Design of Component Oriented Metric to Measure Effort during Software Modules Development

Sh. Ashok[1], Dr. Vijay Deep Gaur[2]

Abstract

In the context of software effort estimation [1], system sizes the taken as a main driver of the system development effort. But other structure design properties, such as coupling, cohesion have been suggested as additional factor. In this paper, using effort data from component oriented development project [2,3], we empirical investigate the relationship between component size and effort for a component and with additional impact structural properties such as connectivity, component interfacing have an effort. This paper can be used as a practical analysis, repeatable and accurate analysis procedure to investigate relationship between component properties and development effort.

Keyword

Component oriented development, COM, DCOM, Traditional Methodology, Effort Estimation.

Introduction

Component based software development [2,3] is a dream of the software industries, where programmers would become merely assembly workers and development process of a new software system would be similar to assembling. And it is demand of today software market because today software project is becoming more and more complex and is hard to manage and control.

In this paper here we will introduce new paradigm for software development as well as provided metric for effort estimation, that will improve the complexity of component, dependency and composite of component based software development. With the help of metrics, a bottom-up measuring process from component to the system can full fill evolution for component oriented software development complexity. The purpose of metrics is characterized with the simplicity, reusability, portability, maintainability etc.

The idea behind component based software development approach is, develop software system by selecting appropriate off-the-shelf component and then to assemble them a well defined software architecture. It is new approach in software engineering community. The purpose of component based software engineering is to develop large system that incorporate previously developed or existing component, thus cutting down an development time and cost. It can also reduce maintenance associated with the upgrading of large system.

It has been proven that software complexity is one of the major contributing factors to the cost of developing and maintaining software. Meanwhile, effort estimation is one of critical factor that directly affect the reusability, portability, reliability and maintainability.

In component based software development the architecture complexity is mainly attributable to the dependencies between component, such as procedure call, message passing and conversation protocol. Here we will introduce component based metrics that will directly affect on the interface among component and component interface is the key factor of component complexity.

1 Research Scholar, Department of Computer Science, Shri Venkateshwara University, Gajraula, India, ashokrtk@gmail.com

2 Assistant Professor, Government College Krishan Nagar, M.Garh, Haryana, India vijaydeepgaur83@gmail.com

Literature survey

CBSE embodies the “the ‘buy, don’t build’ philosophy". CBSE is aiming at realizing long-waited software reuse by changing both software architecture and software process. Because of the extensive uses of components, the Component-Based Software Engineering (CBSE) process is quite different from that of the traditional waterfall approach. CBSE not only requires focus on system specification and development, but also requires additional consideration for overall system context, individual components properties and component acquisition and integration process. This work presents an indicative literature survey of techniques proposed for different phases of the CBD life cycle. The aim of this survey is to help provide a better understanding of different CBD techniques for each of these areas [4].

Purposed Work

Here we will measure the effort of software project that is to develop based on component technology, such as COM [8, 9] /DCOM [11]. COM/DCOM is general architecture for component software. It will define how component and their client interact directly and dynamically. DCOM is a protocol that enables software components to communicate directly over network [10]. These are designed for use across multiple network transports, including internet protocol such as HTTP.

COM AND DCOM HAVE PROVIDED a foundation for building component-based applications. Although they were initially available only on Windows platforms, the ongoing porting efforts to all major versions of Unix and mainframes (11) might turn COM/DCOM into a major cross-platform integration tool. The next generation of COM, called COM+, aims at simplifying the construction of COM applications by providing support in languages and tools and by providing a set of essential object services.

Component Based Effort Estimation Metrics[15,16]

1. **Component Effort (CE) Metric[18,19]:** Estimated elapsed time taken to structure application (hrs)

$$CE = e + b_1 DESIGN\_TOOL + b_2 PROG\_EXP + b_3 TEAM\_SIZE + b_4 PROG\_COMP + b_5 LANG\_EXP + b_6 TYPE\_EXP$$

e - effort man-hrs, spent by programmer to develop application software

DESIGN_TOOL – this is to variable measure the level of productivity tool used by programmers in designing software. Using good designing tool, the productivity ratio of programmer is high. It is very important tool by allowing programmers to use to clarify end user’s requirements at the early stage of software development life cycle. This variable is measured using a five point liker-like scale ranging from (1) very low productivity to (5) very high productivity

PROG_EXP – this variable is to measure the experience of programmers in analyzing and designing application software in computer industries. The measurement, we use to count the number of years that a programmer who has been developing application software in company. The higher number of years of the service is in industry, the more working experience, he has. For this variable, we take average of years of experience among team members for each software project.

TEAM_SIZE – this variable is to measure number of programmer working in a team in analyzing and designing software project. For this variable, the number of programmers assigned to analyze and design the software projects is collected, according to the records of company.

PROG_COMP – this variable is to measure the level of program complexity delivered. Determination of program complexity, at the early stage of software development life cycle is under the control of programmer

LANG_EXP – this variable is to measure level of working experience of programmer, who is in specific kind of programming language. The development time and effort are reduced substantially, if programmer is an experienced one.

TYPE_EXP - this variable measure type of experience based on project type

2. **Component Interlinking Effort (CIE) Metric:** Estimates elapsed time taken to interlink component to build component structure (hrs)

$$CIE = \frac{\text{Number of interlink component}}{\text{Total number of component}}$$

3. **Component Interface Planning (CIP) Metric:** Estimated elapsed time taken to plan component's interface (hrs)

$$CIP = e_1 + e_2 + e_3$$

e_1 – Interface Analysis
e_1 – Interface design
e_1 – Interface Development

Component Interface Building (CIB) Metric: Estimated elapsed time taken to implement component interface (hrs)

Effort estimation, here takes place, top-down or bottom-up based on Component implementation. However, bottom-up is better choice then top-down.

CIB, is determined by using deliverable COM/DCOM in software Application, the value of deliverable component are given below:

Deliverable (COM/DCOM)	Very Low	Low	Medium	High	Very High
Report	4	8	16	32	64
Interface	24	48	96	192	384
Conversion	24	48	96	192	384
Enhancement	4	8	16	32	64
Form/Screen	8	16	32	64	128

4. **Component Testing Effort (CTE) Metric:** Estimated elapsed time taken to test all links in component (hrs)

	←		Effort in	PH →	
Test Case Id	Test case Description	Best Case	Worst Case	Normal Case	Expected
1	**Set up Test Environment**				
1.2	Check Test Environment	1	2	1.5	1.500
1.3	Install Screen Reorder	0.75	1.5	1	1.042
1.4	Insure Defect Reporting Mechanis	1.25	3	2	2.024

	m				
5	**Login Screen on IE**				
5.1	Correct Login	0.05	0.2	0.1	0.108
5.2	Wrong id and Correct Password	0.07	0.2	0.1	0.112
5.3	Correct Id and wrong Password	0.07	0.2	0.1	0.112
5.4	Forgot Password Functionality	0.15	0.3	0.2	0.208
6	**Login Screen on Fire fox**				
6.1	Correct Login	0.05	0.2	0.1	0.108
6.2	Wrong id and Correct Password	0.07	0.2	0.1	0.112
6.3	Correct Id and wrong Password	0.07	0.2	0.1	0.112
6.4	Forgot Password Functionality	0.15	0.3	0.2	0.208
	Total Effort Estimate	3.680	8.300	5.500	5.663

Total Effort 0f Component Oriented Software Development is:

Total Effort = CE + CIE + CIP + CIB + CTE

Effort Estimation Based on Meta mata Metrics [19, 20, 21]

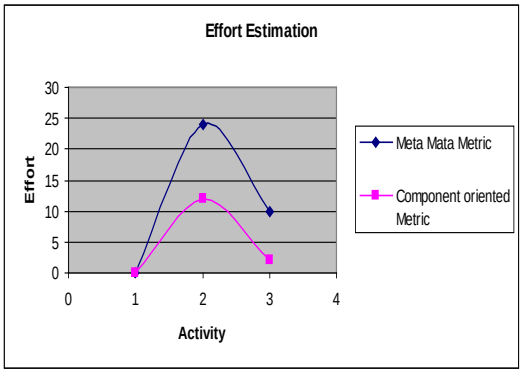
Metric	Measure	Description
CC	Complexity	The amount of decision logic in Code
LOC	Understandability Maintainability	The length of code; related metrics measure Line of comment; effective line of code
WMC	Complexity Understandability Reusability	The number of methods in class
RFC	Design Usability Testability	The number of methods that can be invoked From a class through message
CBO	Design Reusability Maintainability	The of other class to which a class is coupled
DIT	Reusability Testability	The depth of a class within the inheritance Hierarchy
No. of Attributes	Complexity Maintainability	The Amount of state a class maintain as represented By the number of fields declared in the class

Result

Comparison of effort estimation of software project, that is measured based on meta mata metrics that is used in traditional software, with

component oriented software metrics that we have designed in this paper.

Sr No	Major Activity	Effort Estimation Based on Meta mata Metrics	Effort Estimation Based On Component Oriented Metrics
1	**HMS Staff**	46	17.677
2	**Emergency**	57	23.003
3	**Enquiry**	19	6.7
4	**OPD**	36	19.009
5	**Managing Unit Doctor**	63	32.123
6	**Examination**	39	29.123
7	**Nurse Detail**	31	16.23
8	**Patient Status**	28	14.002
9	**Pharmacy/Drug**	49	36.023
10	**Laundry**	24	9.8
11	**Kitchen**	12	2.006



Conclusion

The component oriented software project is implemented based on Microsoft technology such as COM/DCOM. Here we have been designed component oriented metrics that are used to determine effort of component oriented software. These metrics are designed in such a way that it will reduces more than 64 percentage effort of software as compared to meta mata metrics that are used to determine effort of traditional software development..

REFERENCES

1. A. M. Zaremski and J. M. Wing, “Specification matching of software components”, ACM Transactions on Software Engineering & Methodology, 6(4):333-369, October 1997.
2. X.Cai, M.R. Lyu, K. Wong, Component-Based Soft-ware Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes, Pro-ceedings APSEC 2000, Seventh Asia-Pacific Software Engineering Conference, Singapore, December 2000, pp 372-379.
3. 3.X.Cai, M.R. Lyu, K. Wong, Component-Based Soft-ware Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes, Pro-ceedings APSEC 2000, Seventh Asia-Pacific Software Engineering Conference, Singapore, December 2000, pp 372-379
4. D. Box, Essential COM, Addison-Wesley, Reading, Mass., 1998.
5. G.C. Hunt and M.L. Scott, A Guided Tour of the Coign Automatic Distributed Partitioning System, Tech. Report MSR-TR-98-32,Microsoft Research, Redmond, Wash., 1998.
6. Y.M. Wang and W.J. Lee, “COMERA: COM Extensible Remoting Architecture,” Proc. COOTS ’98: Fourth USENIX Conf. Object-Oriented Technologies and Systems, Usenix, Berkeley, Calif., 1998,pp. 79–88; http://www.research.microsoft.com/~ymwang/papers/COOTS98CR.htm.
7. George T. Heineman and William T. Councill, “Component-Based Software Engineering

Putting the Pieces Together", Addison-Wesley, Boston, MA ,880, June 2001.

8. "The Component Object Model Specification," Microsoft Corp., Redmond, Wash.,1995; http://www.microsoft.com/com/comdocs.htm.
9. "COM+," Microsoft Corp., 1998; http://www.microsoft.com/com/complus.htm.
10. "DCE 1.1: Remote Procedure Call Specification," The Open Group, Cambridge, Mass., 1997; http://www.rdg.opengroup.org/public/pubs/catalog/c706.htm.
11. N. Brown and C. Kindel, "Distributed Component Object Model Protocol—DCOM/1.0," Microsoft Corp., 1998; http://www.microsoft.com/com/.
12. Don Box, Essential COM, Addison Wesley, 1998."Microsoft Transaction Server," Microsoft Corp., 1998; http://www.microsoft.com/com/mts.htm.
13. "Millennium: Self-Tuning, Self-Configuring Distributed Systems," Microsoft Research, 1998;http://research.microsoft.com/sn/Millennium
14. M. Kirtland, "Object-Oriented Software Development Made Simple with COM+ Runtime Services," Microsoft Systems J., Vol.12, No. 11, Nov. 1997, pp. 49–59.
15. 16. K. P. Norris, "The Accuracy of Project Cost and Duration Estimates in Industrial R&D", R&D Management, Vol. 2(1), pp.25-36. 1971.
16. P. A. Murmann, "Expected Development Time Reductions in the German Mechanical
17. Engineering Industry", Journal of Product innovation Management, Vol, 11, pp.236-252, 1994
18. 18. K. P. Norris, "The Accuracy of Project Cost and Duration Estimates in Industrial R&D", R&D Management, Vol. 2(1), pp.25-36. 1971. [2] P. A. Murmann, "Expected Development Time Reductions in the German Mechanical Engineering Industry", Journal of Product innovation Management, Vol, 11, pp.236-252, 1994.
19. 19 A. J. Albrecht, "Measuring Application Development Productivity", Proceedings of the IBM Applications Development Symposium, pp83-92, 1979.
20. 19. J. W. Bailey and V. R. Basili, "A meta-model for software development resource expenditures," in Proceedings of IEEE International Conference on Software Engineering, San Diego, California, 1981, pp. 107–116.
21. 20 H. K. N. Leung, "Estimating maintenance effort by analogy," Empirical Software Engineering, vol. 7, no. 2, pp. 157–175, Jun. 2002.
22. 21 A. J. Albrecht, "Software Function, Source Lines of Code, and Development Effort Prediction", IEEE Transactions on Software Engineering, Vol. 9(6), pp639-648, 1983.