Design of Transaction Management System in Component Based Software Model for Secured Object Transmission

Kumar Rahul[1],
Assistant Professor, NIFTEM

Rajiv Ranjan Singh[2],
Assistant Professor, SLC (E), DU

***Abstract* - This paper focused the principle of component provided methodology through transaction for different component sharing among distributed machine or software. Transaction management system functionality defines in the terminology of providing component to the other requesting software, maintaining component, and showing the status of component in either availability of component, component being used or can't access the component. Some of the component can't be access because of security issues of any node in Component Based Software Model (CBSM). Transaction management system (TMS) work successfully if algorithmic functionality of different subsystem of TMS gives numeric status of sharing the component based on time constraint.TMS responsible for abstracting as well as allowing the component for sharing among different system on Component Based Software Model (CBSM).Time constraints in this system measured for every transaction of every requesting software, that is being need of component to execute the program.**

Keywords: Component Based Software Model, Transaction Management System

1. INTRODUCTION

Software reuse, the use of existing software artifacts or knowledge to create new software, is a key method for significantly improving software quality and productivity[1]. This system is basically work on the technology of transactions, there main objective is to recover as well as provide the components among all other system on component based software model (CBSM). At distinct time interval different components are requested by different system so it is difficult for the system (a system that is responsible to provide components to systems) to maintain the components according to the time intervals. When one component is using by any system then other system can't use the components for that time interval. Different programmers create components, often working on different systems with different methodologies. Understanding and tracking different transactions on components is increasingly difficult in large and complex to maintain on component based software model (CBSM)[2]. It is hard to evaluate any systems as robust and efficient if they do not understand the availability and maintainability of system. When any components used by any system and being modified then it will have different status of components from the previous one being used. There will be a number of components exists based on different transaction performed. So for this purpose a transaction table maintained with every system that helps the system to identifying the components. Link based transaction reduces resources consumption. Link based transaction state that how one component become the input for other component in the system. Different components will have a relationship to describe how one component is related to other component. If same result found at system side by using the two's same components then it will show any available from two components can be used. Sometimes other system or components do not properly detect the exact component that is needed for implementation. This leads to a problem of system halting states there no execution takes place.

1.1 Transaction definition

A transaction is a unit of program execution that accesses and possibly updates various data items. Transaction merely shows the outcome, but it reflects the value for other period of execution. In component sharing among system, transaction management system role play as to modify the component, maintain the component, evaluate the component and provide the component to systems. Transaction will have a significant role over components characteristics. Any system requesting for components utilization for individual application, transaction of components between different time interval shows whether components can be reused or not. Any system, on CBSM system will have different application in running mode, and single component required at many of the times in all of the application or some of the application. Once transaction happens then it modifies the statement or the value within the components that reflect in other system while reusing of the components.

1.2 Proposed link based transaction activity

Some of the talks where considering resources as components, while others separated the two concepts[4].

Proposed link states that how a component is interacting with other component. Link ensures that reflection of one component to another component. When any component being used and it require to fetch data then it will get other component, and it will implement through function in system. Any component will reuse other component during execution time of the software. So for reusing the component depend on "link of component". Here, link based transaction of component shows whenever any component will be use by any system then other system will be affected when it will reuse a link based component.

Here many systems will be using to access the component based on the requirement. Transaction of component shows immediate effects on other connected component.

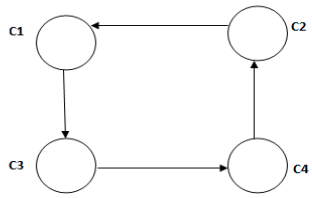

Figure 1.2 Link based component

1.3 Transaction effect on system and component in component based software model (CBSM)

The state of the art in proving correctness is such that, although it generally cannot be applied to a product as a whole (software), it can be applied to module-size pieces of software[3]. According to Vasappanavara, a programming paradigm "it is the manner in which programming elements such as functions, objects and variables are exploited to produce the desired output"[5]. Transaction effect shows how a component is changing for other software. When a component c1 is using by system1 in a particular time interval and system2 is demanding for component c1 at the same time interval then c1 status will reflect "BUSY" in transaction table for specific time interval. Component c2 is being used by s2 and requested by s3, so before delivering the component c2 to s2, it will be checked that the requested time interval by s3. The moment when s2 will release the component c2, it will reflect the current status of c2 in transaction table is free for specific time interval. So the same strategy applied for all the remaining systems as well as components. The moment when a system gets the component,

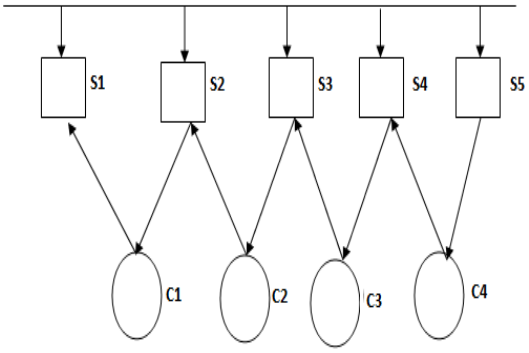

Figure 1.3 Component Based Software Model (CBSM)

there is possibility of change the internal data of component as per requirement. So before hand over the required component to desired system, there is responsibility of transaction manager to check the final update in component and status of component. Status of component means availability of component for all other system in component based software model (CBSM).

We can understand this CBSM with the help of following table where many components as well as many systems are working together to access the components.

System	Component	Time interval	Status
System1	C1	T1-T2	Busy
System2	C2	T2-T4	Busy
System2	C2	T1-T2	Free
System3	C3	T2-T4	Free

Table1.3.A Transaction table for all components under CBN Model as:

Here, time interval T1-T2 states that C1 is busy for this time interval. Similarly for time interval T2-T4, component C2 is busy means system2 is using the component C2. Again, for the same component C2, system2 is making it free before using it in time interval T2-T4. Similarly we can understand the different status of component in either same or different time interval. This table gives us knowledge that how any component can be used by any system in different time interval. When a component such as system3 is using C2 for specific time interval, and they changed the data according to the requirement. Reusing the component does not mean that (whatever component is hiring to reuse in system), it should be used completely in the software under system. Either it can use it partially of the component or can use it fully of the component. So, the final value of C2 will reflect in system3 only. After this transaction over C2, C2 will have two value, *previous* and *now*. So, after this transaction over C2, if other component such as System4 want to reuse the C2, will have two different value of C2. In below table,

system4 is using the modified component C2, and using this modified C2 100%.

System	Component	Previous	Current
System3	C2	X	Y
System4	C2	Y	Y

Table 1.3 B Transaction reflections on C2.

2. Proposed methodology for Designing Transaction Management System (TMS)

Transaction management system state that how to manage the different form of component to provide it to the system on component based software model (CBSM). While accessing the component under any system, if component data change then it will have different value of one component, so TMS decide on request generated by system that which form of component required by any system. TMS uses the table to provide the component to systems.

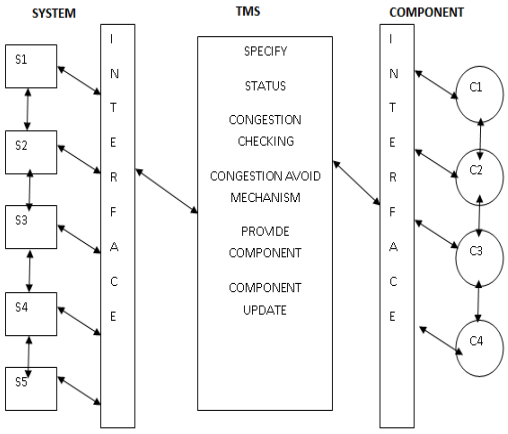

Figure 2. Transaction Management Systems (TMS)

System	Component	Previous	Current
S1	C1	X	Y
S2	C1	Y	Y
S3	C2	P	Q

Table2.A Component with different status after transaction

Component's status after transaction with accessibility value: It states that components will have different status in different time interval. It will maintain in tabular form to show for other system before accessing the component.

System	Component	Time interval	Previous	Current	Status
S1	C1	T1-T2	X	Y	Busy
S2	C2	T2-T3	P	Q	Busy
S3	C1	T3-T4	X	X	Free
S4	C2	T4-T5	Q	Q	Free

Table 2.B Component with accessibility status

2.1 **Specify**: It is the basic functionality of TMS that will recognize system and component individually. It recognizes which system in demand of which type of component? Every system is connected to each other in component based software model. In the similar fashion each component will be connected to each other. The reason is because while accessing of one component other component may require giving outcome of the software at any system.

2.2 **Status**: Status ensures that whether the requested component is free or busy. Based on the table of component with accessibility status, it would be clear that which component is using when? Status is important activities of TMS. Based on this value accessibility of component define.

2.3 **Congestion checking**: Congestion of component ensures that there is no accessibility of component at TMS. TMS will not allow any system to access the component if there is congestion of component. TMS will check components before hand over the component to other system. TMS job is not only of transaction refection but to make sure of component availability to other system also. One component will be access by only one system in finite interval of time. Even there will be priority of individual system to access or reuse the component for internal use.

2.4 **Congestion avoids mechanism**: This mechanism or algorithm apply to avoid congestion of components between system as well as component. It ensures that the required and provided components between systems will be achieved successfully.

2.5 **Provide the component**: TMS provide the component based on required number of component. If transaction changed internal statement or other information of

component then it will have two different format of component, i.e., previous and now.

2.6 **Component update**: it is the last activities of TMS that happen when the transactions plays a major role for different format of component availability for different system on component based software model (CBSM). It updates the value or internal data of component.

3 Dependency analysis between components:

It shows the relationship between components. Since components helps to coordinate between systems, so components provide different system functionalities by interacting, coordinating and cooperating. Generally a set of components depend on each other after any transaction activities occur on any component. It provides complex system functionality through accessing of different components. Whenever any system required components then there will be dependency of components exist between systems. In a simple manner dependency is a bidirectional relationship between two or more components. Dependency values reflect based on transaction of components. Dependency analysis between components implement through graph G where G = (C, E). This is a directed graph which has a finite set of vertices C (called components) and E is the set of dependencies path between components. For any system such as S1 on CNBM (component based software model) if the system uses a number of component then it can be measurable in terms of accessing or not, based on dependency value from adjacency matrix representation of different component of S1 as follows:

	C1	C2	C3	C4	C5
C1	0	1	0	0	1
C2	1	0	1	1	0
C3	0	1	0	1	0
C4	0	1	1	0	1
C5	1	0	0	1	0

Figure 3. Matrix presentations of components

Here, 0- Means there is no dependency,
1- Means there is dependency.

So, if the system S1 is using any component such as C1 and wants to use C2. There is dependency between these two components C1 and C2. Now it will check the availability of component in a particular time interval. But if there is no dependency means individual component are using separately. Transaction on any component will affect the status of other components. If C3 is using by system such as S3 and it required the component C4, so any transaction over C3 will affect on C4 because both are dependent on each other.

4. Transaction based component dependency among different system on CBSM

T1	T2	T3
S1:=READ C1; ALTER(C1); WRITE(S1);	S2:=READ C1; WRITE (S2);	S3:= READ C3; ALTER(C3) WRITE (S3);

Figure 4 Transaction based component

This transaction based component shows that T1, T2 and T3 means time interval (it can be system time). In this transaction system, different system works in different time interval and reusing different components. When system S1 using component C1 and modifying the internal data according to the requirement (in this case given in transaction based component) of system S1, so finally there will be two different value will be available to the other system for reusing, it depends on the requested system which data of available components system wants to reuse (whether previous or current data of component). So the requested system will come to know from the "table maintained with every system" as we have mentioned at *Table 2.B Component with accessibility status*. Modified data of any component will be available only to that requested system or reusing systems that are really using the component. When S2 demanding the component C1 during time interval T2 then again it will be finalize the some important things:

1. Whether system can access a particular component or not.
2. Component is busy or free.
3. Check the congestion status.
4. If congestion can be there then avoid it through congestion avoiding strategy.
5. Ensure dependency analysis between different components.
6. Provide the component.
7. Finally update the component and preserve the previous as well as current data of the component.

5. CONCLUSION

This paper deals transaction effect on component when any system uses it in a finite interval of time. TMS uses congestion control strategy also that helps in a way to resolve congestion among components. Congestion avoided based on dependency between components. It

gives secured component transmission between systems. Dependency among component shows how and when a system access component. Transaction management system having high complexity to maintain the relationship between different system as well as different component with the help of interface. In this paper, TMS uses CBSM to show the transaction effect and availability of component to perform reusability. Many system works simultaneously in same time interval but accessing or reusing the component depend on availability of component.

6. REFERENCES

1. FRAKES, W. AND FOX. C. 1996. Quality improvement using software reuse failure modes model. *IEEE Trans. Softw. Eng. 24*, 4 (April), 274–279.
2. G. Pour, "Software Component Technologies: JavaBeans and ActiveX," Proceedings of Technology of Object-Oriented Languages and systems, 1999, pp. 398 – 398.
3. Brereton, B. and Budgen, D. (2000): Component-Based Systems: A Classification of Issues. In IEEE Computer, November 2000, pp. 54-62.
4. Duclos, F., Estublier, J., and Morat, P., "Describing and using non functional aspects in component based applications," *Proceedings of the 1st international conference on Aspect-oriented software development* Enschede, The Netherlands: ACM Press, 2002, pp. 65-75.
5. Sun Microsystems, *Enterprise JavaBeans (EJB) Specification 2.1*, Web site: http://java.sun.com, 2003.
6. S. Distefano, D. Paci, A. Puliafito, and M. Scarpa, UML design and software performance modeling, in The 19th International Symposium on Computer and Information Sciences (ISCIS'04), Antalya, Turkey. 2004, pp. 27–29.
7. G. Succi, W. Pedrycz, S. Djokic, P. Zuliani, and B. Russo. An empirical exploration of the distributions of the chidamber and kemerer object - oriented metrics suite. *Empirical Software Engineering*, 10(1):81–104, 2005.
8. O. Seng, J. Stammel, and D. Burkhart. Search-based determination of refactorings for improving the class structure of object-oriented systems. In *GECCO '06*, Seattle, Washington, USA, 8-12 July 2006. ACM.
9. M. ´O Cinn´eide, D. Boyle, and I. Hemati Moghadam. Automated refactoring for testability. In *Proceedings of the International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2011)*, Berlin, Mar. 2011.
10. Crnkovic, I. and Grunske, L. (2007). Evaluating dependability attributes of component-based specifications. In ICSE Companion, pages 157-158.