Volume 2, Issue 4, April 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An UML Based Approach for Process Modeling Using COSEML

Ravi Uyyala
Dept. of Computer Science And Engineering,
Bm College of technology
Indore,India
Uyyala.ravi@gmail.com

Dr.Nilay Khare
Dept. of Computer Science And Engineering,
MANIT,BHOPAL
bhopal, India

Abstract: ***Component Oriented Software Engineering (COSE) seems to be the future of software engineering. Currently, COSEML (Component Oriented Software Engineering modeling Language) is the only modeling language that completely supports the COSE approach. Abstract decomposition of the system and their representing components are shown in a hierarchy diagram to support the COSE process model. In COSEML, only static modeling is supported through a single diagram. However, software is about behavior and static modeling is not sufficient to describe the system. So, collaboration diagrams are added to COSEML so that it can support dynamic modeling. In complex systems, modeling the dynamic behavior can produce too many collaboration diagrams. This may create difficulties to see the overall dynamic behavior of the system. A more abstract view is needed that shows the cooperation among the collaboration diagrams. So, the aim of this thesis is, providing an abstract view of the system by using interaction overview diagram like structures that are available in Unified Modeling Language (UML).***

Keywords: Component Oriented Software Engineering (COSE, COSEML (Component Oriented Software Engineering modeling Language), Unified Modeling Language (UML), Collaboration Overview Diagram, Runtime Collaboration Diagrams.

1. Introduction

In the current state of the software industry, there is an exponential increase on the demand for software. Scope and complexity of the software have increased dramatically. Current software industry mostly deals with huge government and military applications. Because of the high competition in the industry, such software systems should be built in less time with less cost. For these reasons, today's software systems are more likely to face the software crisis.To respond to the demand and overcome the software crisis, new approaches have been developed that benefit from component technologies. Among these approaches, Component Oriented Software Engineering (COSE)[1] proposes building software by integrating existing components. A modeling language, COSEML [2], and a graphical modeling editor COSECASE [3] were developed for COSE approach. This modeling language is based on a single hierarchy diagram.

2. Motivation for using collaboration overview diagrams in COSEML

In complex systems, modeling the dynamic behavior can produce too many collaboration diagrams. This may create difficulties to see the overall dynamic behavior of the system. A more abstract view is needed that shows the cooperation among the collaboration diagrams. Such a modeling view can be incorporated to COSEML.

2.1 Collaboration Diagrams in COSEML

In COSEML, two types of collaboration diagrams, abstract and run time collaboration diagrams have been used. COSEML shows abstract decomposition of the system and corresponding real components together on the hierarchical diagram. Emphasis on these two views is supported by defining these two collaboration diagrams.

2.2 Abstract Collaboration Diagrams.

In abstract levels, abstract collaboration diagrams are utilized for supporting the decomposition model of the COSEML. High-level requirements and the behavior of the system can be modeled using this type of collaboration diagram.

Utilizing these diagrams can help to find incompleteness and inconsistency in scenarios and requirements in the analysis phase Thus they are useful for testing the correctness of the structural decomposition.

© 2012, IJARCSSE All Rights Reserved

2.3 Run Time Collaboration Diagrams

For the physical level, run-time collaboration diagrams are utilized. These diagrams show behavioral aspects of the system on component level. Interactions among interfaces are shown using sequence of method calls and event signals in the diagram. Showing these interactions permit to model the implementation of complex operations. Run time collaboration diagrams are intended for the implementation phase and they mostly contain implementation details. These diagrams are suitable for making wiring-level decisions on the model.

2.4 Sequence Numbering

Nested numbering notion is selected for message sequencing in COSEML.

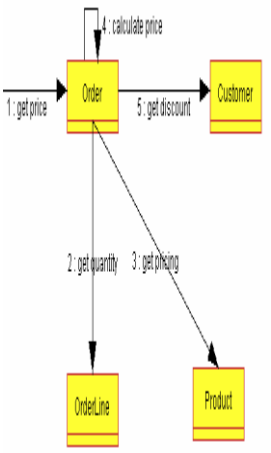


Figure.2.4.Use of flat numbering on message sequences

2.5 Conditions and Loops

Conditional flows are the essential part of dynamic modeling so they are defined in the specification. A conditional message is denoted by putting the conditional expression between square brackets as shown in figure

[condition] message

This visualization represents a conditional message, which is activated if and only if the condition between the square brackets evaluates to true. Another structure, which is also essential for modeling complex flows, is loop structure. When one or more messages are called more than once, these structures are needed.

Notation is similar to that of a conditional message; difference is the asterisk in front of the brackets as seen in figure. Condition between the square brackets is the loop condition and while it evaluates to true, that message and sub messages are iterated. In order to iterate a group of messages, setting a loop condition to the parent message is sufficient.

*[loop condition] message

3.Specification

Collaboration overview diagrams are insprired by interaction overview diagrams avilable in UML. It is a directed graph consisting of nodes and edges. All the symbols used to represent these nodes and edges are explained below. Fig.4. gives the pictorial representation of these symbols.

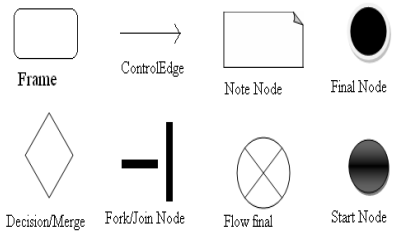


Fig3.1. Different symbols used in collaboration overview diagram

A collaboration overview diagram is a directed graph, consisting of nodes and edges

4. Experimental Study

Implementation Details

Previous version of the COSECASE contains main hierarchy diagram and collaboration diagrams. We are not changing the structure of the previous version, instead we are adding the proposed new diagram as a plugin. To browse the newly added diagram, it has been added to the Diagram Tree as a node. Diagram Tree, is a JTree structure that contains a main hierarchy diagram and zero or more collaboration diagrams. New collaboration diagrams and collaboration overview diagrams can be added; existing ones can be renamed or removed. On the other hand, main hierarchy diagram can only be renamed. It cannot be removed from the diagram. Tree model of the Diagram Tree is shown in figure. All these functionalities are accessed via pop-up menus to save space. In addition, Diagram Tree can be made invisible with a menu item defined in View Menu.

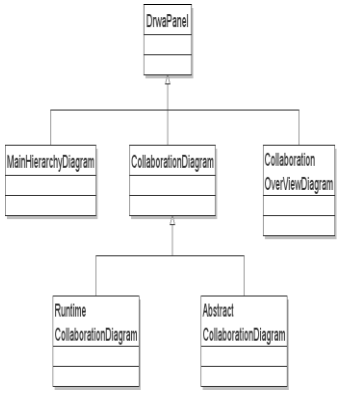


Fig.4.1.New diagram structure in COSEML

Figure shows Existing Link structure in COSECASE

© 2012, IJARCSSE All Rights Reserved

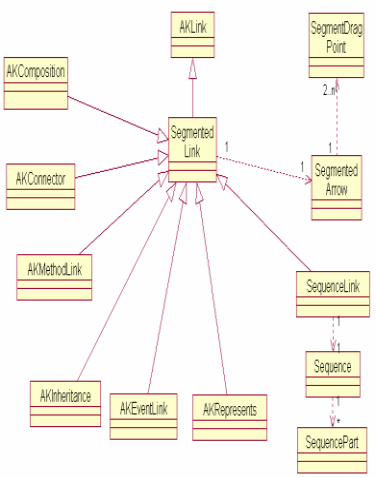
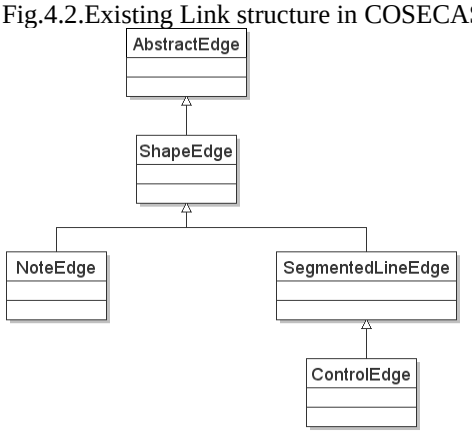


Fig.4.2.Existing Link structure in COSECASE



4.3.Newly added Link structure to COSECASE

5. Case study: E-Store Application

In this case study, an e-store application is modeled with COSECASE. Modeling power of collaboration diagrams and collaboration overview diagrams in the Component Oriented Software Modeling approach is clearly illustrated in this work.

5.1 Logical Decomposition

COSE modeling approach starts with system decomposition. Possible packages at the first level of the decomposition are listed below Account, Product Catalog, Shopping Cart, Order, Customer, Web Figure shows the first level decomposition of the e-store application on COSECASE. Creating accounts and login operations are handled in the Account package. Information about products, prices and product stocks are managed in Product Catalog package. This package also has search functionality for products, prices and stocks in the e-shop. Shopping Cart package manages a virtual shopping cart. This package handles buying decisions of the customers. Products can be added or removed until payment approval.

Another package in the e-store application is the Order package. This package manages the checkout of the items in the shopping cart, shipping of the items to the customers and payment process. Keeping the order status is also handled in this package. Customer package represents the real customers and keeps user preferences and other customer related information in the system. Finally, Web package represents the web pages on the e-store application. Customers interact with the system using the web pages provided by this package

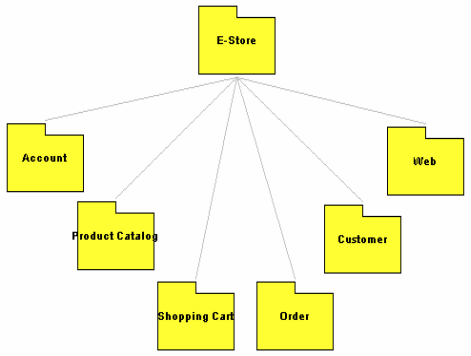


Fig 5.1. First level decomposition of the e-store application.

5.2 Use Cases Realizations with Abstract Collaboration Diagrams.

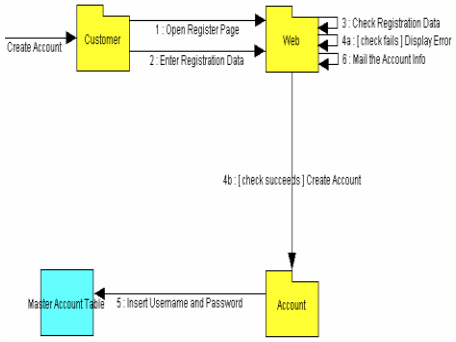
In this section, some of the most important use cases of the system are observed and their realization is modeled using abstract collaboration diagrams.

5.3 Use Case - Create Account

To use the e-store application, customers first need to create an account. A use case with the name Create Account is obvious.

5.4 Steps:

Customer opens the registration page of the e-store., Customer enters registration data (username, password, e-mail) to the registration page. Registration page checks the validity of the data. If check fails, an error message is displayed. If check succeeds, registration page creates the account. Account package inserts a new row to the Master Account Table with given username and password, Registration page mails the account details to the customer, Possible collaboration participants in this use case are Customer, Web, Account and Master, Account Table defined in the main hierarchy diagram. Figure 5.4 shows the realization of Create Account use case with the abstract collaboration diagram



5.4 .Abstract collaboration diagram of Create Accounts Use case login

© 2012, IJARCSSE All Rights Reserved

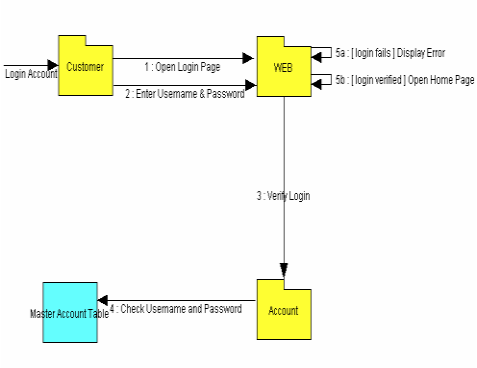


Fig.5.5. Abstract collaboration diagram of Login.

Use Case Browse Products

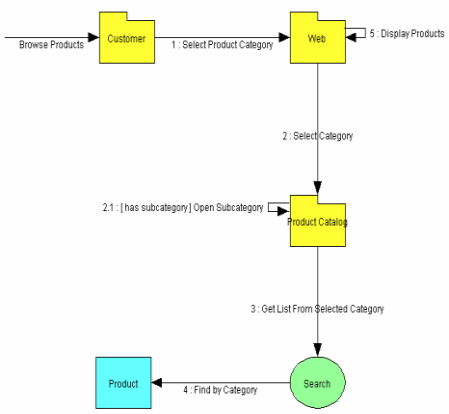


Fig.5.6.Abstract collaboration diagram of Browse Products

Use Case Search by Product Property. Customers should be able to make a search on a specific product property.

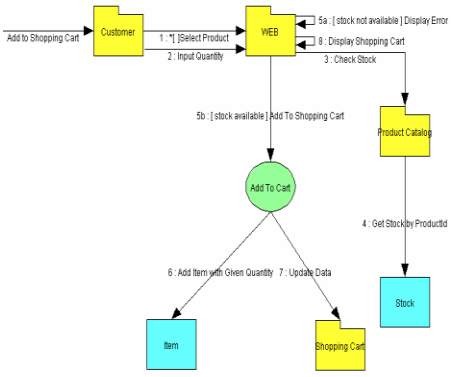


Fig.5.7. Abstract collaboration diagram Search by Product Property.

Use case Add to Shopping Cart

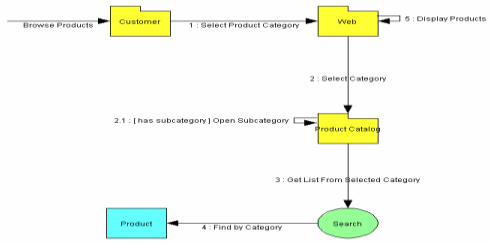


Fig5.8. Use Case Add to Shopping Cart

When customers want to buy a product, they add it to the shopping cart provided by

the e-shop and continue shopping. Adding a product to the shopping cart is given with the

following use case steps. Use Case Edit Shopping Cart

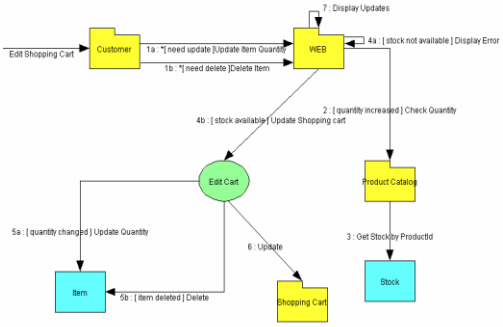


Fig.5.9 Abstract collaboration diagram of Case Edit Shopping Cart

Use Case Update shipping Preferences

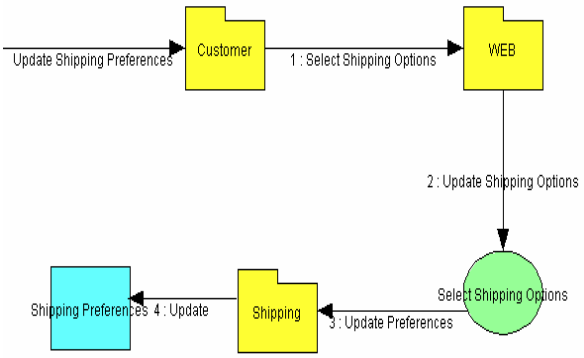
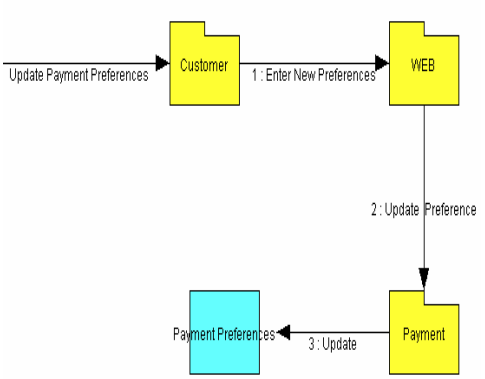


Fig.5.10. Abstract collaboration diagram of Update Shipping Preferences

Update Payment Preferences

© 2012, IJARCSSE All Rights Reserved



5.11. Abstract collaboration diagram of Update Payment Preferences

Below Figure shows all the abstract packages and their representing components that have
been defined in this section.

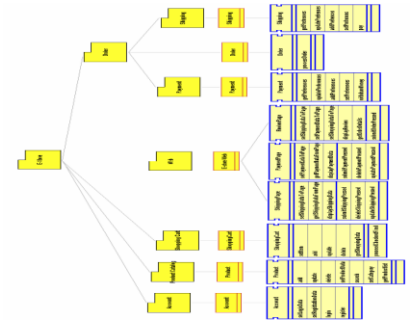


Figure:5.12: the abstract packages and their representing components diagram

6. Collaboration Overview Diagram

A customer visits the e-store and creates an account in order to benefit from shopping for the first time. Later he can use the same login information to loginto the system. Once he successfully logged into the system, he can browse the product catalog and examine the features of the products. While browsing, he makes a selection and adds them to a virtual shopping cart provided by the e-shop application. During this process, customer may edit the contents of this shopping cart. Then he clicks to proceed to checkout button to buy the items in the shopping cart. In the checkout process, he selects the shipping and payment options and then approves the order. Following that, system charges the customer and sends the products to the shipping address.

7. Runtime Collaboration Diagrams

Runtime collaboration diagrams show implementation details of a system behavior and they provide a very suitable medium for component wiring. In the previous section, components of the e-store system are found and included into the main hierarchy diagram. With run time collaboration diagrams, implementation of the use cases can be modeled by showing the interactions among the real components. In this section, the Proceed to Checkout use case, which is the most complex scenario in the e-store application, is modeled using a run time collaboration diagram. Below Figure shows the first phase of the run time collaboration

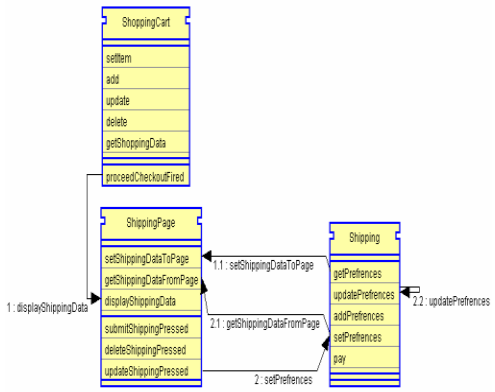


Fig 7.1. First phase of the run time collaboration

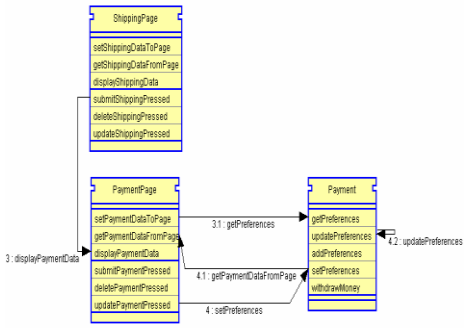


Fig.7.2.Second phase of the run time collaboration

Third phase of the run time collaboration

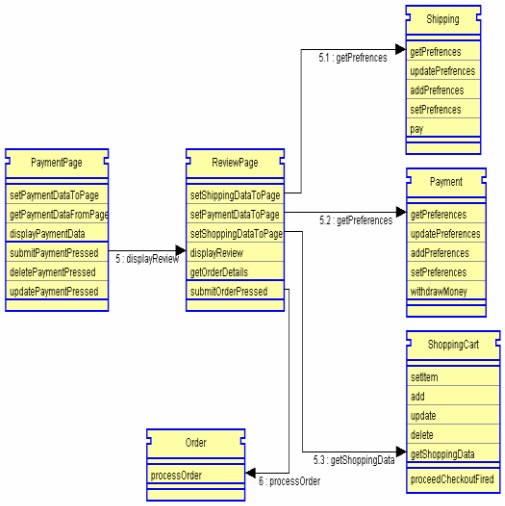


Fig.7.3. Third phase of the run time collaboration

8. Conclusion

In complex systems, modeling the dynamic behavior can produce too many collaboration diagrams. This may create difficulties to see the overall dynamic behavior of the system. A more abstract view is needed that shows the cooperation among the collaboration diagrams. Such a modeling view has been proposed for COSEML.

9. Future Work

There have been different applications of COSE based tools that offered framework kind of environments where

© 2012, IJARCSSE All Rights Reserved

components can be composed to yield executable applications. Such composition, however could not be guided effectively, by the COSEML model: the order of the messages to be fired were totally left to the designer's intuition during the composition. Future frameworks can import the presented collaboration modeling abilities for a further automated and guided composition. Then, the order of message invocations can be retrieved from the COSEML model.

10.References

[1] A.H. Dogru and M.M. Tanik, "A Process Model for Component Oriented Software Engineering", IEEE Software, vol. 20, no 2, pp. 34-41, January 2003.

[2] A.H. Dogru, "Component Oriented Software Engineering Modeling Language: COSEML", TR-99-3, Computer Engineering Department, Middle East Technical University, December 1999

[3] A. Kara, "A Graphical Editor for Component Oriented Modeling", M.S. Thesis, Middle East Technical University, December 2001.

[4] OMG, "Unified Modeling Language: Superstructure", Version 2.0, Formal/05-07-04, http://www.omg.org/cgi-bin/doc?formal/05-07-04, August 2005.

[5] OMG, "CORBA Basics ", http://www.omg.org/gettingstarted/corbafaq.htm, 2009.

[6] Microsoft, "Component Development", http://msdn.microsoft.com/enus/ library/ms834352.aspx, 2009.

[7] Sun, "Reference Documentation ", http://java.sun.com/products/ejb/index.jsp, 2009.

[8] X. Cai, M.R. Lyu, K. Wong, R. Ko, "Component-Based Software Engineering: Technologies,Development Frameworks, and Quality Assurance Schemes ", Proceedings of the Seventh Asia-Pacific Software Engineering Conference, p372, 2000.

[9] G. Booch, J. Rumbaugh, I. Jacobson, "The Unified Modeling Language User Guide", Addison Wesley, April 2000.

[10] OMG, "OMG Model Driven Architecture", http://www.omg.org/mda, 2009.

[11] M. Fowler, "UML Distilled ", 3rd Edition, Addison Wesley, 2004.

[12] A. Abdurazik and J. Offutt, "Using UML Collaboration Diagrams for Static Checking and Test Generation", Third International Conference on the Unified Modeling Language, pp. 383-395, York, UK, October 2000.

© 2012, IJARCSSE All Rights Reserved