ISSN 2278-5612

Refining Complexity Metrics of Component Based Softwares by taking Average Use Factor in Black Box Testing

Richa Mittal[1], Ashima Singh[2]

[1]Research Scholar, Thapar University, Patiala

richa_mittal1987@yahoo.co.in

[2]Assistant Prof, Thapar University, Patiala

ashima@thapar.edu

ABSTRACT: We propose to compute the complexity metrics of component based software in more justified way by taking considerations of their using frequencies. The complexity metrics calculation of the component Based software's by using black box testing is still not refined. The reason is that the various components are not used by the end users uniformly. Again, use of various components depends upon user to user as per their requirements. So therefore calculating straight forward their complexity on the basis of number of components, their interfaces and data types are not sufficient. We must add the factor of their (AUF) average use factor by the customers of different components. As we know that every algorithm or program have 3 complexity states i.e . a) Best Case b) Average case and c) Worst case. As we know that each and every components of software is not used uniformly by the users. So calculating merely on the basis of their no of components, interfaces and data types predicts only theoretical complexity of that software. If we wish to calculate more justified complexity metrics then we must normalize these components on the basis of their frequency of use in normal routine. As it's quite possible that some modules or components are rarely used by common users. In this case those components hardly influence the complexity of that software. Thus we can reduce significantly the complexity of component based software which was earlier hypothetically calculated very high.

Keywords—AUF(Average Use Factor); Black box testing; STECC approaches; Logic component; I-BACCI Approach. Black Box Component; CBSD; CBSDO; Component; Reusability Software Architecture.

Academic Discipline and Sub-Disciplines

Computer Science & Engineering, Software testing

TYPE (METHOD/APPROACH)

Case Study

ISSN 2278-5612

INTRODUCTION

Component-based software development (CBD) is an emerging discipline that promises to take software engineering into a new era. Building on the achievements of object-oriented software construction, CBD aims to deliver software engineering from a cottage industry into an industrial age for Information Technology, wherein software can be assembled from components, in the manner that hardware systems are currently constructed from kits of parts.

This volume provides a survey of the current state of CBD, as reflected by activities that have been taking place recently under the banner of CBD, with a view to giving pointers to future trends. The contributions report case studies — self-contained, fixed-term investigations with a finite set of clearly defined objectives and measurable outcomes — on a sample of the myriad aspects of CBD.

This paper includes chapters dealing with COTS (commercial off-the-shelf) components; methodologies for CBD; compositionality. We use case studies and complexity metrics formula and multiply it with the assumed usability factor points for each component to calculate the complexity. We can use the statistical techniques for the validity of results. The proposed algorithm can be implemented in any suitable programming language or Case Study. In this case those components hardly influence the complexity of that software. Thus we can reduce significantly the complexity of component based software which was earlier calculated very high.

Proposed work:

We use case studies and complexity metrics formula and multiply it with the assumed usability factor points for each component to calculate the complexity. We can use the statistical techniques for the validity of results. The proposed algorithm can be implemented in any suitable programming language or Case Study.

CASE STUDY

Microsoft Word 2007

For component based Black box complexity calculation I choose a popular MS Office suit :

Macro Components of MS Office 2007 :

The following are the Major or Macro components of MS office 2007 :

a) MS Access (For Database creation)
b) MS Excel (For Accounting spreadsheet)
c) MS Groove (File Sharing on Projects)
d) MS Infopath (Designing Forms)
e) MS One Note (Sharing Information)
f) MS Outlook (Email and Contacts)
g) MS Power Point (For Presentation)
h) MS Publisher (Edit news letter , Brochures)
i) MS Word (Word processor)
j) Digital Certificate for VBA Projects
k) Microsoft Clip Organizer
l) MS Language Setting
m) MS office Designing
n) MS Office Picture Manager

Total Installed space Occupied: 523 MB

MS Office Setup Size 562 MB

On the basis of survey a common user use above different available MS office package as follows:

Using Frequencies by average user (UF)

Table 1

Sl No.	**Macro MS Office S/W components**	Rarely Used	Less Used	Moderate Used	Extensive Used	Continuous Used
1	MS Access (For Database creation)		.2			
2	MS Excel (For Accounting spreadsheet)					.5
3	MS Groove (File Sharing on Projects)	.1				
4	MS Infopath (Designing Forms)		.2			
5	MS One Note (Sharing Information)	.1				
6	MS Outlook (Email and Contacts)			.3		
7	MS Power Point (For Presentation)				.4	
8	MS Publisher (Edit news letter , Brochures)		.2			
9	MS Word (Word processor)					.5
10	Digital Certificate for VBA Projects	.1				
11	Microsoft Clip Organizer		.2			
12	MS Language Setting	.1				
13	MS office Designing		.2			
14	MS Office Picture Manager				.4	
	Sum =3.5	.4	1.0	.3	.8	1.0

Using Frequencies by average user (UF)= Sum/(.1+.2+.3+.4+.5)*14

Or 3.5/(1.5)*14 Or, 3.5/21 = .16

Memory Occupancy of Components: (MOC)

Table 2

Sl No.	**Macro MS Office S/W components**	Memory Occupancy (MB)	Negligible Memory Occupancy	Less Memory Occupancy	Moderate Memory Occupancy	Large Memory Occupancy	Very Large Memory Occupancy
1	MS Access (For Database creation)	28.3					.5
2	MS Excel (For Accounting spreadsheet)	14.9			.3		
3	MS Groove (File Sharing on Projects)	4.30		.2			
4	MS Infopath (Designing Forms)	5.2		.2			
5	MS One Note (Sharing Information)	31.1					.5
6	MS Outlook (Email and Contacts)	9.2		.2			

7	MS Power Point (For Presentation)	14.4			.3		
8	MS Publisher (Edit newsletter, Brochures)	12.0			.3		
9	MS Word (Word processor)	16.2				.4	
10	Digital Certificate for VBA Projects	2.4	.1				
11	Microsoft Clip Organizer	3.2		.2			
12	MS Language Setting	2.1	.1				
13	MS office Designing	8.3			.3		
14	MS Office Picture Manager	8.5			.3		
	Sum =4.3		.2	.8	1.5	.8	1.0

Memory Occupancy of Components: (MOC)= Sum/(.1+.2+.3+.4+.5)*14

Or MOC=4.3/21=0.2

System Recourses Uses: (SRU)

Table 3

Sl No.	**Macro MS Office S/W components**	CPU Uses	Graphics & Multimedia Uses	Main Memory Uses	Buffer memory uses	Network uses
1	MS Access (For Database creation)	.5	.2	.3	.2	.1
2	MS Excel (For Accounting spreadsheet)	.5	.2	.3	.2	.2
3	MS Groove (File Sharing on Projects)	.2	.2	.2	.3	.3
4	MS Infopath (Designing Forms)	.2	.2	.1	.2	.1
5	MS One Note (Sharing Information)	.2	.2	.3	.4	.5
6	MS Outlook (Email and Contacts)	.2	.2	.3	.2	.5
7	MS Power Point (For Presentation)	.3	.2	.3	.3	.2
8	MS Publisher (Edit news letter , Brochures)	.2	.2	.3	.3	.2
9	MS Word (Word processor)	.3	.3	.3	.4	.2
10	Digital Certificate for VBA Projects	.1	.1	.1	.1	.1
11	Microsoft Clip Organizer	.1	.2	.2	.2	.1
12	MS Language Setting	.1	.1	.1	.1	.1
13	MS office Designing	.2	.2	.3	.2	.2
14	MS Office Picture Manager	.3	.2	.3	.3	.3
	Sum=15.7	3.1	2.7	3.4	3.4	3.1

Table Complexity = Sum/Max Weight age

Or , 15.7/(14*.5*5)=15.7/35=.44

Volume of Software Components (VSC):

Table 4

Sl No.	**Macro MS Office S/W components**	Too Few	Few	Moderate	Rich	Super Rich
1	MS Access (For Database creation)					.5
2	MS Excel (For Accounting spreadsheet)					.5
3	MS Groove (File Sharing on Projects)		.2			
4	MS Infopath (Designing Forms)		.2			
5	MS One Note (Sharing Information)	.1				
6	MS Outlook (Email and Contacts)			.3		
7	MS Power Point (For Presentation)					.5
8	MS Publisher (Edit news letter , Brochures)				.4	
9	MS Word (Word processor)					.5
10	Digital Certificate for VBA Projects	.1				
11	Microsoft Clip Organizer	.1				
12	MS Language Setting	.1				
13	MS office Designing			.3		
14	MS Office Picture Manager				.4	
	Sum=4.2	.4	.4	.6	.8	2.0

System Recourses Uses: (SRU)= Sum/(.1+.2+.3+.4+.5)*14= 4.2 / 21=0.2

Table 5

Cumulative Occupied Memory of Installed components (COM)

Sl No.	**MS Office Package**	Total Memory Occupancy (MB)	Negligible Memory Occupancy	Less Memory Occupancy	Moderate Memory Occupancy	Large Memory Occupancy	Very Large Memory Occupancy
1	MS Office 2007	**615 MB**				.4	

Cumulative Occupied Memory of Installed components (COM) =

Weighted Memory Occupancy /.5 = .4/.5 = .8

As we know that complexity depends primarily on following factors:

1) Memory Occupied
2) Execution time or Time to Live (TTL)
3) Resourced uses
4) Number of Components
5) Using Frequencies of components
6) DataType

The first factor is static. As memory occupied is a static feature and its not varies user to user .The second factor is variable user to user depending upon their habit and needs.

However whatever the user's age , education , need or habit to use a software package and its components ; a user can't use all the components uniformly. Thus the execution of such components does not affect the system uniformly and thus we must made sufficient provision to reduce this theoretical complexity calculation.

In the above table we can say that rarely used software components less affects the complexity of software as most of time they only occupies the memory of secondary storage which is available in abundant and not a critical resource .

Complexity Calculation

CCM(BBAU)= ∑(A+B+C+D+E)*E

Or CCM(BBAU)= ∑ (UF+MOC+SRU+VSC+COM) / N

Where the terms stands for Using Component Complexity Metric Black Box Average Use(CCM(BBAU) , Frequencies (UF) , Memory Occupancy of Components: (MOC), System Recourses Uses: (SRU), Volume of Software Components (VSC) and Cumulative Occupied Memory of Installed components (COM). N= Number of Tables Used

Weighted Complexity Calculation: W**CCM(BB)=(.16+0.2+.44+0.2+.8)/N**

W**CCM(BB)=1.80/5=0.36**

Therfore final computed complexity of components of MS office Suit by using black box method is 0.36 which is pretty good and falls within moderate complex software range .

Micro Software Components of MS Office :

MS Office is a jumbo software suit and contains huge number of micro components in each Macro component. Amongst them we choose MS Word which is extensively used amongst all macro components for case study .

MS Word Software Components:

MS word is the biggest bundle of software components available in the package. For simplification we choose some selected components amongst them for case study :

Table 6

Microsoft Word Components

Sl No.	Components of MS Word	Rarely Used	Less Use	Moderate Use	Extensive Used	Extremely Used
1	Find and Replace		√			
2	Status bar					√
3	Menu Bar					√
4	Print			√		
5	Mail Merge		√			
6	Design		√			
7	Layout		√	√		
8	Table Tools		√			
9	Shapes					
10	Word Art			√		
11	Font				√	
12	Formatting				√	
13	Equation Editor	√				
14	References	√				
15	Edit				√	
16	Spelling and Grammar					
17	Help			√		
18	View				√	
19	Translate	√				
20	Look Up	√				
21	Hyperlink	√				
22	Auto fit		√			
23	Page layout			√		
24	Symbols			√		
25	Workspace					√
26	Save / Save As					√

We can use the same process for the micro software components for complexity computations in black box model.

Advantage of WCCM(BBAU) over othe available available blackbox component based software complexity calculation methods:

Advantage of W**CCM(BBAU)** Component Complexity Metric (Black Box Average Use method) is more realistic and simpler than all available complexity determining methods in black box methods. Its include the factor of Average Use factor which earlier ignored . This reduces significantly the complexity computation than others who calculates theoretically without considering end user requirement.

CONCLUSIONS

The earlier techniques ignores the usability factor of the different software components which are the most major aspect and varies user to user . We propose to compute the complexity metrics of component based software in more justified way by taking considerations of their using frequencies. The complexity metrics calculation of the component Based software's by using black box testing is still not refined. The reason is that the various components are not used by the end users uniformly. Again, use of various components depends upon user to user as per their requirements. So therefore calculating straight forward their complexity on the basis of number of components, their interfaces and data types are not sufficient. We must add the factor of their average use by the customers of different components. As we know that every algorithm or program have 3 complexity states i.e . a) Best Case b) Average case and c) Worst case. As we know that each and every components of software is not used uniformly by the users. So calculating merely on the basis of their no of components, interfaces and data types predicts only theoretical complexity of that software. If we wish to calculate more justified complexity metrics then we must normalize these components on the basis of their frequency of use in normal routine. As it's quite possible that some modules or components are rarely used by common users. In this case those components hardly influence the complexity of that software. Thus we can reduce significantly the complexity of component based software which was earlier hypothetically calculated very high.

Although the CBSD is increasingly being adopted for software development, because of its advantages like reduction in development effort, time and cost , increase in quality with many others. But it still consists of a number of issues or factors that affect many aspects of CBSD like integration and testing effort and affect the produced software cost, maintainability, quality etc Many of these factors or issues can be resolved by using the independent components or the

components with low coupling. Thus a technique has been proposed for the component based software development organizations to be followed during the component based software development. Thus it will help in reducing the integration effort, testing effort, cost and providing the more maintainable and good quality software system. After adding the component using frequencies we can more justify the complexity metrics.

REFERENCES

[1] Navneet Kaur, Ashima Singh, "A Complexity metric for black box components : International Journal of Soft computing and Engineering , Volume-3, Issue-2, May 2013.

[2] Egon Valentini, Gerhard Fliess and Edmund Haselwanter,"A Framework for Efficient Contract-based Testing of Software Components". Proceedings of the 29th Annual International Computer Software and Applications Conference [COMPSAC'05], IEEE, 2005, p.219-222.

[3] Fevzi Belli and Christof J. Budnik. "Towards Self-Testing of Component Based Software". Proceedings of the 29th Annual International Computer Software and Applications Conference [COMPSAC'05], IEEE, 2005,p.205- 210.

[4] Sami Beydeda. "Research in Testing COTS Components Built-in Testing Approaches".3rd ACS/IEEE Conference on Computer Systems and Applications, 2005 IEEE, 2005, p.101.

[5] Chengying Mao."Built-in Regression Testing for Component-based Software Systems".31st Annual international computer software and Applications Conference [COMPSAC 2007], IEEE, 2007, p.723-728.

[6] Huaikou Miao, Shengbo Chen, Huanzhou Liu and Zhongsheng Qian," An Approach to Generating Test Cases for Testing Component-based Web Applications.Workshop on Intelligent Information Technology Application". IEEE, 2007,p.264-269.

[7] Jiang Zheng, Laurie Williams, Brian Robinson and Karen Smiley. Regression Test Selection for Black-box Dynamic Link Library Components. Second International Workshop on Incorporating COTS Software into Software Systems: Tools and Techniques [IWICSS'07]. IEEE, 2007, p.9.

[8] Navneet Kaur, Ashima Singh, "Generating More Reusable Components while Development: A Technique, International Journal of Innovative Technology and Exploring Engineering , Volume-2, Issue-3, February 2013.

[9] W. B. Frakes and K. C. Kang, "Software reuse research: status and future," IEEE Transactions on Software Engineering, vol. 31, pp. 529-536, 2005.

[10] M. Morisio , "Success and Failure Factors in Software Reuse," IEEE Transactions on Software Engineering, vol. 28, pp. 340-357, 2002.

[11] Dan Laurenţiu Jişa , “Component Based Development Techniques – comparison ,” International Conference on Computer Systems and Technologies – CompSysTech,2004.

[12] B. Weide , "Reusable software components," Advances in computers, vol. 33, pp. 1-65, 1991

About Authors

Richa Mittal received her three years polytechnic diploma from Thapar Polytechnic patiala. She received her B.tech degree in Computer Science and Engineering from Thapar university patiala. Currently she is persuing her degree for Master of Engineering (M.E) in Computer science and Engineering from Thapar University,patiala,India. Her research interests include Software testing, software engineering, software’s, software complexity, software components, Software reuse, Software architecture, software process reengineering, and software metrics.

Ashima is Assistant Professor in computer science and Engineering department, Thapar University , Patiala, India and pursuing Ph.D in computer Science from Bathinda , India(2001).she obtained her Master of Technology degree in Computer science from Department of Computer science and Engineering , Punjabi university Patiala India(2005).Her research interests include Software Process Reengineering , Software engineering ,Agile Software Development, Software quality improvement in small scale interprises, software Reuse, software Process customization and Automation and software Process metrics.