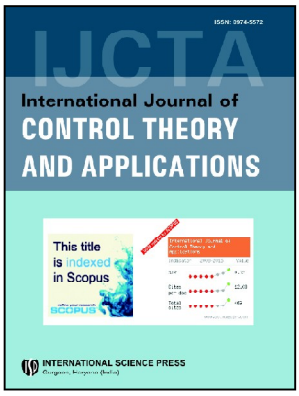

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 9 • Number 50 • 2016

Adaptability Considered Dynamic Reconfiguration System for Component Based Software System

D. Vivek[1*], S. Duraisamy[2] and V. Venkatesa Kumar[3]

[1*]*Assistant Professor, Master of Computer Applications, Tamilnadu college of Engineering, Karumathampatti, Coimbatore*
**Corresponding Author Email: anandhan.vivek@gmail.com*
[2]*Assistant Professor, chikkanna Government Arts College, Tirupur*
E-mail: sdsamy.s@gmail.com
[3]*Assistant Professor, Department of CSE, Anna University Regional Campus, Coimbatore*
Email: mail2venkatesa@gmail.com

Abstract: Component based software development is the most recently growing technology which is utilized by various developers for building the efficient software which satisfies user's runtime requirements. Dynamic reconfiguration would affect the performance of run time tasks in the complex software system where there would present more interdependent modules. Changes made in one module would reflect massive changes in another module which needs to be predicted and keep in track before performing dynamic reconfiguration. In our previous work QoS aware Dynamic Reconfiguration System (QoS-DRS) is introduced for assuring the efficient and better development of component based system. However that system does not focus on interdependency before performing dynamic reconfiguration. For the large software modules it would be more difficult to maintain the QoS metrics. This is resolved in the proposed research by introducing the novel method namely Directed Acyclic Graph based Adaptive Dynamic Reconfiguration System (DAG-ADRS). In this work DAG of the software is constructed in order to maintain the flow level. Thus the various inter linkage analysis between the different modules can be learned well therefore the better performance can be obtained. This method concentrates on the flows present between the different modules present in the software during reconfiguration. And also this method considers additional QoS parameters for assuring the efficient and better development of component based system. This DAG would make ease of software dynamic reconfiguration and also following additional QoS parameters are considered: portability, throughput, capacity and turnaround time. To find the combined effect of QoS attributes in the software components, in this work Pareto Optimal method is used. The weights for the separate QoS parameters are given manually based on user requirements by using which dominant software components would be identified. The overall experimental evaluation of the proposed research method is conducted in the java simulation environment from which it is proved that the proposed research method leads to provide better result than the existing method.

Keywords: Component based software system, Adaptability, QoS metrics, Dynamic reconfiguration, Directed acyclic graph, interdependency

I. INTRODUCTION

Software systems are becoming increasingly complex and providing more functionality [1]. To be able to produce such systems cost-effectively, suppliers often use component-based technologies instead of developing all the parts of the system from scratch[2]. The motivation behind the use of components was initially to reduce the cost of development, but it later became more important to reduce the time to market, in order to meet rapidly emerging consumer demands [3].

At present, the use of components is more often motivated by possible reductions in development costs. By using components it is possible to produce more functionality with the same investment of time and money [4]. When components are introduced in a system, new issues must be dealt with e.g. dynamic configurations, variant explosion and scalability. Some of these issues are addressed with the discipline ComponentBased Software Engineering (CBSE) [5, 6]. CBSE provides methods, models and guidelines for the developers of component-based systems [7].

Componentbased development (CBD) denotes the development of systems making considerable use of components [8]. Although very promising, CBSE is a new discipline and there are many associated problems which remain unsolved. Many solutions can be arrived at, by using principles and methods from other engineering disciplines, such as configuration management [9].

Apart from all these advantages, the most important motivation factor being offered by component-based development approach is its capability to improve and assure quality [10]. Quality is the foremost and the most important feature of any software system. The main concern while using componentbased development approach is to assure the quality of each component being used. The quality of a component can be considered as a guarantee that the software system being developed will be able to carry out the required services [11]. In order to assure the quality of the software product being developed, the concept of quality metrics is being implemented.

The overall organization of the proposed research work is given as follows: In the section 2, detailed discussion about various related research works which are introduced with the goal of achieving efficient dynamic reconfiguration is done. In section 3, detailed discussion about the proposed research methodology which is introduced with the goal of performing dynamic reconfiguration efficiently with the suitable examples and diagrams. In section 4, experimental comparison scenario is given and analyzed in the detail. Finally in section 5, overall conclusion of the proposed research methodology is provided.

II. RELATED WORK

In [12] have represented an approach by which the Quality of service (QoS) for dynamic reconfigurable CBSS can be improved. By dynamic reconfiguration, it can be said that a system can be changed from one configuration to other without shutting down or rebooting the system. The major challenge faced during reconfiguration is to maintain the QoS while reconfiguration. The main aim is to reduce system disruption which can be achieved by using appropriate reconfiguration strategies and testing each strategy by using appropriate reconfiguration benchmark result and evaluating the results at the end.

In [13] suggested that the true benefit of dynamic configuration can only be achieved if it causes minimum disruption to the ongoing application. If the qualities of assurance features are preserved during the dynamic reconfiguration then the system has significant advantage over static reconfiguration system.

In [14] discussed about Component-based decisive architecture for reliable autonomous systems. In order to achieve the quality of component- based software system, there is a need to develop a scalable framework which will assure the safety requirements. The functional and non-functional aspects for decision making strategies are also taken into consideration.

In [15]implemented a systematic approach for developing qualifying system. This workattempts to select the shelf components and remove it for safety critical systems.

In [16] has given an overview about modeldriven quality assurance tools. These tools are used to assure the traceability of existing models and artifacts and also support development phases like software testing and validation.

In [17] described the autonomic trust management for CBSS. Trust is an important issue in the case of CBSS as components and their requirements vary very rapidly. An adaptive trust control model is introduced which can specify, evaluate the trust relationships.

In [18] has discussed the methods by which the testability of component-based software system can be improved. Quality assurance is taken as key research content in this paper.

In [19] represented the advantages of using component based approach for software development.

III. IMPROVED DYNAMIC RECONFIGURATION SYSTEM

Component-based software development uses appropriate off the shelf software components to create softwaresystems. The notion of assembling complete systems out of prefabricated parts is prevalent in many branches ofscience and engineering such as manufacturing. This leads to the creation of prompt and economical products. This is possible because of the existence of standardized components that meet a manufacturer's functional and nonfunctional (quality) requirements. Also, the task of the manufacturer is made much easier because of the presence ofstandardized component catalogs outlining their functional and non-functional attributes.

In this work DAG of the software is constructed in order to maintain the flow level. Thus the many inter linkage analysis between the different modules can be learned well thus the better performance can be obtained. This method concentrates on the flows present between the different modules present in the software during reconfiguration. And also this method considers additional QoS parameters for assuring the efficient and better development of component based system. This DAG would make ease of software dynamic reconfiguration and also following additional QoS parameters are considered: portability, throughput, capacity and turnaround time.

3.1. DAG Construction

Data-flow analysis is a technique for gathering information about the possible set of values calculated at various points in a computer program. A program's control flow graph (CFG) is used to determine those parts of a program to which a particular value assigned to a variable might propagate. The information gathered is often used by compilers when optimizing a program. A canonical example of a data-flow analysis is reaching definitions.

A simple way to perform data-flow analysis of programs is to set up data-flow equations for each node of the control flow graph and solve them by repeatedly calculating the output from the input locally at each node until the whole system stabilizes, i.e., it reaches a fixpoint. This general approach was developed by Gary Kildall while teaching at the Naval Postgraduate School.

It is the process of collecting information about the way the variables are used, defined in the program. Data-flow analysis attempts to obtain particular information at each point in a procedure. Usually, it is enough to obtain this information at the boundaries of basic blocks, since from that it is easy to compute the information at points in the basic block. In forward flow analysis, the exit state of a block is a function of the block's entry state. This function is the composition of the effects of the statements in the block. The entry state of a block is a function of the exit states of its predecessors. This yields a set of data-flow equations:

For each block b:

Out_b = {\displaystyle out_{b}=trans_{b}(in_{b})}{\displaystyle in_{b}=join_{p\in pred_{b}}(out_{p})}$trans_b$ (in_b)

$in_b = join_{p\epsilon\ predb}$

In this,{\displaystyle trans_{b}}$trans_b$is the transfer function of the block$b${\displaystyle b}. It works on the entry statein_b{\displaystyle in_{b}}, yielding the exit stateout_b{\displaystyle out_{b}}. The operation'join' {\displaystyle join}combines the exit states of the predecessor's$p \in pred_b${\displaystyle p\in pred_{b}}of b{\displaystyle b}, yielding the entry state of b{\displaystyle b}.

After solving this set of equations, the entry and/or exit states of the blocks can be used to derive properties of the program at the block boundaries. The transfer function of each statement separately can be applied to get information at a point inside a basic block.

Each particular type of data-flow analysis has its own specific transfer function and join operation. Some data-flow problems require backward flow analysis. This follows the same plan, except that the transfer function is applied to the exit state yielding the entry state, and the join operation works on the entry states of the successors to yield the exit state.

The entry point (in forward flow) plays an important role: Since it has no predecessors, its entry state is well defined at the start of the analysis. For instance, the set of local variables with known values is empty. If the control flow graph does not contain cycles (there were no explicit or implicit loops in the procedure) solving the equations is straightforward. The control flow graph can then be topologically sorted; running in the order of this sort, the entry states can be computed at the start of each block, since all predecessors of that block have already been processed, so their exit states are available. If the control flow graph does contain cycles, a more advanced algorithm is required.

The most common way of solving the data-flow equations is by using an iterative algorithm. It starts with an approximation of the in-state of each block. Out-states are then computed by applying the transfer functions on the in-states. From these, the in-states are updated by applying the join operations. The latter two steps are repeated until it reaches the so-called fixpoint: the situation in which the in-states (and out-states in consequence) do not change.A basic algorithm for solving data-flow equations is the round-robin iterative algorithm:

For i ← 1 to N

Initialize node i

While (sets are still changing)

For i ← 1 to N

Recompute sets at node i

3.2. QOS Attributes Considered

Having defined an abstract component model, along with related reconfiguration concepts, it is possible to introduce the dependent concepts in terms of QoS characteristics and key problems, with related work reviewed according to each QoS characteristic [21]. The alignment of the QoS characteristics is in line with their strength, which means: 1) A QoS characteristic is stronger than all its previous ones and 2) to support a QoS characteristic, its previous QoS characteristics must be supported in advance. In addition, although related work in the same classified group varies in technique, it is assumed that the variation is essentially trivial in terms of the defined QoS characteristics. In this work the following QoS attributes are considered for the dynamic reconfiguration efficiency.

1. **Dependability:**It is a measure of confidence that the component is free from errors. In systems engineering, dependability is a measure of a system's availability, reliability, and its maintainability, and maintenance support performance, and, in some cases, other characteristics such as durability, safety and security [20].

2. **Security:**It is a measure of the ability of the component to resist an intrusion.Software security is the idea of engineering software so that it continues to function correctly under malicious attack. A central and critical aspect of the computer security problem is a software problem [20].
3. **Adaptability:** It is a measure of the ability of the component to tolerate changes in resources and user requirements.Adaptability is to be understood here as the ability of a system (e.g. a computer system) to adapt itself efficiently and fast to changed circumstances. An adaptive system is therefore an open system that is able to fit its behavior according to changes in its environment or in parts of the system itself [20].
4. **Maintainability**: It is a measure of the ease with which a software system can be maintained.The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment [20].
5. **Portability:**It is a measure of the ease with which a component can be migrated to a new environment.Portability is a characteristic attributed to a computer program if it can be used in operating systems other than the one in which it was created without requiring major rework. Porting is the task of doing any work necessary to make the computer program run in the new environment [20].
6. **Throughput:**It indicates the efficiency or speed of a component.If a system identifies end-users by some form of log-in procedure then a concurrency goal is highly desirable. By definition this is the largest number of concurrent system users that the system is expected to support at any given moment. The work-flow of a scripted transaction may impact true concurrency especially if the iterative part contains the log-in and log-out activity.If the system has no concept of end-users, then performance goal is likely to be based on a maximum throughput or transaction rate.
7. **Capacity:**It indicates the maximum number of concurrent requests a component can serve. Its primary goal is to ensure that IT resources are right-sized to meet current and future business requirements in a cost-effective manner. One common interpretation of capacity management is described in the ITIL framework. ITIL version 3 views capacity management as comprising three sub-processes: business capacity management, service capacity management, and component capacity management.
8. **Turn-around Time:**It is a measure of the time taken by the component to return the result.In general, turnaround time (TAT) means the amount of time taken to fulfill a request. The concept thus overlaps with lead time and can be contrasted with cycle time.

Based in this QoS metrics software CPU saturation level is measured for the dynamic reconfiguration process.

3.3. Pareto Optimality

Pareto optimality is a concept in multi-criteria optimization that allows for the optimization of a vector of multiple criteria, enabling all tradeoffs among optimal combinations of multiple criteria to be evaluated. The figure on the top left illustrates a simple case of minimizing two criteria simultaneously (Z_1, Z_2), with the solid line indicating the Pareto optimal frontier (whereby any improvement with respect to Z_1 comes at the expense of Z_2). Each point along that frontier represents a unique model parameterization and/or model structure, so Pareto optimality identifies multiple Pareto optimal solutions. Through this procedure a scientist is able to investigate differences among the multiple optimal solutions that are able to optimize varying combinations of assessment criteria. Pareto optimality originated with the concept of efficiency in economics, and has recently been applied to various problems in ecology.

In proposed algorithm, it stores only non-dominated solutions. It performs a few comparisons to classify points into either dominated set or non-dominated set. Then population is sorted according to the descending order of importance to the first objective value. In this way the solutions which are good in first objective will come first in the list and those having bad value will come in last.First initialize a set S1, for keeping non

dominated solutions only. Then starts with the first solution and add this solution to non-dominated set S1. Since first point is best in terms of first objective so no point can dominate this point in first objective, so it will be non-dominated. Finally it is compared with every other solution of the list with this set S1 and updates this set when it found another non-dominated solution and skip on those solutions which are dominated by any element of the set. For example if solutions in the list are unique in first objective function value, then for second point it requiresneed only one comparison to decide whether this is dominated or non-dominated.

The reason can be explained as follows, this solution can be dominated by only first solution (which is best in first objective).it cannot be dominated by other solutions because its value for first objective function is greater than all solutions except first. Similarly for the third solution itrequires at most two comparisons from first and second point. And for the last point of list it needs to compare this solution to all non-dominated solutions. If solutions in list are not unique in first objective function value, then it has to make certain modification in proposed algorithm. Like it has to check every solution to its immediate successors, if any immediate solution dominates this solution then it has to remove this point from the non-dominated set S1.Finally displays the non-dominated solutions. The proposed algorithm can be executed using the following steps.

1. Sort all the solutions (P1…PN) in decreasing order of their first objective function (F1) and create a sorted list (O)
2. Initialize a set S1 and add first element of list O to S1
3. For every solution Oi (other than first solution) of list O, compare solution Oi from the solutions of S1
4. If any element of set S1 dominate Oi, Delete Oi from the list
5. If Oi dominate any solution of the set S1, Delete that solution from S1
6. If Oi is non-dominated to set S1, Then update set S1 = S1 U Oi
7. If set S1 becomes empty add immediate solution at immediate solution to S1
8. Print non dominated set S1

3.4. Moelling QOS Assurance

Modeling stateless equivalence has two aspects. First, stateshould be transferred by redirecting references to statevariables from the replaced component into the replacingcomponent. If many objects make up the component's state,these objects can be encapsulated into a single wrapperobject. Consequently, state can be transferred by redirectingonly a single reference, i.e., the reference to the wrapperobject. By such encapsulation and redirection, state size isindependent of application and a state transfer can betreated as virtually instantaneous.Second, state sharing between the replaced componentand the replacing component is also applicable to theproblem of unintentional blocking. State sharing requiresthe following constraints:

1. The replacing component should provide backwardcompatibility of the state format for the replacedcomponent.
2. Access to theshared state must be mutuallyexclusive.
3. The application logic permits state sharing.
4. The replacing and replaced components reside onthe same physical machine.

Mutual exclusion is needed for access to the shared stateand ensures atomicity of the state update and non-corruptionof state. However, whether the state is being used bythe old or the new component, processing is contributingto the overall QoS of the system, i.e., maintaining QoSduring the reconfiguration. The third constraint implies thatstate sharing is application dependent. State sharing is notapplicable to every application, but it is assumedthat, it can beused for a variety of applications.Identifier generators ornumber counters are simple examples of components whereapplication logic permits such state sharing. Althoughcomponent state migration is an open issue,the proposed state sharing is applicable to distributedenvironments, where components can be

distributed as longas the replacing and replaced components reside onthe same physical machine. The above constraints can beeasily fulfilled by many domain applications and are morepractical to apply. If state sharing is possible, it can beachieved at the second timefor all stateful components. Because state sharing can beachieved instantaneously by redirecting a single reference,state sharing enables all version v_{new} components to standbyby the second time point and is able to avoid theunintentional blocking problem.

In some circumstances, state sharing may be impossible (ifthe application does not fulfill Constraint 3) and unintentionalblocking could occur. However, if possible, statesharing should be applied, and improved QoS performancecan thus be achieved for reconfiguration. It is assumed thatstate sharing is possible in most cases.

IV. EXPERIMENTAL RESULTS

In this section, evaluation of the proposed research methodologies is done in the java simulation environment in order to predict their fault detection capability. Each work is implemented and simulated under various configuration parameters to know their performance measure values. The comparison is made between the existing research works namely QoS aware Dynamic Reconfiguration System (QoS-DRS) and the proposed research work namely Directed Acyclic Graph based Adaptive Dynamic Reconfiguration System (DAG-ADRS). The performance measures that are considered for evaluating the improvement of the proposedresearch methodologies are, "Sensitivity, Specificity, Precision, and Delay". The comparison results of this performance metrics are illustrated and explained in the following sub sections.

4.1. Sensitivity

The ratio of accurate detection of possibility of occurrence and return as true result is called as sensitivity rate. The mathematical evaluation of sensitivity rate is given as follows:

$$Sensitivity = \frac{Number\ of\ true\ positives}{Number\ of\ true\ positives + Number\ of\ \ false\ negatives}$$

The graphical representation of the sensitivity measurement values of the proposed research methodology is given in figure 1.

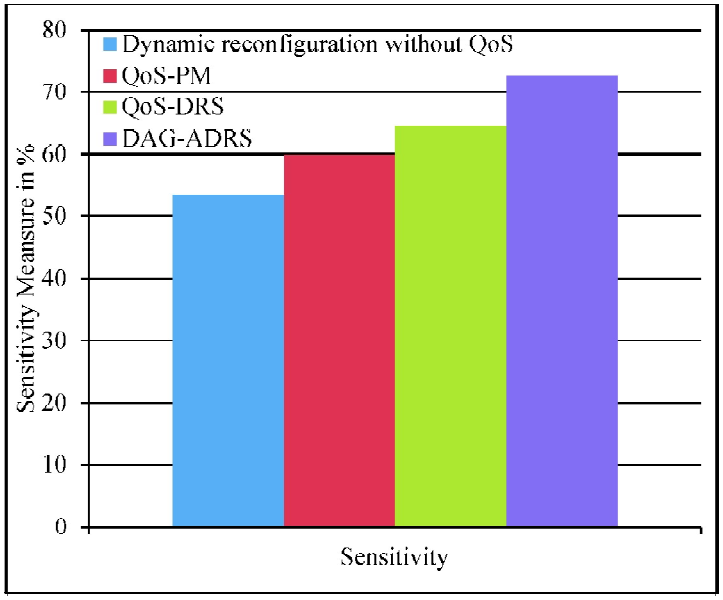

Figure 1: Sensitivity Measure

In figure 1 Sensitivity measure comparisons of the proposed research methodologies is given. This graph proves that the proposed research method can accurately predict the reconfiguration components in the software efficiently with improved performance. It can be concluded that the proposed research method DAG-ADRS provides 19 % improved result than the existing methodologies.

4.2. Specificity

Specificity is defined as the ratio of correct identification wrong outcomes as positives without the presence of any conditions. Mathematical calculation procedure of specificity is given as follows:

$$Sensitivity = \frac{Number\ of\ true\ negatives}{Number\ of\ true\ negatives + Number\ of\ \ false\ positives}$$

The graphical representation of the specificity measurement values of the proposed research methodology is given in figure 2.

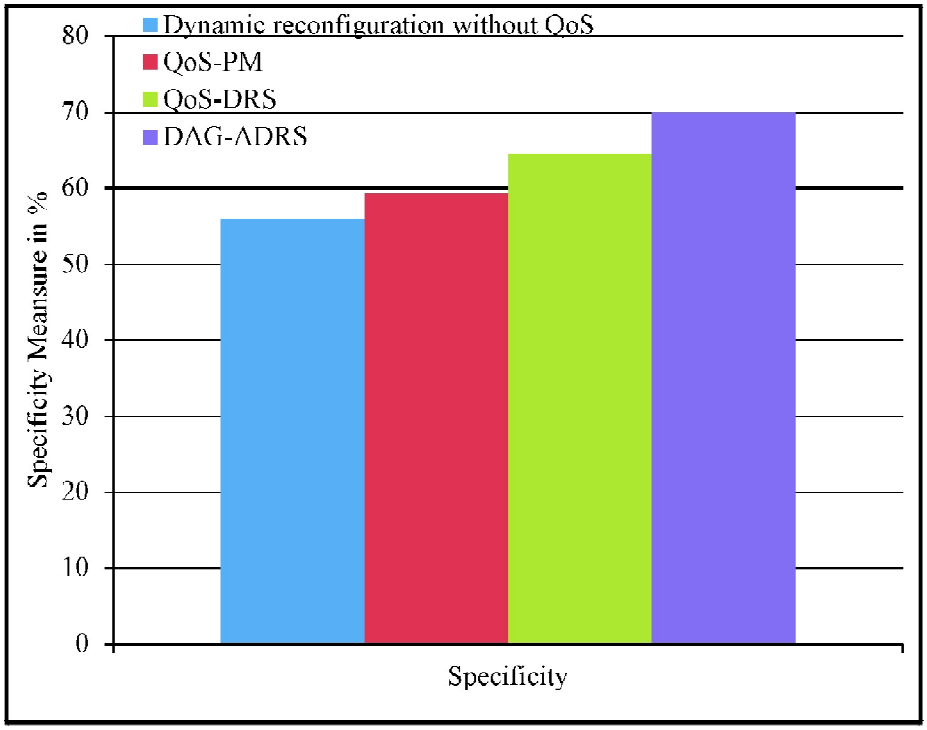

Figure 2: Specificity Comparison

Figure 2 shows specificity measure comparisons of the proposed research methodologies. This graph proves that the specificity metric evaluated provides significant results with 21% improved specificity rate than the existing methodologies. This evaluated results clearly indicates that the proposed research method can accurately predict the reconfiguration components present in the software efficiently with improved performance.

4.3. Precision

Ratio of retrieving relevant outcomes from the available information is defined as precision and the defined outcome. The mathematical calculation procedure of precision is given as follows:

$$Precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{relevant\ documents\}|}$$

Precision takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system.

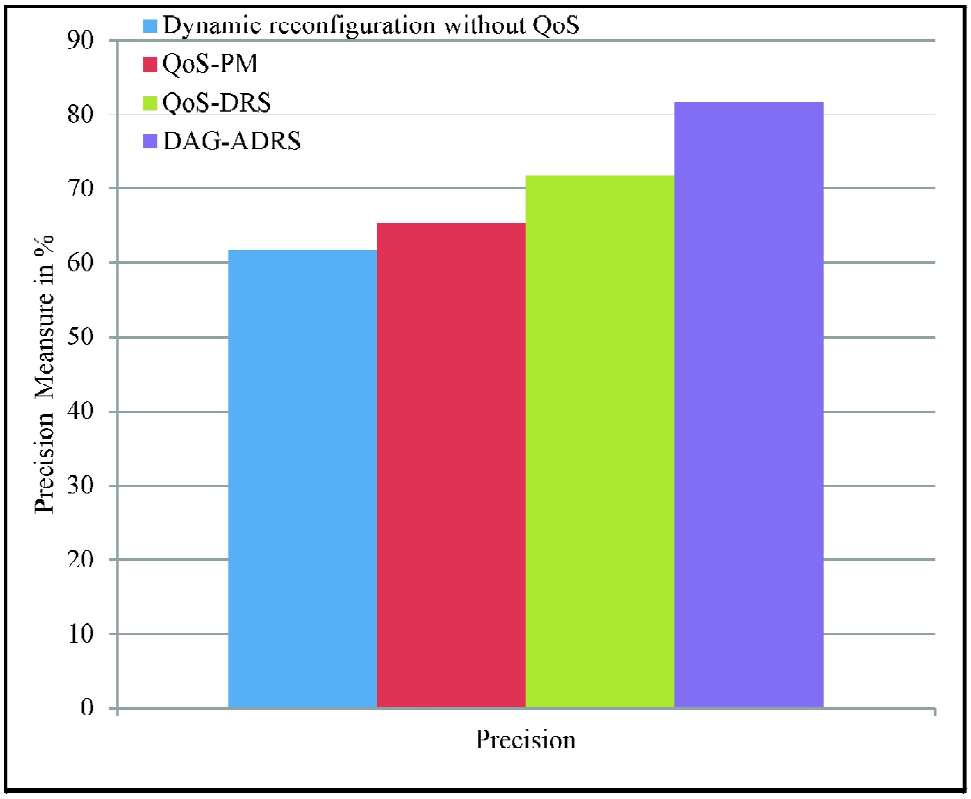

Figure 3: Precision comparison

Precision metric comparisons of the proposed research methodologies are given in figure 3. It is concluded that the proposed method DAG-ADRS provides 34 % performance improvement than the existing methodologies. This is mainly due to the adaptive nature of the proposed approach which results in significant improvement.

4.4. Delay

Amount of time waited to perform and finish dynamic reconfiguration o the software components is defined as delay;

Delay is defined as the time taken to find the number of reconfiguration components present in the software. The delay should be less to lead to efficient software development which can be done on time. The comparison of the delay consumed for proposed research methodologies are illustrated in the figure 4.

The above graph illustrated the comparison graph for the delay parameter value between the proposed and existing research methodologies. It shows 12% performance improvement by performing dynamic reconfiguration process as soon as possible before loading it.

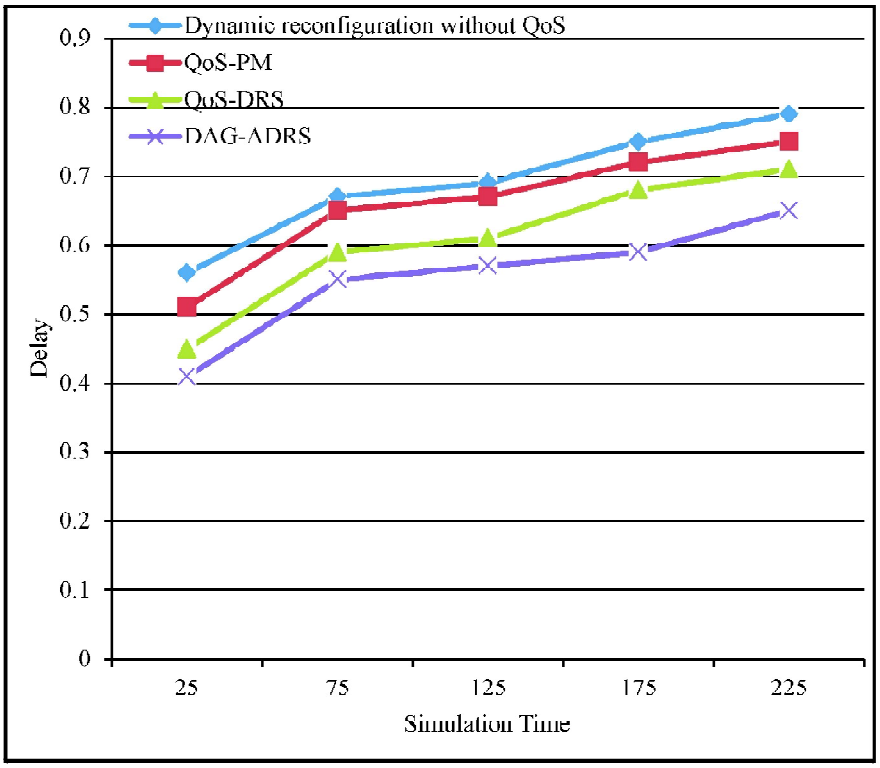

Figure 4: Delay comparison

V. CONCLUSION

This research work has presented a QoS framework for software components for dynamic reconfiguration of component-based software systems. The research under pinning this work has successfully attempted QoS assurance. In the proposed research framework, QoS aware Dynamic Reconfiguration System (QoS-DRS) is introduced for assuring the efficient and better development of component based system. QoS metrics are the units for measuring the QoS attributes of a software component. Quantification of the QoS attributes of software components is one of the important goals of the proposed QoS framework. Hence, there is a need for standardized metrics to compare the QoS attributes of different software components. In the proposed research work, various QoS metrics such as dependability, security, adaptability, maintainability and portability is considered to achieve optimal and better reconfiguration. The overall experimental evaluation of the proposed research method is conducted in the java simulation environment from which it is proved that the proposed research method leads to provide better result than the existing method.

REFERENCE

[1] Fontana, J. A., Iyengar, S. S., Pitchford, A. R., Smith, N. R., & Tolbert, D. M. (2000). U.S. Patent No. 6,167,564. Washington, DC: U.S. Patent and Trademark Office

[2] Mørch, A. I., Stevens, G., Won, M., Klann, M., Dittrich, Y., &Wulf, V. (2004). Component-based technologies for end-user development. Communications of the ACM, 47(9), 59-62

[3] Crnkovic, I. (2005, May). Component-based software engineering for embedded systems. In Proceedings of the 27th international conference on Software engineering (pp. 712-713). ACM

[4] Cai, X., Lyu, M. R., Wong, K. F., &Ko, R. (2000). Component-based software engineering: technologies, development frameworks, and quality assurance schemes. In Software Engineering Conference, 2000. APSEC 2000. Proceedings. Seventh Asia-Pacific (pp. 372-379). IEEE

[5] Crnkovic, I. (2001). Component based software engineering—new challenges in software development. Software Focus, 2(4), 127-133.

[6] Jifeng, H., Li, X., & Liu, Z. (2005, October), Component-based software engineering. In International Colloquium on Theoretical Aspects of Computing (pp. 70-95). Springer Berlin Heidelberg.

[7] Heineman, G. T., & Councill, W. T. (2001), Component-based software engineering (p. 818). Reading: Addison-wesley.

[8] Liu, X., Combe, C., & Wang, B. (2007), U.S. Patent Application No. 11/786, 311.

[9] Crnkovic, I., Chaudron, M., & Larsson, S. (2006, October), Component-based development process and component lifecycle. In Software Engineering Advances, International Conference on (pp. 44-44). IEEE.

[10] Vitharana, P. (2003), Risks and challenges of component-based software development. Communications of the ACM, 46(8), 67-72.

[11] Gill, N. S. (2003), Reusability issues in component-based development. ACM SIGSOFT Software Engineering Notes, 28(4), 4-4.

[12] Reeta, R., &Mariappan, A. K. (2014, April). An approach to assure QoS for dynamically reconfigurable component-based software systems. InCommunications and Signal Processing (ICCSP), 2014 International Conference on (pp. 1353-1357). IEEE.

[13] Li, W. (2012), Qos assurance for dynamic reconfiguration of component-based software systems. Software Engineering, IEEE Transactions on, 38(3), 658-676.

[14] Ramaswamy, A., Monsuez, B., &Tapus, A. (2013, May), Component based decision architecture for reliable autonomous systems. In Collaboration Technologies and Systems (CTS), 2013 International Conference on (pp. 605-610). IEEE.

[15] Esposito, C., Cotroneo, D., Barbosa, R., & Silva, N. (2011, April), Qualification and selection of off-the-shelf components for safety critical systems: A systematic approach. In Dependable Computing Workshops (LADCW), 2011 Fifth Latin-American Symposium on (pp. 52-57). IEEE.

[16] Crelier, O., Roberto Filho, S. S., Hasling, W. M., &Budnik, C. J. (2011, September). Design principles for integration of model-driven quality assurance tools. In Software Components, Architectures and Reuse (SBCARS), 2011 Fifth Brazilian Symposium on (pp. 100-109). IEEE.

[17] Yan, Z., &Prehofer, C. (2011), Autonomic trust management for a component-based software system. Dependable and Secure Computing, IEEE Transactions on, 8(6), 810-823.

[18] Liangli, M., Houxiang, W., &Yansheng, L. (2006, October), The Design of Dependency Relationships Matrix to improve the testability of Component-based Software. In Quality Software, 2006. QSIC 2006. Sixth International Conference on (pp. 93-98). IEEE.

[19] Alvaro, A., Almeida, E. S., &Meira, S. L. (2005). Quality attributes for a component quality model. 10th WCOP/19th ECCOP, Glasgow, Scotland.

[20] D. Vivek, Dr.S. Duraisamy and Dr.V. Venkatesa Kumar, "An improved dynamic reconfiguration system of component based software systems in the pharmaceutical industry", International journal of pharma and biosciences, 2017.

[21] Arun, R., & Mohanasundaram, R. (2012). EGMP: Multicasting Routing Protocol with QoS Improvements. Bonfring International Journal of Research in Communication Engineering, 2(Special Issue Special Issue on Communication Technology Interventions for Rural and Social Development), 98-101.