Towards a Glue-Code Specification Framework for Component-Based Systems

Sajjad Mahmood* and Mohammed A. Al-Qadhi†
Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
Email: smahmood@kfupm.edu.sa*, moh6666_qadi@hotmail.com†

***Abstract*—Component integration is a key process activity during the development of Component-Based Software (CBS). CBS integrators use software components developed by either in-house or purchased off-the-shelf to develop an application. CBS integration process often requires adaption of selected components to meet the CBS-to-be requirements as individual components are usually developed for general purposes. Latest industrial studies for CBS development have indicated that system integrators reply on individual experiences to write the glue-code of a CBS. Hence, there is a need for an integration framework to handle the potential mismatches between individual components and identify the functionalities not provided by the available components for a CBS. In this paper, we 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.**

***Index Terms*—Component Based Systems; Component Integration; Glue-Code Specification; UML; Use Case Model.**

I. INTRODUCTION

Software component is a fundamental building block for an application. CBS development focuses on integrating pre-existing software components to build a software application [1], [2], [3], [4]. A system integrator puts together software components developed by different vendors who are usually unaware of each other [5]. Hence, the integration phase of a CBS development life cycle is a challenging activity as individual components are usually designed for general purposes and they might not completely satisfy requirements of a CBS-to-be. Furthermore, detailed documentation is rarely available for the majority of components [5], [6] and system integrators have to rely on component interface documentation during the integration phase of a CBS.

The glue-code written during the integration phase plays an important role in the overall success of a CBS. The glue-code provides a platform to integrate potentially mismatching components and implement the missing functionalities required to meet requirements of a CBS. CBS research [2], [6] has shown that system integrators rely on their experience to write the glue-code and there is a lack of glue-code development framework to support the important integration phase of the CBS development life cycle.

In this paper, we present an initial glue-code specification framework for writing the glue-code of a CBS. The glue-code specification framework will help system integrators in early identification of potential mismatches between component interfaces and missing functionalities required to satisfy stakeholder requirements of a CBS. We introduce the notation of use case conceptual mapping and component-based sequence mapping to specify interactions between components of a CBS. We also use a hotel reservation system [7] as a running example to explain the glue-code specification framework.

The rest of this paper is organized as follows: Section II reviews the related literature. In Section III, we present the glue-code specification framework. We conclude the paper and discuss future work in Section IV.

II. RELATED WORK

Vigder and Dean [8] presented the concept of wrappers to glue software components by considering the elements of architecture during the integration process. Rine et al. [9] used adapters to integrate components. Each component has an associated adapter and components request services from each other through their associated adapters.

Dietrich et al. [10] used active rules to design wrappers to adapt components. The wrappers are automatically generated as components and they act as proxy objects. These proxy objects intercept method calls and provide

 ISBN: 978-1-61208-230-1

the functionality required by the overall component-based system. Similarly, Canal et al. [11] presented a model based approach for component adaptation using synchronous vectors based notation to automatically generates adapter protocols during development of a CBS.

Kim et al. [12] proposed a process for CBS development where component integration occurs at the release phase. Zitouni et al [13] presented a contract-based approach to analyze and model the properties of components and their composition in order to detect and correct composition errors. The approach allows to characterize the structural, interface and behavioral aspects of the components. Chi [14] defined the signature view and behavior view of software components and used pi calculus expressions to model behavior of components.

The current CBS development approaches lack a systematic process to bridge the gap between requirements analysis and integration specification of a CBS. This results in an integration phase that heavily relies on system integrator's skills which increase the challenges associated with the glue-code specification of a CBS.

III. GLUE-CODE SPECIFICATION FRAMEWORK

The glue-code specification framework presents a systematic structure to identify the required interfaces and specify interactions between components of a CBS. The glue-code specification framework also helps reduce CBS development risks by identifying the interface mismatches between components and the missing functionalities required to implement a CBS.

The glue-code specification framework consists of two phases, namely, use case conceptual mapping and component based sequence mapping. The first phase - use case conceptual mapping - starts with a process of modeling a use case as a required interface and subsequently specifies all the required and provided interfaces for a CBS. The second phase - component based sequence diagram - uses the extended UML sequence diagram to model different scenarios associated with a use case to identify mismatches between required and provided interfaces of a CBS. Figure 1 shows the glue-code specification framework.

A. Use Case Conceptual Mapping

The UCCM phase takes the Unified Modeling Language (UML) use case diagram [15] and component interface documentation as an input to develop a realization mapping between software components and requirements of a CBS. First, we adopt UML component specification technique [7] to model each use case of a CBS as a 'conceptual interface' which consists of a set of operations corresponding to individual interactions of the use case. Figure 2 shows 'IMakeReservation' system interface for the hotel reservation system.

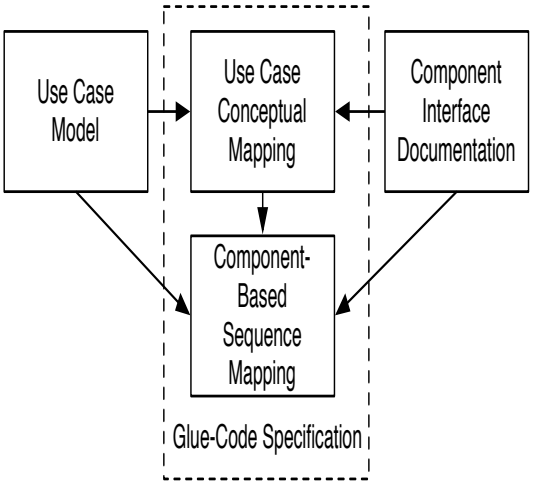

Fig. 1. Glue-Code Specification Framework

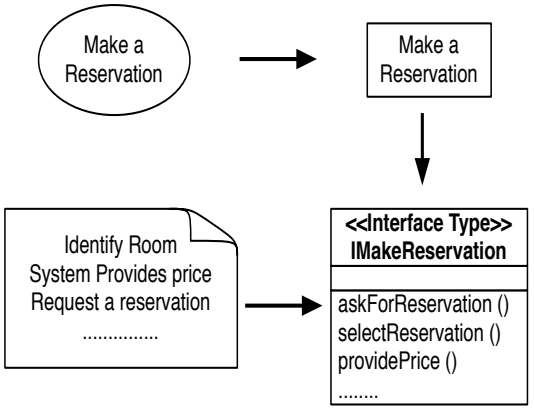

Fig. 2. 'IMakeReservation' System Interface

Second, each interface of a component is represented as a 'concrete interface' which consists of a number of concrete operations which are uniquely identified by a 'concrete operation code'. A 'concrete operation code' consists of components' name, interface number and corresponding operation number. For example, the concrete operation code (B-I2-O2) will represent the second operation of the second interface of component B.

Finally, a UCCM realization table is generated for each 'conceptual interface'. A UCCM realization table shows all conceptual and concrete interfaces that help realize a

 ISBN: 978-1-61208-230-1

TABLE I
'IMAKERESERVATION' REALIZATION TABLE

Conceptual Operations	Realization
$askForReservation()$	$Missing$
$selectReservation()$	$Missing$
$providePrice()$	$Partially(D-I2-O6)$
$getCustomerInfo()$	$Missing$
$reservation()$	$(A-I1-O1)AND$
	$(D-I2-O8)AND$
	$(B-I2-O5)OR(E-I2-O4)$
$refineReserveDetails()$	$Missing$
$provideAlternative()$	$refineReservDetails()AND$
	$(D-I2-O6)OR(E-I5-O3)$
$acceptOrReject()$	$Missing$
$failure()$	$Missing$
$notifyBillingSys()$	$Missing$

'conceptual interface'. Table I shows UCCM realization table for the 'IMakeReservation' interface.

B. Component Based Sequence Mapping

The CompBSM phase uses UML sequence diagram to present scenarios associated with each 'conceptual interface' of a CBS. The sequence diagram developed during the CompBSM phase has a glue-code component that helps a system integrator in identifying the execution precedence, missing functionalities and potential mismatches between a set of participating interfaces. In this paper, we extend UML by introducing new stereotypes to help system integrators in specifying the glue-code required to integrate candidate components. The message interaction stereotypes are as follows:

1) <<Initiation>>: To specify first/initial messages.
2) <<Missing>>: To specify missing functionalities.
3) <<LibraryImporting>>: To specify importing header files and built-in libraries from components.
4) <<Adapter>>: To specify messages that involve mismatched data types involved in an interaction.
5) <<TemporaryStorage>>: To specify interactions that provides temporary variables to store values to be used through a scenario.

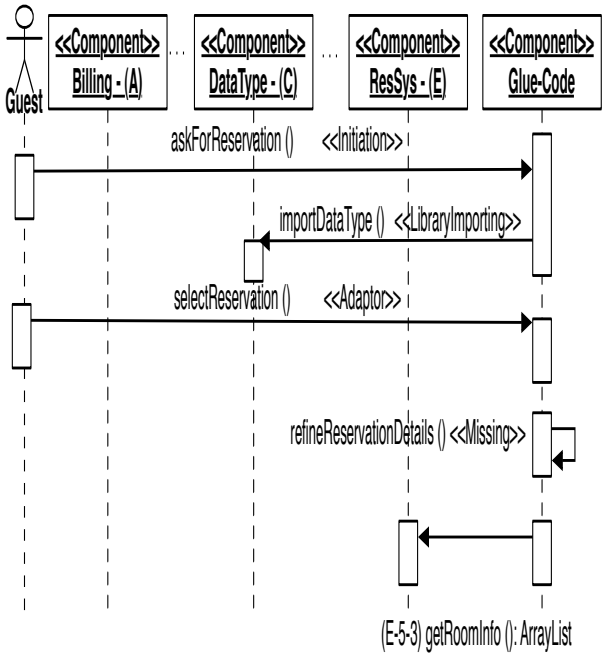

Fig. 3. 'IMakeReservation' Sequence Diagram

6) <<ExceptionHandling>>: To specify interactions that provides exception handling functionality for the system.

Figure 3 shows a partial sequence diagram for the IMakeReservation use case. A guest actor starts the scenario by invoking method 'askForReservation ( )' from the glue-code component of a CBS. The interaction is labeled with <<Initiation>> stereotype to indicate the first interaction in the scenario. Next, glue-code component makes a message call to 'importDataTypes ()' method of the 'datatype - (C)' component. The interaction is labeled with <<LibraryImporting>> stereotype to specify that built-in datatype library is called form the 'datatype - (C)' component. Next, 'selectReservation ()' method is executed which is tagged with the <<Adaptor>> stereotype. This indicates that some sort of adaptation code needs to be written in the glue-code component to collect the required information from the customer.

Furthermore, 'refineReservationDetail' () method is tagged with <<Missing>> stereotype to indicate that the required functionality is not provided by any available component and it needs to be implemented in the glue-code component of a CBS. Finally, '(E-5-3) getRoomInfo ()'method is invoked that shows that 'getRoomInfo ()' method associated with 'ResSys' component is called to realize the desired functionality required for the IMak-

 ISBN: 978-1-61208-230-1

eReservation use case.

IV. Conclusion and Future Work

We have presented an initial glue-code specification framework to help system integrators in understanding potential mismatches between component interfaces and missing functionalities required to meet stakeholder requirements of a CBS. The glue-code specification framework consists of two parts; namely, use case conceptual mapping and component-based sequence mapping phases. The use case conceptual mapping is used to specify the component interfaces involved in realizing individual use cases a CBS. Furthermore, component-based sequence mapping phases uses UML sequence diagram to specify the component interface mismatches and missing functionalities required to successfully developing a CBS.

In this paper, we have presented a preliminary work on the glue-code specification framework. For future work, we plan to complete the framework by considering both integration and composition issues related to the glue-code of a CBS. There is a need to apply the framework on real world case studies to better understand its potential benefits for the system integrators of a CBS. We plan to develop an automated tool for supporting glue-code specification framework. Furthermore, there is also a need to investigate the potential benefits of the complete glue-code framework on integration testing, maintenance and evolution phases of a CBS.

Acknowledgement

The authors would like to acknowledge the research support provided at King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia.

References

[1] S. Mahmood, R. Lai, and Y. S, Kim, "Survey of component based software development," *IET Software*, vol. 1, no. 2, pp. 57–66, 2007.

[2] J. Li, R. Conradi, O. P. N. Siyngstad, C. Bunse, M. Torchiano, and M. Morisio, "Development with off-the-shelf components: 10 facts," *IEEE Software*, vol. 26, no. 2, pp. 80 – 87, 2009.

[3] M. A. Khan and S. Mahmood, "Optimal component selection for component-based systems," in *2009 International Conference on Systems, Computing Sciences and Software Engineering*, pp. 467 – 472, 2009.

[4] M. A. Khan and S. Mahmood, "A graph based requirements clustering approach for component selection," *Advances in Engineering Software*, vol. 54, pp. 1–16, 2012.

[5] A. Cechich and M. Piattini, "Early detection of cots component functional suitability," *Information and Software Technology*, vol. 49, no. 2, pp. 108 – 121, 2007.

[6] S. Mahmood and A. Khan, "An industrial study on the importance of software component documentation: A system integrators perspective," *Information Processing Letters*, vol. 12, pp. 583–590, 2011.

[7] J. Cheesman and J. Daniels, *UML Components A Simple Process for Specifying Component Based Software*. Addison-Wesley, 2001.

[8] M. R. Vigder and J. Dean, "An architectural approach to building systems from cots software components," in *Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research*, p. 22, IBM Press, 1997. Toronto, Ontario, Canada.

[9] D. Rine, N. Nada, and K. Jaber, "Using adapters to reduce interaction complexity in reusable component based software development," in *Proceedings of the 1999 symposium on Software reusability*, pp. 37–43, ACM Press, 1999. Los Angeles, California, United States.

[10] S. W. Dietrich, R. Patil, A. Sundermier, and S. D. Urban, "Component adaptation for event-based application integration using active rules," *Journal of Systems and Software*, vol. 79, no. 12, pp. 1725 – 1734, 2006.

[11] C. Canal, P. Poizat, and G. Salaun, "Model-based adaptation of behavioral mismatching components," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 546 – 563, 2008.

[12] S. Kim, S. Park, J. Yun, and L. Y, "Automated continous integration of component-based software: An industrail experience," in *Proceedings of 23rd IEEE/ACM International Conference on Automated Software Engineering*, pp. 423 – 426, 2008.

[13] A. Zitouni, L. Seinturier, and M. Boufaida, "Contract-based approach to analyse software components," in *13th IEEE International Conference on Engineering of Complex Computer Systems*, pp. 237 – 242, 2008.

[14] Z. Chi, "Software components composition compatability checking based on behaviour description," in *IEEE International Conference on Granular Computing*, pp. 757 – 760, 2009.

[15] K. Bittner and I. Spence, *Use Case Modeling*. Addison-Wesley, 2002.

 ISBN: 978-1-61208-230-1