www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 5 May 2017, Page No. 21245-21257
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i5.11

Intelligent Method for Cost Estimation during Software Maintenance

Durga Puja, Raghav Mehra, BD Mazumdar

Dept. of CSE
Bhagvant University Ajmer,Rajasthan
durgapuja10@gmail.com
Dept. of CSE Bhagvant University
Ajmer,Rajasthan
raghav.mehrain@gmail.com
Dept. of CSE ICST SHEPA
Varanasi,UP
bireshwardm@gmail.com

ABSTRACT

Software maintenance is very important and time consuming task which require a lot of parameters to be taken care during the whole cycle of maintenance of software. The software maintenance engineer has used various cost estimation models in their research work like COCOMO-I, COCOMO-II, SLIM, SEER-SEM and FP. The problem with these models is that they require the complete data and information making any decision and benchmark. Therefore the cost of software maintenance varies with types and complexity of the software under the maintenance. We have identified the various factors which effects the cost of the software maintenance. The various cost estimation models require a lots of metrics and only after calculating the complete parameters, then only the software cost can be determined.

To reduce the software maintenance cost the proposed model has been introduce which can overcome the cost even there is missing information. The Probability factor (PF) method has been used which can overcome the cost. The proposed model used the various factors as input and accordingly the probability of each factors has been calculated.

- Some of quality factor (in different software) can't be computed
- Sometimes cost of compute all of quality factors are very high
- We need long time for compute all of quality factors
- In some cases we need to a lot of people (user, quality manager, member of quality assurance's team) to compute quality factors

Some other researchers have also deal with uncertainty in quality factor and matrices. Motameni et al. (2010) proposed a model for software quality with BNs and ISO9126 quality model. There model doesn't have last quality model's problems and can predict software quality with incomplete and uncertain data. Also by this model the time and cost of calculating software quality is reduced.

1. Introduction

Software maintenance is the very crucial activity which requires a lot of time and cost. Software cost estimation models attempt to estimate the effort required to develop a software project more accurately by using a mathematical formula of expected project inputs. Boehm’s maintenance model consists of three major phases: understanding, modifying and validating the software [93]. Software maintenance is the recurring process of any software product during its life cycle [Mishra]. When a scratch of the new software products comes into the human mind, the maintenance aspects should also come with its significance. In this paper we have identified the software cost estimation factors, which can effects the cost of software maintenance if during development these factors taken care. . From the literature review we have identified various factor which affects the cost of program comprehension if they are taken during software development.

2. Problem Description

The software maintenance activity is the most time consuming activity and even 50-60% time is consumed in program comprehension [93]. Till date there are no empirical models and theories which can estimate the cost of maintenance. In the review of program comprehension, the various cost factors has been identified. These factors are the most significant, while going for development of software application and software maintenance.

We have created a problem description table (Table 1) for the cost estimation on the basis of three important parameters: human factor, cognitive factors and technical factors. The human factors are Bugs (BG), Cloned Code (CC), Dead Code (DC), Re-Documentation (REDUC), Total Experience (TEXP), Language Experience (LEXP), System OS Experience (SYSEXP), Hardware Experience (HWEXP), Programmer ability (PAVA) and Education (EDUC). The Cognitive factors are Attitude of programmer (AP), Techniques used (TU), Comment to code ratio (CTC) and Number of changes (NOC). The technical factors are Complexity (CR), Line of Code (LOC), Code relations (CR), Concern Attributes (CA), Depth of inheritance Tree (DIT), Response for a Class (RESCL), Weighted Method per class (WEPM), Message Passing Coupling (MEPM), Lack of Cohesion in methods (LOCM), Data Abstraction Coupling (DAC), Number of operators and operands (NOOP), Task Magnitude (TASK), Time (TIME), , Maintainability (MT), Level of understandability (LOU), Cyclcomatic Complexity (CMC) and Tool used (TLU).

Maintenance Metrics/Factors	BG	CC	DC	REDUC	LOC	CR	COA	DPINH	RESCL	WEPM	MEP	LOCM	DAC	NOOP	TEXP	LEXP	SYSEXP	HWEXP	PAVA	EDUC	TASK	TIME	ECUS	MTC	LEVUND	CC	NOC	CMC	TLU
complexity	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Experience of Programmer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
Task	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	0	0	0	0	0	0	1
Maintainability	0	0	0	0	1	0	1	1	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1	1	1	0	0

Table-1 Problem Description table

SNO.	Factor Affective PC	Cognitive factor	Human factors	Technical factors
1	Bugs [86]		**Y**	
2	Colned Code [86]		**Y**	
3	Dead Code [86]		**Y**	
4	Re-Documentation [86]		**Y**	
5	Complexity [86]			**Y**
6	Attitude of programmer [89]	**Y**		
7	Line of Code [89]			**Y**

DOI: 10.18535/ijecs/v6i5.11

8	Code relations [89]			**Y**
9	Concern Attributes [89]			**Y**
10	Depth of inheritance Tree [89]			**Y**
11	Response for a Class [89]			**Y**
12	Weighted Method per class [89]			**Y**
13	Message Passing Coupling [89]			**Y**
14	Lack of Cohesion in methods [89]			**Y**
15	Data Abstraction Coupling [89]			**Y**
16	Number of operators and operands [86]			**Y**
17	Total Experience [86]		**Y**	
18	Language Experience [87]		**Y**	
19	System OS Experience [87]		**Y**	
20	Hardware Experience [87]		**Y**	
21	Programmer ability [87]		**Y**	
22	Education [87]		**Y**	
23	Task Magnitude [86]			**Y**
24	Time [89]			**Y**
25	Techniques used [89]	**Y**		
26	Maintainability [89]			**Y**
27	Level of understandability [87]			**Y**
28	Comment to code ratio [89]	**Y**		
29	Number of changes [89]	**Y**		
30	Cyclcomatic Complexity [86]			**Y**

Table-2 Program comprehension factors affecting cost.

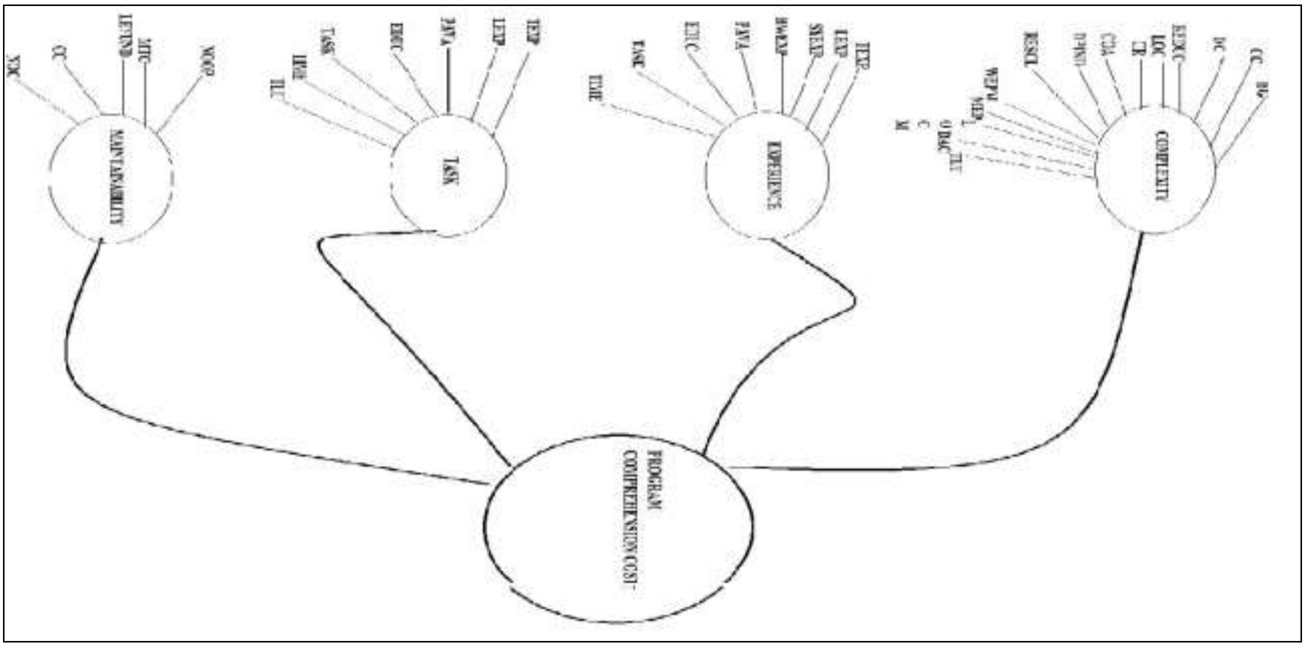

Figure 1- Bayesian network model for program comprehension cost factors (Montani et al., 2010)

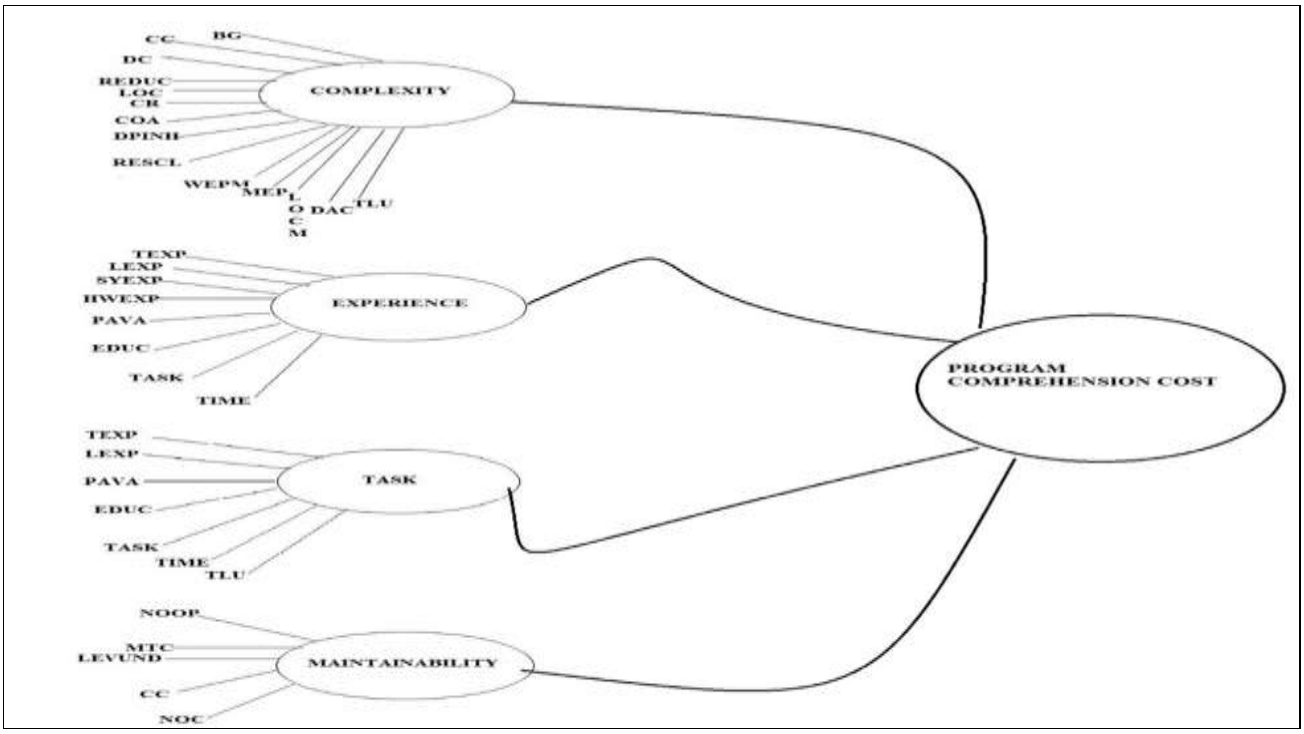

Figure 2 - Hierarchical Correlation of Cost Factors and program comprehension

Rule Base

The basic rules for the cost comprehension is depends on the various parameters. The different parameters which are present in the software accordingly the cost will be varies. If (condition) of a rule relates characteristics by means of the logical operators (and). These characteristics are the branches that arrive to a node, that is, the evidences and the result of previous nodes. The system works with probability factors (PF) which are equivalent to values associated to nodes and evidences represented on the graph. When a condition is satisfied, a resulting CPF is obtained from the PF of each entry parameter. The value attained to a node represent the maximum PCF that may be achieve by this node, that is, when all parameters are satisfied in a particular disease. The final result is obtained by the execution of all the pertinent rules. The rules build from hierarchical model are given below:

Parameter	Weight (W)	Prior probabilities (P)	W*P	Parameter	Weight (W)	Prior probabilities (P)	W*P
COMPLEXITY				**EXPERIENCE**			
BG	0.07	0.5	0.035	TEXP	0.12	0.5	0.06
CC	0.07	0.5	0.035	LEXP	0.12	0.5	0.06
DC	0.07	0.5	0.035	SYSEXP	0.12	0.5	0.06
REDUC	0.07	0.5	0.035	HWEXP	0.12	0.5	0.06
LOC	0.07	0.5	0.035	PAVA	0.12	0.5	0.06
CR	0.07	0.5	0.035	EDUC	0.12	0.5	0.06
COA	0.07	0.5	0.035	TASK	0.12	0.5	0.06
DPINH	0.07	0.5	0.035	TIME	0.12	0.5	0.06
RESCL	0.07	0.5	0.035	**MAINTAINABILITY**			

WEPM	0.07	0.5	0.035	NOOP	0.2	0.5	0.1
LOCM	0.07	0.5	0.035	MTC	0.2	0.5	0.1
DAC	0.07	0.5	0.035	LEVUND	0.2	0.5	0.1
TLU	0.07	0.5	0.035	CC	0.2	0.5	0.1
MEP	0.07	0.5	0.035	NOC	0.2	0.5	0.1
TASK							
TEXP	0.14	0.5	0.07	EDUC	0.14	0.5	0.07
LEXP	0.14	0.5	0.07	TASK	0.14	0.5	0.07
PAVA	0.14	0.5	0.07	TIME	0.14	0.5	0.07
				TLU	0.14	0.5	0.07

Table1 Probabilities of cost matrix

Rules for cost estimation

R11: IF BG(p1) & CC(p2) & DC(p3) & REDUC(p4) & LOC(p5) & CR(p6) & COA(p7) & DPINH(p8) & RESCL(p9) & WEPM(p10) & MEP(p11) & LOCM(p12) & DAC(p13) & TLU(p14) THEN **COMPLEXITY**

R12: IF TEXP(p1) & LEXP(p2) & SYSEXP(p3) & HWEXP(p4) & PAVA(p5) & EDUC(p6) & TASK(p7) & TIME(p8) THEN **EXPERIENCE**

R13: IF TEXP(p1) & LEXP(p2) & PAVA(p3) & EDUC(p4) & TASK(p5) & TIME(p6) & TLU(p7) THEN **TASK**

R14: IF NOOP(p1) & MTC(p2) & LEVUND(p3) & CC(p4) & NOC(p5) THEN **MAINTAINABILITY**

R1: IF **COMPLEXITY(p1) &** IF **EXPERIENCE(p2) &** IF **TASK(p3) &** IF **MAINTAINABILITY THEN PROGRAM COMPREHENSION COST (P)**

The value given in brackets in left hand side is Probability factor of the corresponding parameters whereas the value given in the bracket in right hand side is probability factor of the rule.

Probability Factor

The probability-*factor* (PF) model is a method for managing uncertainty in rule-based systems which is developed by managing uncertainty in rule-based systems which is developed by Shortliffe and Buchanan (1975). The formula is as follows:

MB(h, e1 & e2) =MB(h, e1) + MB(h,e2) * (1-MB (h, e1)……………(1)

MD(h, e1 & e2) =MD(h, e1) + MD(h,e2) * (1-MB (h, e1)……………(2)

MD(h, e1 & e2) = 0 , IF MB(h, e1 & e2) = 1 i.e. all the evidences (e1 &e2) approves the hypothesis (h).
CCF= MB(h, e1 & e2) + MD(h, e1 & e2) ……………..(3)

Where MB is Measure of Belief and MD is measure of disbelief and CPF id is Cumulative Probability-*Factor*.

Bayesian Network

Bayesian network graph is implemented by Homayun et al., (2010) on software quality model. Bayesian network graph for software quality has shown in Figure 2. It has three levels. First level has cost comprehension and their arcs go to metrics in second level. Cost factor's arcs go to third level in code comprehension node. Each node (cost, cost factors, metric) has five states: Very High (degree =5), High (degree =4), Medium (degree =3), Low (degree =2) and Very Low (degree =1). The value in bracket is degree associated with the level. The data (probability) for first level (quality metric) is varies from company to company or organization to organization. For example security in a company is more important than another one. Therefore the quality experts of each company can fill them in conditional probability table (CPT) according to their quality metrics, knowledge, and experience of previous software in their Company. Using the previous knowledge of the similar project and software is one of the most important factors to filling CPT. The formula used for computing the probability of each quality factor is given below:

$$P=\frac{\sum_{i=1}^{n} p_i * d_i}{L * \sum_{i=1}^{n} p_i} \text{ -------------------(4)}$$

Where p_i and d_i are probability (as shown in Table 1) and degree of ith quality matrix. L is number of level (i.e. 14).

For example 'TASK' and its seven metrics are: TEXP (M with degree= 7), LEXP (M with degree= 7), PAVA (M with degree= 7), EDUC (M with degree= 7), TASK (M with degree= 7), TIME (M with degree= 7) & TLU (M with degree= 7).

d = 7 + 7 + 7 + 7 + 7 + 7 + 7 = 49

p = (0.07* 7 + 0.07* 7 + 0.07* 7 +0.07* 7 + 0.07* 7 + 0.07* 7 + 0.07* 7) / 14 * (0.07+0.07+ 0.07+ 0.07+0.07+ 0.07+ 0.07) = 0.44 using formula 4.

Calculating in percentage we get 44%. It means we have Task in this example by probability of 44 percent. Similarly, for other quality factor CPT.

RESULT AND DISSCUSION

In the hierarchal tree shown in figure 1, we assume that the cost factor equally depends upon Complexity, Task, Maintainability and Experience. Therefore, equal weights ¼ is assigned to each parameter. For different organization the weight is different for different parameters. Further these quality factors depends equally of upon quality matrix. For example task depends upon seven quality matrix as shown in figure 1. We assume that functionality equally depends upon the fourteen quality matrix (the dependencies varies from company to company) therefore equal weights i.e. 1/14 is assigned to each quality matrix.

In the hierarchal tree (Figure 3) the evidence in the support of level one i.e. Task are e1: TEXP (p1=0.5), e2: LEXP (p2=0.5), e3: PAVA (p3=0.5), e4: EDUC (p4=0.5), e5: TASK (z5=0.5), e6: TIME (z6=0.5) & e7: TLU (z7=0.5). Where p1, p2, p3, p4, p5, p6 and p7 are PF. Therefore the CPF for rule 1 is calculated as follows.

For e2

MB=0.5 + 0.5 (1-0.5) =0.75, MD=0.0

CPF= 0.75

For e3

MB=0.75 + 0.5 (1-0.75) =0.825, MD=0.0

CCF= 0.825

For e4

MB=0.825 + 0.5 (1-0.825) =0.9125, MD=0.0

CCF= 0.9125

For e5

MB=0.9125 + 0.5 (1-0.9125) =0.956, MD=0.0

For e6

MB=0.956 + 0.5 (1-0.956) =0.978, MD=0.0

For e7

MB=0.978 + 0.5 (1-0.978) =0.989, MD=0.0

5. Conclusion

The proposed model used the various factors as input and accordingly the probability of each factor has been calculated.

- Some of quality factor (in different software) can't be computed
- Sometimes cost of compute all of quality factors are very high
- We need long time for compute all of quality factors
- In some cases we need to a lot of people (user, quality manager, member of quality assurance's team) to compute quality factors

6. References:

- Scott R. Tilley Dennis B. Smith “Coming Attractions in Program Understanding” Technical **Report** CMU/SEI-96-TR-019, ESC-TR-96-019 December 1996.
- Michael P. O’Brien “Software Comprehension – A Review & Research Direction" ,2003.
- Cain, J.W., & McCrindle, R.C., “Software Visualisation using C++ Lenses”, 1998
- A.v. Mayrhauser and M. Vans, “Program Comprehension during Software Maintenance and Evolution,” *IEEE Computer*, vol. 12, pp. 44-55, 1995.
- M. Koch, “Introduction to CSCW,” Technical University of Munich, Germany http://www11.informatik.tu-muenchen.de/cscw/, 1996.
- Landauer, T. K., Foltz, P. W., & Laham, D. (1998). “Introduction to Latent Semantic Analysis” . *Discourse Processes,* **25**, 259-284
- Dunsmore A. “Comprehension and Visualisation of Object-Oriented Code for Inspections “Empirical Foundations of Computer Science (EFoCS) EFoCS-33-98, 1998.
- Sneed,H. M., & Bavaria,A., "Source Animation as a means of Program Comprehension for object-oriented System" IEEE, 2000.
- Deimel, L., Naveda, J., (1990), “Reading Computer Programs: Instructor’s Guide and Exercises”, *Technical Report CMU/SEI-90-EM-3,* Software Engineering Institute,Carnegie Mellon University.
- Burd, L., Munro, M., & Young, P., (2000), “Using Virtual Reality to Achieve Program Comprehension”, *http://www.year2000.co.uk/munro.html.*
- Margaret-Anne Storey, “Theories, tools and research methods in program comprehension: *past, present and future”* Springer Science + Business Media, LLC 2006. Software Qual J (2006) 14:187–208 DOI 10.1007/s11219-006-9216-4.

- Wang Kechao, "Overview of Program Comprehension" 2012 International conference on Computer Science and Electronics Engineering. 978-0-7695-4647-6/12 $26.00 © 2012 IEEE. DOI 10.1109/ICCSEE.2012.285.

- Letovsky, S., (1986), "Cognitive Processes in Program Comprehension", *Empirical Studies of Programmers: 1st Workshop*, p 58.

- Michael P. O'Brien "Expectation-based, inference-based, and bottom-up software comprehension" JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE J. Softw. Maint. Evol.: Res. Pract. 2004; 16:427–447 Published online in Wiley Inter Science (www.interscience.wiley.com). DOI: 10.1002/smr.307

- Collard,M.L., Kagdi, H.H., & Maletic., J.I., "An XML-Based Lightweight C++ Fact Extractor" IEEE,2002

- Biggerstaff T.J., Mitbander B.W. and Webster D. 1993. The concept assignment problem in program understanding. In *Proceedings of the 15th international conference on Software Engineering*, pp. 482–498.

- Soloway, E., Ehrlich, K., (1984), "Empirical Studies of Programming Knowledge", *IEEE Transactions on Software Engineering*, IEEE Computer Society, Vol. SE-10, No. 5, September

- Chen Peter & Xiaolei, Du "Improving software comprehension process by adoption of cognitive theories in large scale complex software maintenance", University of Gothenburg Chalmers University of Technology, Department of computer science and engineering, Goteborg, Sweden, June 2012.

- Simon, H "Cognitive Models Bottom-up", Seminar on program analysis and program comprehension, June 23, 2014.

- Wei, S., " A survey and categorization of program comprehension Techniques", IEEE,2002

- D. F. Jerding, *ISVis*, http://www.cc.gatech.edu/morale/tools/isvis/isvis.html

- P. K. Linos, *A Preliminary Report on Program Comprehension Tools (PCT's)*, Tennessee Technological University, http://www.csc.tntech.edu/~linos/pcts.html

- MicroGold Software Inc., *With Class 98*, http://www.microgold.com

- *Program Comprehension Tools Bibliography*: http://www.csc.tntech.edu/~linos/pctinfo.html

- TakeFive Software, *SNiFF+*, http://www.takefive.com

- Yin.,M., Li.,B., & Tao., C., "Using Cognitive Easiness Metric for Program Comprehension" IEEE,2008

- P. Young, "*Program Comprehension"*, Visualisation Research Group, Center for Software Maintenance, University of Durham, 1996.

- P. Young, *Software Visualisation*, Visualisation Research Group, Centre for Software Maintenance, University of Durham, 1996

- *LOOK!,* Objective Software Ltd, 1 Michaelson Square, Kirkton Campus, Livingston, EH54 7DP, UK.

- A.Dunsmore, "Survey of Object-Oriented Defect Detection Approaches and Experiences in Industry", Technical Report – EFoCS-36-2000, Computer Science Department, Strathclyde University, August 2000.

- Alastair Dunsmore, Marc Roper, and Murray Wood "Practical Code Inspection for Object-Oriented Systems" 2000, IEEE

- Von Mayrhauser, A. and Vans, A. M., "Program understanding behavior during debugging of large scale software", in Proceedings of Seventh workshop on Empirical studies of programmers, 1997, pp. 157–179.

- Shneiderman, B. and Mayer, R., "Syntactic / semantic interactions in programmer behaviour: a model and experimental results", International Journal of Computer and Information Sciences, vol. 8, no. 3, 1979, pp. 219 - 238.

- Pennington, N., (1987), "Comprehension Strategies in Programming", *Empirical Studies of Programmers: 2nd Workshop*, p 100.

- Von Mayrhauser, A. and Vans, A. M., "From program comprehension to tool requirements for an industrial environment", in Proceedings of Second Workshop on Program Comprehension, Capri, Italy, 1993, pp. 78-86.

- Sim, S. E. and Storey, M.-A. A Structured Demonstration of Program Comprehension Tools. In Proceedings of Seventh Working Conference on Reverse Engineering (Toronto, Ontario, Canada, 1999), IEEE Computer Society, 184.

- Mayrhauser, A. v. and Lang, S. On the Role of Static Analysis during Software Maintenance. In Proceedings of 7th International Conference on Program Comprehension (Pittsburgh, PA, 1999), IEEE Computer Society, 170-177.

- Knight, C.(1998) Visualization for Program Comprehension: Information and Issues.

- K. Chen, V. Rajlich, "Case Study of Feature Location Using Dependence Graph," In Proceedings of the 9th IEEE International Workshop on Program Comprehension, Los Alamitos, 2000, pp 241-249.
- A. Kohler. Der C/C++-Projektbegleiter. dpunkt.verlag, 69115 Heidelberg, 2007.

- Schauer, R and Keller, R. K. Integrative Levels of Program Comprehension, 2008 15th Working Conference on Reverse Engineering.

- Deng, F. and Jones, A. J, *Weighted System Dependence Graph*, 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation.

- Tilley, S. R., Paul, S., & Smith, D. B., 1996 “Towards a Framework for Program Understanding”, 4th International Workshop on Program Comprehension.

- Tjortjis, C., Sinos, L., & Layzell, P. (2003). Facilitating Program Comprehension by Mining association Rules from Source Code, Proceedings of the 11th IEEE International Workshop on Program Comprehension (IWPC'03), pp. 125-132.

- Tsarkov, D., & Horrocks, I. (2006). FaCT++ Description Logic Reasoner: System Description, International Joint Conference on Automated Reasoning (IJCAR 2006).

- Von Mayrhauser, A., & Vans, A. M. (1994). Comprehension Processes During Large Scale Maintenance, 16th International Conference on Software Engineering.

- Walenstein, A. (2002). Cognitive Support in Software Engineering Tools: A Distributed Cognition Framework. PhD thesis. Simon Fraser University.

- Walkinshaw, N., Roper, M., & Wood, M. (2005). Understanding Object-oriented Source Code from the Behavioral Perspective, Proceedings of the 13th IEEE International Workshop on Program Comprehension (IWPC'05), pp. 215-224.

- Rajlich, V., & Wilde, N.(2002). The Role of Concepts in Program Comprehension, Proceedings of the 10th IEEE International Workshop on Program Comprehension (IWPC'02), pp. 271-278.

- Haarslev, V., Moller, R., & Wessel, M. (2004). Querying the Semantic Web with Racer+ nRQL, KI-2004 International Workshop on Applications of Description Logics (ADL'04).

- Devanbu, P. T., Brachman, R. J., Selfridge, P. G., & Ballard, B. W. (1990). LaSSIE - a Knowledge-based Software Information System, Proceedings of the 12th international conference on Software engineering. Nice, France.

- Brooks, R. (1983). Towards a Theory of the Comprehension of Computer Programs. International Journal of Man-Machine Studies, 18(6), pp. 542-554.

- Weinand A., Gamma E., Marty R.: Design and Implementation of ET++, a Seamless Object-Oriented application Framework. Structured Programming Vol. 10, No. 2, 1989.

- Sametinger. J, “Improving program comprehension of object oriented software with object-oriented documentation” 1993.

- Vinz.,B.L., & Etzkorn., L.H., "Improving program comprehension by combining code understanding with comment understanding" IEEE,2008.

- Kadar., R., & Sulaiman., S., "The Effectiveness of Zoom Visual Flow (ZViF) Technique in Program Comprehension Activities" IEEE, 2010.

- Kothari., S.C., "Scalable Program Comprehension for Analyzing Complex Defects”, IEEE,2008.

- Cséri.,T., ugyi., Z.S., & Porkoláb.,Z. "Rule-Based Assignment of Comments to AST Nodes in C++ Programs" , IEEE,2012. ACM 978-1-4503-1240-0/12/09

- Paydar. S, et. al. "A Semantic Web Based Approach for Design Pattern Detection from Source Code" 2012, 2nd International eConference on Computer and Knowledge Engineering (ICCKE), October 18-19, 978-1-4673-4476-0/12/$31.00 ©2012 IEEE. Page 289-294.

- Aris.,T.N.M., & Nazeer., S.N., "Object-Oriented Programming Semantics Education based on Intelligent Agents" , IEEE, 2011, pp. 404-407.

- Ala Abuthawabeh.,A., & Zeckzer., D., "IMMV: An Interactive Multi-Matrix Visualization for Program Comprehension" IEEE, 2013.

- Blinman, S. et, al. "Program Comprehension: Investigating effects of naming styles and documentation" Copyright © 2005, Australian Computer Society, Inc. This paper appeared at the 6th Australasian User Interface Conference (AUIC2005), Newcastle. Conferences in Research and Practice in Information Technology, Vol. 40. M. Billinghurst and A. Cockburn, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

- Dunsmore, A. "Comprehension and Visualisation of Object-Oriented Code for Inspections" Empirical Foundations of Computer Science (EFoCS) Department of Computer Science University of Strathclyde Livingstone Tower Glasgow G1 1XH, U.K. August 1998.

- Singh, R., et al "Dependence cache slicing - A case study" published in *Computing Trendz Journal* of SMS Varanasi. Vol. III, No. 2, July 2013.ISSN 2230-9152. http://www.smsvaranasi.com/Computing_Trendzt_journal.htm.

- Sneed, M. H et, al. "Source Animation as a means of Program Comprehension for object-oriented Systems" IEEE, 2000. 0-7695-0656-9/00.

- Tjortjis, C. et, al "From System Comprehension to Program Comprehension" Proceedings of the 26 th Annual International Computer Software and Applications Conference (COMPSAC'02) 0730-3157/02 $17.00 © 2002 IEEE.

- Rajlich, V. et, al "The Role of Concepts in Program Comprehension" Copyright 2002 IEEE. Published in the Proceedings of IWPC 2002, IEEE Computer Society Press, Los Alamitos, CA, ISBN 0-7695-1495-2, pp.271-278.

- Limpiyakorn, Y. et al. "Applying the signature concepts to plan-based program understanding" Proceedings of the International Conference on Software Maintenance (ICSM'03), 1063-6773/03 $17.00 © 2003 IEEE.

- Mahmoud, A et, al. "Evaluating Software Clustering Algorithms in the Context of Program Comprehension" ICPC 2013, San Francisco, CA, USA 978-1-4673-3092-3/13/$31.00 c 2013 IEEE pp. 162-171.

- Kanellopoulos, Y. et, al "Data Mining Source Code to Facilitate Program Comprehension: Experiments on Clustering Data Retrieved from C++ Programs" 2003.

- Zhang, Y. et, al "An Ontology-based Approach to Software Comprehension- Reasoning about Security Concerns" Computer Software and Applications Conference, 2006. COMPSAC '06. 30th Annual International (Volume:1) DOI 10.1109/COMPSAC.2006.27. IEEE.

- J. S Sottet, F Jouault "Program comprehension" Proc. 5th Int. Workshop on Graph-Based Tools, 2009.

- Harman, M. "Search Based Software Engineering for Program Comprehension" 15th IEEE International Conference on Program Comprehension (ICPC'07) 0-7695-2860-0/07 $20.00 © 2007.

- Dietrich, J. et, al. "Using social networking and semantic web technology in software engineering – Use cases, patterns, and a case study" The Journal of Systems and Software 81 (2008) 2183–2193.

- Bradley L. Vinz et, al "Improving program comprehension by combining code understanding with comment understanding" Knowledge-Based Systems 21 (2008) 813–825.

- Y. Manman, "Using Cognitive Easiness Metric for Program Comprehension" Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on 23-25 June 2010 Page(s): 134 – 139, Chengdu, China IEEE.

- S.C. Kothari "Scalable Program Comprehension for Analyzing Complex Defects" The 16th IEEE International Conference on Program Comprehension. 978-0-7695-3176-2/08 $25.00 © 2008 IEEE DOI 10.1109/ICPC.2008.

- Feigenspan, J. et, al. "How to Compare Program Comprehension in FOSD Empirically – An Experience Report" FOSD '09 Proceedings of the First International Workshop on Feature-Oriented Software Development Pages 55-62. ACM New York, NY, USA ©2009.

- Feigenspan, J. et, al. "Program Comprehension of Feature-Oriented Software Development"

- Deng. F. et, al. "Weighted System Dependence Graph" 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation.

- Soh, Z. "Context and Vision: Studying Two Factors Impacting Program Comprehension" 2011 19th IEEE International Conference on Program Comprehension. 1063-6897/11 $26.00 © 2011 IEEE DOI 10.1109/ICPC.2011.37. Pages 258-261.

- Freitas, J. L el. al. "A Comment Analysis approach for Program Comprehension" 2012 IEEE 35th Software Engineering Workshop. 1550-6215/13 $26.00 © 2013 IEEE DOI 10.1109/SEW.2012.8.

- Haiduc, S. et. al "Supporting Program Comprehension with Source Code Summarization" 2012

- Binkley, D et. al "The impact of identifier style on effort and comprehension" Empir Software Eng (2013) 18:219–276. DOI 10.1007/s10664-012-9201-4.

- Singh, R et. al "Dependence cache slicing –A case study" SMS Varanasi.

- Banker. D , Rajiv "Software complexity and maintainability" 1992.

- Banker. D , Rajiv "SOFTWARE COMPLEXITY AND SOFTWARE MAINTENANCE COSTS" 1992

- "How to save on software maintenance costs" Omnext while paper, March 2010.

- Kasto. N et al. “Measuring the difficulty of code comprehension tasks using software metrics” Proceedings of the fifteenth Australasian computer education conference (ACE2013), Adelaide, Australia. Conference in research and practice in information technology (CRPIT, Vol. 136.

- Feigenspan, J. et al. “ Exploring software measures to access program comprehension”

- Hayes, J. H et al “A Metrics-Based Software Maintenance Effort Model”

- “Software Effort Estimation An Exploratory Study of Expert”

- J. C. GRANJA-ALVAREZ et al. “A Method for Estimating Maintenance Cost in a Software Project: A Case Study” SOFTWARE MAINTENANCE: RESEARCH AND PRACTICE, VOL. 9, 161–175. 1997 by John Wiley & Sons, Ltd.

- Welker, K.D. and Oman, P.W. Software Maintainability Metrics Models in Practice, *Journal of Defense Software Engineering*, Volume 8, Number 11, November/December 1995, 19-23.

- Agent Based Code Comprehension Model Using Semantic Knowledge Base In IJERT, Volume. 3, Issue. 05, May - 2014. (ISSN: 2278-0181) www.ijert.org