Analysis and Validation of Software Developement In Distributed Environment

Yin Nyein Aye
Faculty of Information Science
University of Computer Studies (Taunggyi)
yinnyeinaye @ucstgi.edu.mm

Hsu Mon Kyi
Faculty of Computer Science
University of Computer Studies (Taunggyi)
hsumonkyi@ ucstgi.edu.mm

Chan Myae Aye
Faculty of Information Science
University of Computer Studies (Taunggyi)
chanmyaeaye@ ucstgi.edu.mm

Abstract

Software development is a collaborative process where teams of developers work together in order to complete tasks. Developers have to make decisions based on data which can be modified by other developers. To prevent wrong decisions, the system has to make sure that the data is consistent between developers and modifications on one developer are propagated to other developers. Developers can download the source code files that are stored in cloud storage server. And then they modify their source code and upload these update source code file to the cloud storage server to see others developers. However, they can read others developers' source code file but they can't write or modify it. Petri Net Model has been presented to compute control flow complexity for this system. The result can be shown that the nine metric values are upward trends when the number of developers increases in CSCW. This paper focuses on the complexity of proposed consistency control model by using Petri net-based representation. Moreover, an effort has been made to evaluate the control flow complexity measure in terms of Weyuker's properties that is fulfilled by any complexity measure to be suitable for this system.

Keywords— Software Development, CSCW, Complexity, Petri Net

1. Introduction

Collaborative work (CW) means is used mainly in the business settings which are known as computer supported collaborative work (CSCW) to accomplish the software productivity [4]. Nowadays, the implementation of CSCW on cloud is a challenging issue among researchers in various fields such as artificial intelligence, computer science, network communication, distributed systems and so on. CSCW applications may have a huge variation in requirements for reliability and consistency.

Eric Brewer's CAP Theorem [15] says that in a distributed database system, Consistency, Availability and Partition tolerance. In CAP theorem is described only two of the three properties consistency, availability and tolerance to network partitions can be achieved at the same time [1]. Thus, cloud developers have to choose a datastore according to their application requirements. Lowering the consistency level also has often the advantage of better performance and response times than strong consistency. However, choosing a lower level of consistency poses uncertainties for many applications. Maintaining consistency within a single database node is relatively easy for most databases. The real problems start to maintain consistency between multiple database servers [11]. Collaborative Editing allows people to work simultaneous on the same document or source base (e.g., Google Docs, Version control, Wiki's). The main functionality of collaborative editing system is to detect conflict during editing and to track the history of changes. Traditionally these systems work with strong consistency. Most parts of the document which are frequently updated by several users would be handled by strong consistency guarantees to avoid conflicts all together. If a client makes a write operation on server A, we do not make sure that this is consistent with server B, or C, or D. Therefore, distributed shared systems are designed as different consistency models to achieve high performance of operations on shared data.

Software development requires enforcing consistency for providing each individual with the modifications performed by all other developers. Two developers edit a document at the same time and send their changes to the application. To avoid lost updates, Monotonic write consistency model is used for a collection of concurrent processes. By using monotonic write consistency, it is shown that a write operation by a developer on a source code file is completed before any successive write operation on it. If there are two consecutive writes by the developers and a source code file has already written the value of the second write. The collaborative software development is an effort to build a distributed computing platform over the network. Therefore, complexity for real world system is difficult to measure. In this paper Petri net based representation is used to measure complexity of the system based on this concurrency control model and Weyuker's properties to validate the complexity. Section 2 presents literature review, section 3 explains the types of software development models, section 4 the target scenario for computer supported collaborative work and section 5 describes model assumption, section 6 explains control flow complexity metrics for collaborative work, section 7 discuss about Weyuker's properties for this representation. Finally, section 8 addresses the conclusion.

2. Literature Review

Process measurement can and should be used in every phase of the process development life-cycle, including the analysis, design, implementation, testing, and maintenance phases. Control-Flow Complexity (CFC) measure to analyse the degree of complexity of this software development. Process complexity can be defined as the degree to which a software development is difficult to analyse, understand or explain. Nowadays, complexity analysis has an increased importance. Therefore, methods should be used to support the design and redesign of processes to reduce their complexity. The CFC can be used to analyse the complexity of business processes, as well as workflow and Web processes. Computer Supported Cooperative Work (CSCW) is the study of how people use technology, with relation to hardware and software, to work together in shared time and space. Moreover, it aims to provide similar improvements for ‘‘multiple individuals working together in a conscious way in the same production process or in different but related production processes" [16]. Multiple views of software development are required to keep all of the developers' view consistent under change [17, 18]. Several systems have been developed in order to provide better means of communication and awareness over the actions of others. Many distributed collaborative tools are used to support for interaction among developers [19]. Current research in cloud is very active in academia as well as in industry. Most of cloud-based applications does not require at the same level for consistency which is required to complete the task. A variety of applications needs for different consistency level [20, 21, 22]. M. C. Mateus [16] proposed Vector-Field Consistency (VFC) algorithm which relies on two distinct concepts: location-awareness and continuous consistency model based on client-server architecture. In this paper, an effort has been made to evaluate the control-flow complexity measure in terms of Weyuker’s properties. [23, 24]

3. Types of Software Development Models

Software development is complex and relies on decision making. Therefore, software development teams work on critical systems has a very structured process with rapidly changing requirements, a less formal, flexible process is likely to be more effective.

In this section, software development models are described. Each model represents a process from a specific perspective view is shown in Table 1.

Table 1. Types of software development models and how to apply and their advantages and disadvantages

No.	Types of software developme nt models	How to Apply	Advantag es	Disadvantag es
1.	Water Fall Model	Used with projects that have a defined set of require ments	simple and understan dable smaller Project	does not work well for complex projects
2.	V-model	Like as water fall but upward s after the coding	smaller projects	unforeseen changes/upd ates throughout the software lifecycle
3.	Incremental Model	iterativ e and increm ental develop ment	a great solution for some change requests projects	require more resources, staff and monetary, behind the project
4.	RAD Model	modific ation of the Increm ental Model	reduced developm ent time and allows for more customer feedback	limited modelling and planning skills
5.	Agile Model	process adaptab ility and user engage ment with rapidly	decreases the amount of time	relies on end-user interaction
6.	Iterative Model	relies on specifyi ng and implem enting individ ual parts of the softwar e	easy to identify problems early	can take longer and be more costly
7.	Spiral Model	combin es elemen ts of both the Iterativ e and Waterf all develop ment models	more accurate estimates for budget and schedule	requires team members that are well-versed in risk evaluation
8.	Prototype Model	relies on creatin g prototy	reduced time and costs	cause user confusion between prototype and finished

		pes of the softwar e applicat ions or system softwar e		product can add excessive developmen t time

There are most popular types of software development models. But these models are not focus on the complexity of process measurement. Therefore, the case study focuses on the complexity of proposed consistency control model by using Petri net Model. The complexity of collaborative and teamwork processes is connected to effects such as readability, effort, testability, reliability and maintainability of processes. The complexity of a process is also strongly associated with the degree of difficulty a user has to understand and use a process. Therefore, it is important to develop measures to automatically identify complex collaborative and teamwork processes.

Petri nets are a model and tool which may be successfully combined for diverse applications such as performance evaluation, decision support, and training on complex systems. Petri Nets (PN) is a graphical paradigm for the formal description of the logical interactions among parts or of the flow of activities in complex systems. Petri Net is a well-known model to represent the workflow both in business activities and computer systems. It is more concise and can express complex parallel execution behaviours. Therefore, using Petri Net to describe the composite and dynamic behaviours is very suitable. The effectiveness of flow of the system requires modelling, measurement, the estimation of complexity, defects, process size, and effort of testing, time, resources, and quality of service. To achieve an effectiveness of the system, the complexity analysis of system can be used.

4. Target Scenario for computer supported collaborative work

The main objective of this case study is to allow all the developers shared application states and see the same view at the same time. Developers access the data from their source code files and send back to update source code file to the Server site. The requests are divided into two classes: non-modifying operations (reads) and modifying operations (writes). For each developer Read: Developer issues a read request and waits for the read to perform. Write: Developer issues a write request and waits for the write to perform. The developers start their jobs. Then the source code files are modified and send back to the Server site. After getting the synchronization from each developer, the Server site sends back synchronization to developers. They can compare their timestamp and get update new from other developers' update source code file. The collaborative software development is composed of a set of activities, tasks or services put together to achieve the system correctly. Therefore, complexity for real world system is difficult to measure. Therefore, the Petri net-based representation is used to measure complexity for this system. In this model, the places are the activities of the developers and the server and transitions are the operations of the system.

Figure1 illustrates editing and merging approach to collaborative software development. Developer 1 edits data (denoted by A'), Developer 2 edits data (denoted by B') and Developer N edits data (denoted by N'). Therefore, they can modify it at the same time. After updating, developer 1 and 2 keep their update source code file. In distributed environment, the last edit data (A', B',….., N') is sent for other developers to use. The developer 1 can update his source code file and sends back to server site. And also, developer 2 updates his source code file and sends back to server site and developer N also updates. The server site sends back developer 1's source code file to developer 2's site and sends back developer 2's source code file to developer 1's site and sends back to developer N' source code file to developer 1's site and developer 2' s site. The meaning of the symbols for target scenario 1 is shown in Table 2.

Table 2. Meaning of the Symbols for Target Scenario

Symbol	Meaning
A	Developer 1's Source Code File
B	Developer 2's Source Code File
A'	Developer 1' Updated Source Code File
B'	Developer 2' Updated Source Code File

5. Model assumption

In this case study, assume that the developers work concurrently and update their source code files. It can be said that this system gives for the developers as a write access concurrently. After updating their source code files, the clients send synchronization to the server. When the server gets the synchronization from the clients, the server sends synchronization to clients as a periodically manner. Therefore, the communication cost for this system is O (n^2).

The case study focuses on the complexity of proposed consistency control model by using Petri net Model. The complexity of collaborative and teamwork processes is connected to effects such as readability, effort, testability, reliability and maintainability of processes. The complexity of a process is also strongly associated with the degree of difficulty a user has to understand and use a process. Therefore, it is important to develop measures to automatically identify complex collaborative and teamwork processes.

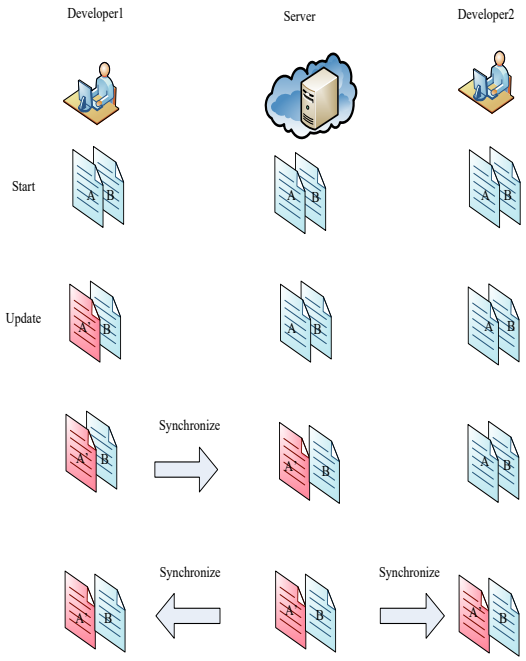

Figure. 1 Target Scenario of Software Development

The Software development is a collaborative process where teams of developers work together in order to complete tasks. Shared documents are replicated at the local storage of each collaborating site, so that operations can be performed at local sites immediately and then propagated to remote sites. The developer 1 and developer 2 start the process concurrently. At first, they receive the old source code file and then they update their own source code files. After updating their source code files, they send synchronization to server to know other developers. After the server has received synchronization from developers, it sends again synchronization to developers. Software development requires getting consistency for providing each individual with the modifications performed by all others developers. Multiple views of software development are required to keep all of the developers' view consistent under change. The requests for operations are handled by developers synchronously and operations are performed by the shared objects sequentially. Developers own their source code files. And then they modify their source code files and send back these update source code files with class file and java file to the server to see other developers. However, they can read other developers' source code files, but they can't write or modify it. To illustrate, the mapping of collaborative work on CSCW concepts onto Petri-net Model in Figure 2 and 3 respectively.

6. Control flow complexity metrics for collaborative work

In this section, the control flow complexity metrics for the collaborative work discussed. Collaboration and group work processes can become highly complex. Process complexity means the degree to which a process is difficult to understand. Using Petri nets-based process modelling can be used to examine the behaviour of the process and to calculate its performance measures.

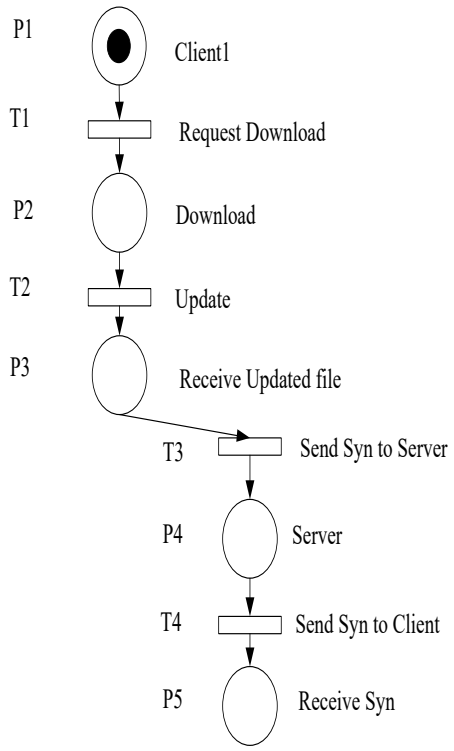

Figure. 2 Petri Net Model for 1 Client

The methods and theory developed have had a reduced industrial acceptance. According to some research, another reason is that there is a lack of serious validation of proposed metrics and a lack of confidence in the measurement. To overcome this difficulty, control flow complexity metric can be evaluated. The following sections are the measurement of the complexity for this system.

6.1 Count-based Measurement

Count-based metric is a basic method to measure the static structure complexity of this system is represented by Petri net.

(1) Number of Places

The total number of places in Petri net-based process, and it reflects the data exchange times in CSCW system. According to the definition 1, it can be easily measured by the following equation.

$$N_p = |P|, \qquad (1)$$

where P is the place set in Petri net. The larger value of N_p is the more frequent data storage, transfer or exchange information in collaborative work.

$$N_p = |\{P1, P2, P3, P4, P5\}| = 5$$

As mention in section, N_p=5 in this system.

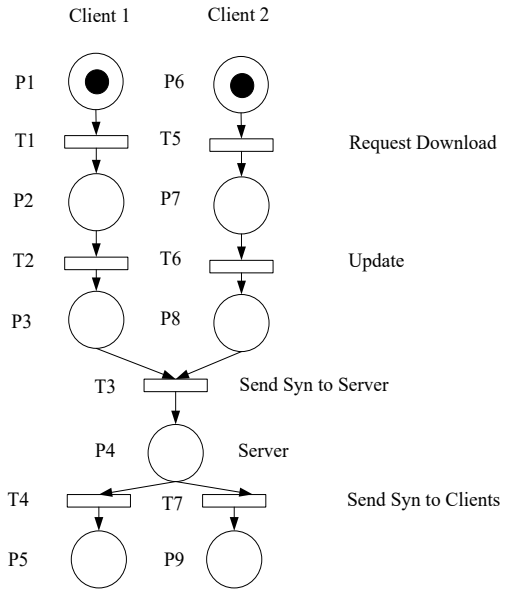

Figure. 3 Petri Net Model for 2 Clients

(2) Number of Transitions

In Petri net-based process, a transition usually stands for an operation or parallel process in collaborative system. Hence, the number of transitions can reflect the activity in the system. It is calculated as shown in equation (2).

$$N_T = |T|, \quad (2)$$

where T is the transition set in Petri net. If a process contains more activities, it must be more complex. Therefore, the Petri-based system with higher N_T means higher structure complexity.

$$N_T = |\{T1, T2, T3, T4\}| = 4$$

N_T = 4 in this system.

(3) Number of Services

In a collaborative system, the number of services directly reflects interaction complexity with the real system. This item can be expressed as follows.

$$N_S = |S| = |\{S\}|, \quad (3)$$

, where $S \subset T$ is work set in collaborative system and s_i refers the specific task used in the Petri based process. The number of service nodes can be defined as transition nodes. These transition nodes are T1, T2, T3, T4 and T5. Therefore, the number of services in the system is 5.

(4) Average Degree of Place (ADP)

The more average degree means the higher interaction strength in this system. The average degree of place (ADP) can be computed by the following equation.

$$ADP = \frac{\sum_i \deg(p_i)}{|P|} \quad (4)$$
$$= \frac{\sum_i [indeg(p_i) + outdeg(p_i)]}{|P|}$$

where $p_i \in P$ is the ith place in Petri net and $\deg(p_i)$ is the degree of node corresponding place p_i in system. It can be divided into two parts: $i(p_i)$ and $outdeg(p_i)$.

$$ADP = \frac{\sum_i \deg(p_i)}{|P|} = \frac{\sum_i [indeg(p_i) + outdeg(p_i)]}{|P|} = \frac{8}{5} = 1.6$$

For this system, ADP can be used to reflect the data transfer complexity. The data interaction in this system is not so complex.

(5) Average Degree of Transition (ADT)

ADT can be yielded according to equation (5)

$$ADT = \frac{\sum_i \deg(T_i)}{|T|} = \frac{\sum_i [indeg(t_i) + outdeg(t_i)]}{|T|}$$

, where $t_i \in T$ is the ith place in Petri net and $\deg(\,)$ is the degree of node corresponding transition in this system.

$$ADT = \frac{\sum_i \deg(T_i)}{|T|} = \frac{\sum_i [indeg(t_i) + outdeg(t_i)]}{|T|} = \frac{8}{4} = 2$$

The ADT value is greater than 2 therefore it means that the system has parallel execution ability.

(6) Transfer Number per Service

The item of transfer number per service (i.e.TNS) can be used as degree and it can be calculated by equation (6).

$$TNS = \frac{|F|}{N_s} = \frac{\sum_i [deg(p_i) + deg(t_i)]/2}{N_s} \quad (6)$$

, where F is a set of directed arcs in Petri net. TNS value of this system can be expressed as

$$TNS = \frac{|F|}{N_s} = \frac{\sum_i [deg(p_i) + deg(t_i)]/2}{N_s} = \frac{8}{5} = 1.6$$

Edges in Petri net-based system represent the data and control logic. The numbers of transfers is used to improve the system. The small amount means the current system has a good structure.

(7) Cyclomatic Complexity

For this system, its cyclomatic complexity can be calculated as below.

$$CC = |F| - |P| - |T| + 2 \text{ (or) } CC = |F| - |P| - |T| + 2p \quad (7)$$

For this system, Petri Net has one start place and one finish place that be calculated like this equation $CC = |F| - |P| - |T| + 2$. If Petri Net has two or more start places and two or more finish places, it can be calculated like this equation $CC = |F| - |P| - |T| + 2p$. p is the number of start or finish places. For this system, CC value is 10-6-5+2= 1. CC > 10 is usually taken as an indicator that a process is excessively complex. In this system, CC value is 1. Therefore, this system is not complex.

6.2Execution Path-based Measurement

In this section address execution path-based metrics to scale the dynamic structure of the system. During the execution system, the computing time is determined by the execution path in this current scenario.

(1) Execution Path

The sequence composed of place nodes and transition nodes is called execution path. According to the notation, these paths can be expressed as below.

$$Path1 = P1 \rightarrow T1 \rightarrow P2 \rightarrow T2 \rightarrow P3 \rightarrow T3 \rightarrow P4 \rightarrow T \rightarrow P5$$

The execution path for developer 1 is represented as Path1.

(2) Execution Path Complexity

The control complexity of P_t can be defined as the sum of complexities of all places and transitions in this system.

$$C(P_t) = \sum_i C(P_i) + \sum_i C(T_j) \quad (8)$$

where P_i is the place node in Path P_t and t_i is transition node in this path. Therefore, the complexities of an execution path are path 1= 11. After getting the complexities of all execution paths, the dynamic complexity can be scaled by average execution path complexity (AEPC). The execution probability of each path can be addressed by $prob\ (Pti)=1/k$, where k is the number of execution paths. In this system =1. Therefore, the probability of Path 1 is 1. Then, the AEPC can be calculated as the following equation (9)

$$AEPC = \sum_{i=1}^{k} prob\ (Pt_i).C(Pt_i) \quad (9)$$
$$= prob(Pt_1).C(Pt_1) + \cdots prob(Pt_k).C(Pt_k)$$

Table 3 shows the complexity weights for the place and transition nodes.

Table 3. Complexity Weights of Key Structure Nodes

Type	Type Name	Basic Structure	Weight
1	Branch	or split	2
		and split	4
		or join	2
		and join	4
2	Iteration	While	3
		repeat Until	3
		for Each	3
3	Concurrency	Flow	4
		join node	4
4	Service Invocation	Service invoking	2

Based on the above analysis for each node's complexity, the complexity of Path 1 can be calculated according to equation (9).

$$C(Path1) = 1 \times 4 + 2 \times 4 + 2 \times 1 = 14$$

The average execution path complexity based on cognitive informatics can be computed as follows:

$$AEPC_{CI} = prob(Pt_1).C(Pt_1)$$
$$= 1 \times 14 = 14$$

The higher the value of path, the more complex is a process design. The analysis results can show reasonably the complexity feature of the system. From the result, this system is not complex. Based on the above analysis for each node's complexity, the complexity of the one developer, two developers, three developers, four developers and five developers in a collaborative software development can be calculated respectively according to the CFC metric equations.

The value of the comparison of complexity metrics is shown in Table 4. From the result in the Table 4, the metrics values for two, three, four, five clients are greater than one developer. So, it has more complex control logic than one developer. The metrics value calculated by these complexity measurement methods can reflect the real system.

Table 4. Value of the comparison of Complexity Metrics for Collaborative Work

No	Metrics	Metric Values for Clients						
		1	2	3	4	5	6	N
1.	N_p	5	9	13	17	21	25	$4k+1$
2.	N_T	4	7	10	13	16	19	$3k+1$
3.	N_s	4	7	10	13	16	19	$3k+1$
4.	ADP	1.6	1.67	1.69	1.70	1.72	1.79	$\frac{7k+1}{4k+1}$
5.	ADT	2	2.14	2.2	2.23	2.25	2.26	$\frac{7k+1}{3k+1}$
6.	TNS	2	2.14	2.2	2.23	2.25	2.26	$\frac{7k+1}{3k+1}$
7.	CC	1	3	5	7	9	11	$2k-1$
8.	AEPC	9	9	9	9	9	9	9
9.	$AEPC_{CI}$	14	14	14	14	14	14	14

7. Weyuker's Properties

Weyuker properties have been applied to software engineering and have been seriously discussed in the literature [22-24]. Weyuker properties are a widely used formal analysis approach and were therefore chosen for some validation of complexity metrics for our system. The nine categories of Weyuker to evaluate software metrics' properties on source code metrics. This system proposed Weyuker' properties to validate the CFC. The following nine Weyuker's properties are described.

Property 1: This property requires that a good metric should be able to classify between two different processes such that they do not return similar measurement results.

- **Property 2:** This property states that a changing process must also cause a change to its complexity.
- **Property 3:** This property states that there exist two different processes whose data types and values are identical but whose variable names differ.
- **Property 4:** This property proclaims that two processes could look identical externally but indeed be different in their internal structure.
- **Property 5:** This property states that two interacting

processes may have zero or additional (but never negative) complexity to that which is present in the two initial processes themselves.

- **Property 6:** This property states that it is possible to have two identical processes, but when concatenated to a third same process, their resulting complexities are not equal. This is an indicator that the action of combining two processes has the potential of leading complexity additional to the original processes.
- **Property 7:** This property states that the order of statements affects complexity i.e., two identical processes can have different complexity when the order of their statements is changed.
- **Property 8:** If two processes differ only in the choice of names for changed structures, then two processes are equal.
- **Property 9:** This property states that interaction between parts of a process cause additional positive complexity i.e., it makes additional complexity a requirement when two processes keep on interacting for some time.

7.1 Analysis of Weyuker's properties

The analysis of Weyuker's properties is shown in Table 5.

Table 5. Analysis of Weyuker's Properties

Property	Property satisfied/dissatisfied by this system
1	Satisfied
2	Dissatisfied
3	Satisfied
4	Satisfied
5	Satisfied
6	Satisfied
7	Dissatisfied
8	Satisfied
9	Dissatisfied

Property 1: Two developers developed by two program say A and B should not be same i.e. the number of linearly independent paths should be vary and therefore it satisfies the property 1.

Property 2: Our CFC measure does not follow this property because it makes no provision for distinguishing between programs which perform very little computation and massive amounts of computation for the same decision structure.

Property3: This property 3 satisfies because two different program say A and B should be same and therefore it satisfies the Nonuniqueness property of Weyuker.

Property 4: Two distinct programs should not be same, even though it should computes the same function, the complexity of programs need not be equal and therefore it satisfies the property 4.

Property 5: The property 5 satisfies the Monotonicity property of Weyuker.

Property 6: Our CFC system satisfies the property 6, let us assume three programs A, B and C, the Weight metrics of A and B should be same and the Weight metrics of A+B should also same as B+C.

Property 7: The property 7 does not satisfy; even though the order of the statements has been changed, the complexity of a program is completely independent of the placement.

Property 8: The property 8 satisfies the Renaming property of Weyuker i.e. if program A has been renamed as B, program won't change.

Property 9: The property 9does not satisfies, let us assume that $\Sigma Ci(A) = 8$, $\Sigma Ci(B) = 5$, $\Sigma Ci(A+B) = 10$, then the $\Sigma Ci(A) + \Sigma Ci(B) < \Sigma Ci(A+B)$; therefore it doesn't satisfies the property 9 of Weyuker.

8. Conclusion

In conclusion, Petri net Model is used to compute the control flow complexity of the collaborative software development on cloud. Cloud computing technology is an effort to build a distributed computing platform over the network. Therefore, complexity for real world system is difficult to measure. Although the cloud computing in real executing scenarios is very complex, it can be represented by the Petri net Model. The use of the Control Flow Complexity (CFC) metric allows users to improve processes because reducing the time spent reading and understanding processes.

The benefits of the CFC metric is that it can be used as a maintenance and quality metric, it gives the relative complexity of process designs, and it is easy to apply. Difficulties of the CFC metric include the incompetence to measure data complexity, only control-flow complexity is measured. The charts can be shown that the metric values for the number of places, the number of transitions, the number of services, average degree of places, average degree of transitions, average degree of services, transfer number of services are upward trends when the number of clients increase. The cyclomatic complexity, average execution path and average execution path with cognitive informatics are different results. These metric values remain steady when the numbers of clients increase. Clients can gain access the system with consistent state at time t. Furthermore, to increase the confidence level of the CFC measure we discussed about to get the suitable validation procedure using Weyuker's nine properties. Since our system happens to fully to satisfy six of the Weyuker's nine properties. Therefore, it can be categorized as a good structured one.

References

[1]D. J. Abadi, "Consistency tradeoffs in modern distributed database system design: CAP is only part of the story", IEEE Computer, 45(2):37–42, 2012.

[2]H. Attiya,V. Gramoli, A. Milani, "COMBINE: An Improved Directory-Based Consistency Protocol", February 10, 2010.

[3]E. A. Brewer, "Towards robust distributed systems. (Invited Talk)", Principles of Distributed Computing, Portland, Oregon, July 2000.
[4]J. Cardoso, "Approaches to Compute Workflow Complexity", Dagstuhl Seminar Proceedings 06291, Dagstuhl, Germany, 16-21 July 2006.
[5]H. Zhu, M. Zhou, "Formalizing the Design of a Collaborative System by Petri Nets", IEEE, ISSN 1062-922X ,6-9 October 2002.
[6]S. Ramaswamy, “Petri Net based Approach for Establishing Necessary Software Design and Testing Requirements”, IEEE Conference on Systems, Man and Cybernetics Nashville, Oct. 2000.
[7]Chengying Mao, Kristian Bisgaard Lassena, Wil M.P. van der Aalst, Complexity Metrics for Workflow Nets,Elsevier August 5, 2008.
[8]Chengying Mao, Control Flow Complexity Metrics for Petri Netbased Web Service Composition,JOURNAL OF SOFTWARE, VOL. 5, NOVEMBER 2010. 2010 ACADEMY PUBLISHER
doi:10.4304/jsw.5.11.1292-1299
[9]Dr George L. Benwell and Dr Stephen G. MacDonell, Assessing The Graphical And Algorithmic Structure Of Hierarchical Coloured Petri Net Models, September 1994
[10]J. Cardoso, "Business Process Control-Flow Complexity: Metric, Evaluation, and Validation", International Journal of Web Services Research, 5(2), pp 49-76, April-June 2008.
[11]J. Cardoso, "About the Complexity of Teamwork and Collaboration Processes ", IEEE Computer Society, ISBN: 0-7695-2050-2, pp. 218-221.
[12]J. Cardoso, "Approaches to Compute Workflow Complexity", Dagstuhl Seminar Proceedings 06291, Dagstuhl, Germany, 16-21 July 2006.
[13]G. Convertino, U. Farooq, M. B Rosson, J. M. Carroll, “Old is Gold: Integrating Older Workers in CSCW”, Proceedings of the 38th Hawaii International Conference on System Sciences, pp. 1-10, 2005.
[14]S. Ramaswamy, "A Petri Net based Approach for Establishing Necessary Software Design and Testing Requirements", IEEE Conference on Systems, Man and Cybernetics Nashville, Oct. 2000
[15]Kristian Bisgaard Lassena, Wil M.P. van der Aalst, "Complexity Metrics for Workflow Nets", Elsevier August 5, 2008.
[16]M. Cortez Mateus, "Vector-Field Consistency for Collaborative Software Development”, June 24 201
[17]Fetai, H. Schuldt, "Cost-Based Adaptive Concurrency Control in the Cloud", Technical Report CS-2012-001.
[18] L. S. González, F. Ruiz, F. García, J. Cardoso, ,"Towards Thresholds of Control Flow Complexity Measures for BPMN models ", SAC’11, March 21-25, 2011, TaiChung, Taiwan. ACM 978-1-4503-0113-8/11/03, Copyright 2011.
[19] S. Kumawat, A. Khunteta, "A Survey on Operational Transformation Algorithms: Challenges, Issues and Achievements", International Journal of Computer Applications (0975 – 8887), Volume 3 – No.12, July 2010.
[20] H. E. Chihoub , S. Ibrahim , G. Antoniu , M. S. Perez, "Consistency in the Cloud: When Money Does Matter",22 November 2012.
[22] Md. A. Islam, S. V. Vrbsky, "Tree-Based Consistency Approach for Cloud Databases", IEEE, pp 401-404, November 30- December 3-2010.
[23] J.Cardoso, Control-flow Complexity Measurement of Processes and Weyuker’s Properties,6th International Enformatika Conference. Transactions on Enformatika, Systems Sciences and Engineering, Vol. 8, pp. 213-218, Budapest, Hungary, October 26-28,2005ISBN: 975-98458-7-3
[24] S.Misra,An Analysis of Weyuker's Properties with Measurement, Theory,28 June 2010
https://www.researchgate.net/publication/286167934