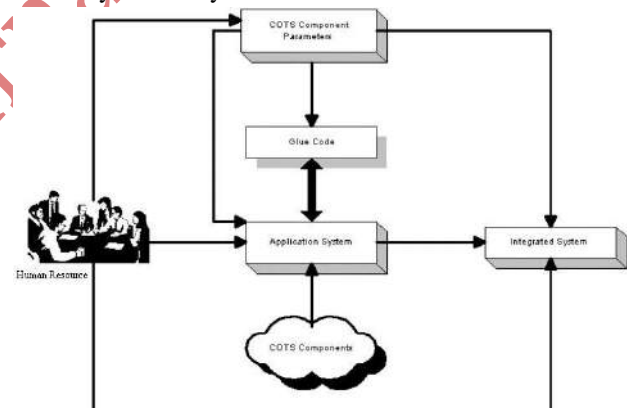


Fig2. System Configuration of COTS glue code development and integration[20]

B. **Use Glue Code in .NET**

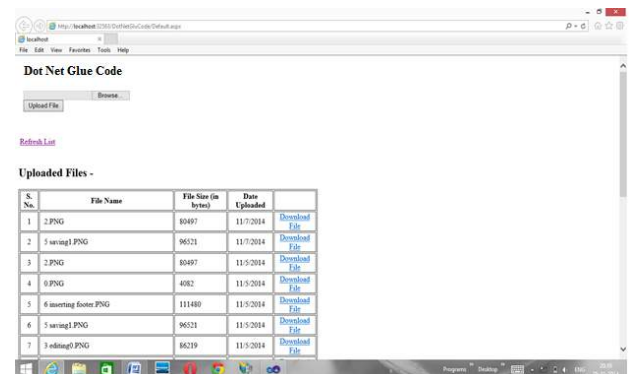


Fig3. This table shows how to upload the file.

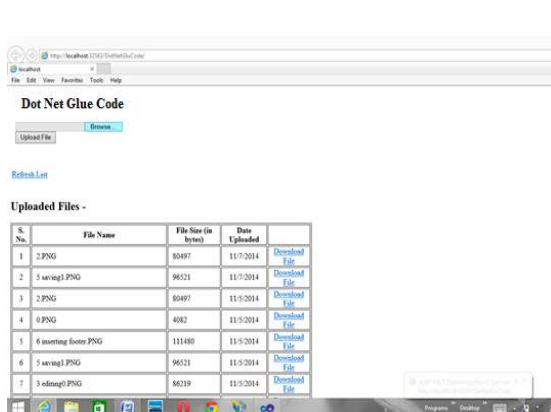


Fig4. This table shows the how to browse the file.

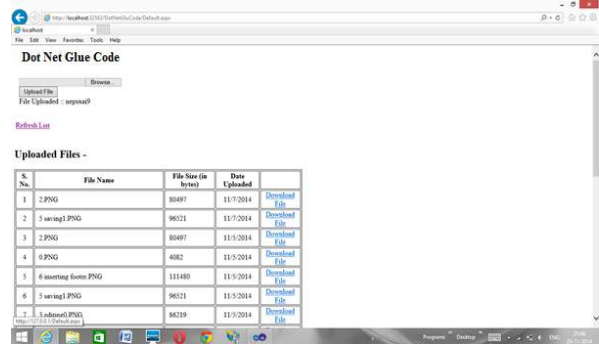


Fig8. File uploaded.

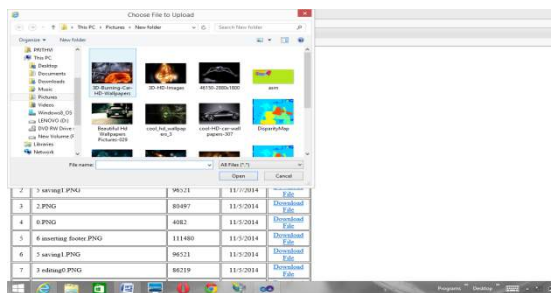


Fig5. Here,we choose any file.



Fig9. The above highlighted row shows file is uploaded.

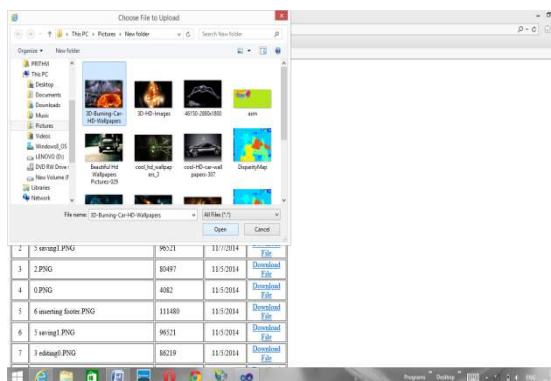


Fig6. Suppose we choose any of the above file.

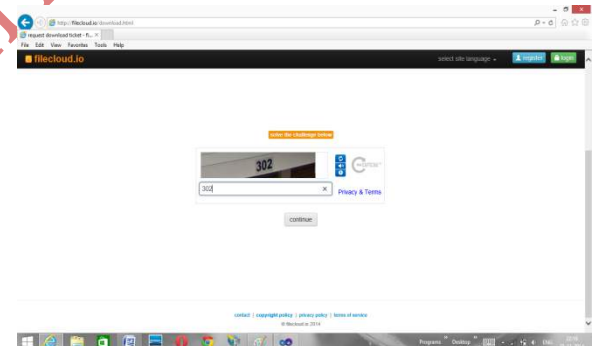


Fig10.If we want to download that file, then dialog box will appear and here we enter the above shown characters.

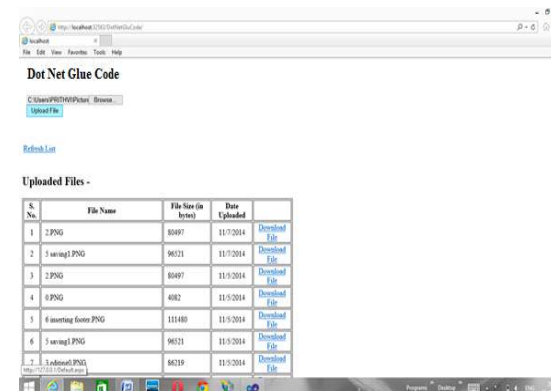


Fig7. Then upload that file.

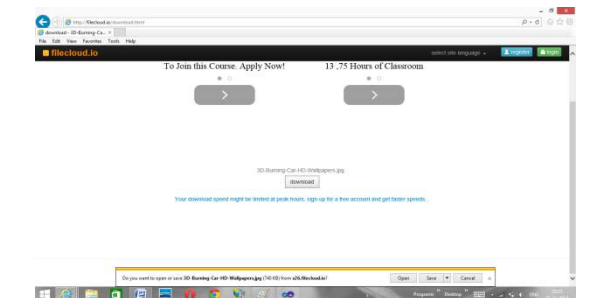


Fig11. This shows that whether we want that file should be open, save and cancel.

5. Conclusion

Commercial off-the-shelf (COTS) components can be developed by different developers using different languages and different platforms. 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.

Our research work is successfully implemented without any error. We implement web service that work as a glue code between .Net platform and filecloud.io website. We take the example of this work that explains in above snap shots.

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.
- [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.