Estimation of Software Complexity within an Optimized Time Period Using a Multi Agent System

Mina Attarha,Hamid Reza Naji,Mohammad Malakooti
Atta.mina@yahoo.com;hamidnaji@ieee.org;malakooti@iau.ae
007-956
Mina Attarha

Abstract

This paper presents estimation of software complexity at an optimized time in software development process. Software complexity is the main factor that greatly affects on software delivery time, expenditure within the budget allotted and securing the satisfaction of the users. These are the elements that guarantee the success of the software. Many attempts have been made for measurements of complexity at the phase of code design but recognition of complexity and controlling it at these phases are rather late for measurement of complexity. By measuring the factors that cause complexity at the early stages of software development life cycle we can save and prevent many problems such as errors and defects that may appear in the later phases of software production. In this paper we are proposing that the optimized time for measuring complexity is at the phase of requirements and increase the speed of measurement using a multi agent system. The objective of this work is to reduce the cost of software production, resources and maintenance.

Key Words: Multi Agent System, Software Complexity, Requirement Engineering

.

1. Introduction:

The increase costs of software development and maintenance is mainly due to software complexity, and in the recent decades many software organizations have been searching for ways and methods to reduce these costs. These efforts have been mainly focused on measuring complexity at the phases of code and design (for example LOC[8], Halstead[9,10], McCabe[10,11], Henry and Kafura [12], KLCID [13]) and in measurement of complexity on these basis efforts have been made to measure software complexity at the later phases of development while no effort has been spent on measuring complexity at the earlier phases of development. The origination of software complexity is the stage of requirement [1]. The reason for focus on this paper on the process of requirement for being the best starting point for complexity metrics in software from requirement is best express by Fredrick Brooks which illustrate why requirement are so important" the hardest part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all of the interfaces to people, to machines, to other software systems. No other part of the work so cripples th resulting system if done wrong. No other part is more difficult to rectify later''[2]. "if you don't get the requirements right, it doesn't matter how well you do anything else" [3]. In this paper we use Multi Agent System, by utilizing the metrics proposed by Sharma [4] and Keshavarz [5] for measure of software complexity based on requirements in the first phase of software lifecycle. For computation of complexity metric based on requirements, by application of Multi Agent Systems, complexity formula in different parts will be modeled and at the end by relying on these models a final model for early estimation of complexity based on requirements will be presented.

In this paper, we adapt and use a reduced form of the belief, desire, and intention (BDI) architecture for agents that other have successfully applied in software for several years [6,7]. Beliefs represent the set of working assumptions that the agent has about itself and the environment in which is functions. This forms the agent's informational state- a state in which the agent can locally modify incomplete or inaccurate information through its interactions with other agent and the environment. Desire represents the high-level set of objectives and goals that the agent is trying to achieve. The agent's desires must be realistic not conflict with each other. Intentions represent the agent's deliberative state in which the detailed sequences of actions, or plans, made to environment and other cooperating agents through actuators are maintained.

2. Estimation of software complexity based on requirement using a multi agent system

The novelty of using multi agent system in estimation complexity is the advantage that it enables the software engineer to measure the complexity before any other phase in the process of development of the software. The benefit of this method is that the degree of software complexity can be estimated before the phases of design and implementation so that the development process would be able to produce good quality software at faster time and lower cost. To achieve this result even in a faster time and produce better quality software in this paper we are proposing to use multi agent system. By using multi agent system and due to the fact that these agents work in parallel which causes a lot of work to be accomplish in parallel system and also since the agents are autonomous and intelligent they are capable to compute and estimate complexity at each stage without human interference. The paradigm of the proposed system is discussed in the following sections:

3. Input/Output (I/O) Complexity

This complexity has reference to input and output of software system and their interface and is computed by equation (1). To compute input output complexity we need one Multi-Agent which has four input and one output and this agent(1) will function as sigma figure(1) .

- Input: Information which through interface enter the system .(No. of Inputs)
- Output: Information which through interface exit the system (No. of Outputs)
- Interface: User Interface where the Input are to be issued and output to be seen and specifically number of integration required.

So, Input Output Complexity can be calculated as equation (1):

$$IOC = \sum_{i=1}^{4} No.of\ Inputs_i + No.of\ Outputs_i + No.of\ Interfaces_i$$

Index i has reference to the type of interface(Hardware, software, functional , connectivity)
No.of Interface is indicative of number of Interface, and *No.of input i* in indicative of number of entries, and *N.of output i* indicates number of exits type of interface *i* .

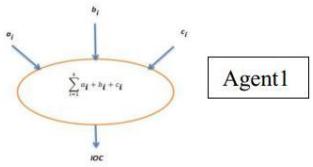


Figure 1. Agent1

4. Functional Requirement Complexity (FR)

Functional requirement is the basic activities of the software and has the principal role in acceptance of the software in other hand functional requirements should define the fundamental actions that must be take place. Generally these are commands that begin with the phrase " system should ….." and should be externally perceivable by users. To have better understanding of functions, classifying functional requirements into Sub-function and

Sub-process seems suitable. Also we must remember that each sub function may include some subs of its own. This issue has been taken into consideration. For example , a system has three functions and each function has two or three sub functions which in turn they have their own sub functions (Please see figure 2) .Also volatility can be classified as in(Table 1), and finally functional requirement complexity can be calculated as equation(2).

$$FR = \left(\sum_{i=1}^{3} Coefficien\, t_i \times No_i \right) \times \left(\sum_{j=1}^{level-1} \sum_{k=1}^{n} SF_k \right) \quad (2)$$

n= (represents) numbers functions on level j

SF_k = represents number of sub-functions of Kth function requirements

No_i = represents number of function requiremets from i type of volatility

$Coefficient_i$ = represent coefficient of type I volatility

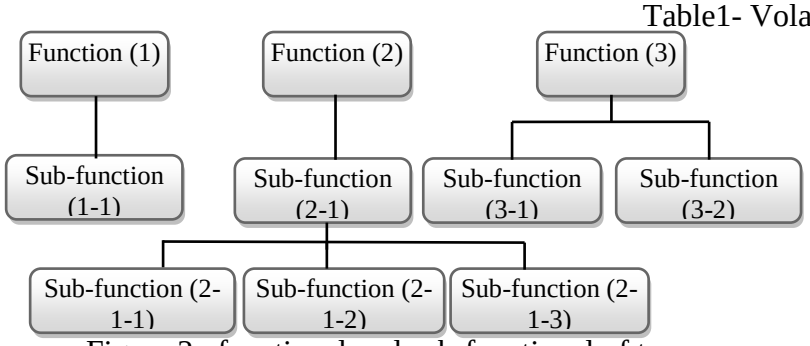


Figure 2. functional and sub-functional of tree

Table1- Volatility coefficient of requirements

Type of Volatility	Volatility Coefficient
stable	1
Rarely	2
often	3

To calculate functional requirements we need 3 Agents. Agents 2 and 3 in parallel, will compute functional requirement formula and both will send the answers to Agent 4 figure (3).

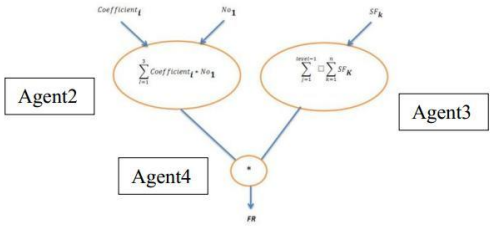


Figure 3. Agents2-4

5. Non-Functional Requirement Complexity (NFR)

Non- functional requirement refers to quality requirements related to the software and not fulfilling those leads to customer's dissatisfaction. These will be dealt with as detached from functional capability of the software. More number of non-functional requirements and the degree of importance and compulsion to do them leads to more complexity in the product. A way to requirements is to distinguish classes of requirements as essential, desirable and optional. Therefore non-functional requirement complexity will be computed on the basis of equation (3).

$$NFR = \left(\sum_{i=1}^{3} Coefficient_i \times No_i \right) (3).$$

No_i is the number of non-functional requirement with the importance degree of type i and $Coefficient_i$ is the importance coefficient of type of i (Table 2). From the point of view of importance of activities in the system, we have also considered the coefficient of non-functional requirements Table 2. More force to do non-functional requirements may impose higher degree of complexity to the software.

Table 2- Non-Functional requirement and its coefficient

Importance Degree	Importance Coefficient
Optional	1
Desirable	2
Essential	3

One agent is needed to calculate non-functional requirement, that will take two inputs and the output will be non-functional requirement complexity figure (4).

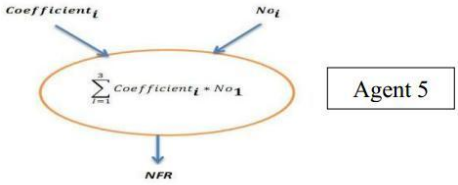


Figure 4. Agent5

6. Requirement Complexity (RE)

This attribute refers to the totality of complexities of requirements, functional , sub functional and non-functional requirements. This complexity can be studied as equation (4).

$$RC = FR + NFR \quad (4)$$

To calculate complexity in these requirements we needed one Agent. The speed of the process is clear, and in order to compute the requirements, in parallel we calculated the complexities of functional and non-functional. For example we did not wait to complete computing the complexity of non-functional before the functional was finished figure (5).

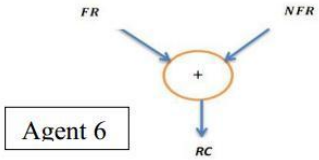


Figure 5. Agent6

7. Design Constraint

It refers to No. of constraints are to be consider during development of product by any statuary body/ agencies which includes No. of Regulatory constraints, hardware constraint, communication constraints, database constraints etc. this metrics can be mathematically defined as 6:

$$DC = \sum_{i=1}^{n} C_i \quad (6)$$

As figure 6 shows we need one agent in our model for calculation of forced design constraints.

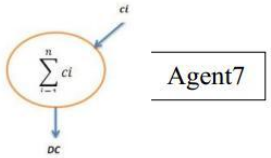


Figure 6. Agent7

Where C_i is number of constraints and value of C_i will vary from 0 to n.

$$C_i = \begin{cases} 0 & \text{If Blind Development} \\ \text{Value} & \text{If Constraints exists} \end{cases}$$

8. Product Representation Complexity

This metric refers to the type of accessibility of software for the users, number of synchronize users, and number of user classes (6).

$$PRC = No.UserClass \times AccessType \times No.SynUsers$$

In this term No. ofUserClass represent number of user class and Accesstype represent coefficient of type of accessibility and No.Syn User represent number of synchronize user.

Table 3. Coefficient of type of user accessibility

Complexity coefficient	Type of accessibility
1	Local accessibility
2	Remote accessibility

Table 4. Coefficient number of synchronize users

Complexity Coefficient	Number of user
1	1-50
2	1-5000
3	+ 5000

F ... put and o...

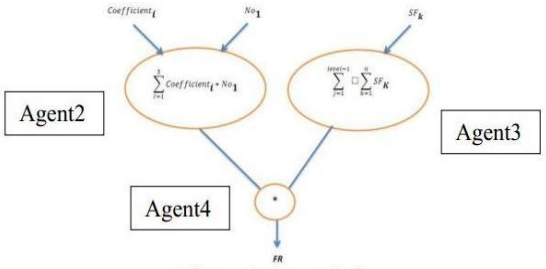


Figure 3. Agents2-4

9. Human Resource Complexity

Human resources play a very significant part in development and improvement of the product.

For this reason the human resource complexity will be reviewed on two levels:

9.1 Team-Project Complexity

As it was stated in the complexity model, team project comprises of expertise and experiences of the members. Technical specialty (TS) plays an important role in effective development of the project. The expertise of the team members in some complicated methods can be measured. In this metric for simplicity in calculation of ability and capability of the personnel , their characteristics have been used in COCOMO expense model. Table 5-5 and complexity of team project will be computed on the basis of equation (7).

Table 5. Specification of team-project for cost COCOMO model

Specification	Very little	little	Normal	High	Very High
Analyzable	١,٤٦	١,١٩	١,٠٠	٠,٨٦	٠,٧١
programmable	١,٤٢	١,١٧	١,٠٠	٠,٩٥	
Work experience	١,٢٩	١,١٣	١,٠٠	٠,٩١	٠,٨٢
Language programmable	١,٤٢	١,١٧	١,٠٠	٠,٨٦	٠,٧١
Virtual machine experience	١,٢١	١,١٠	١,٠٠	٠,٩٠	

$$TS = \prod_{i=1}^{5} x_i \qquad (7)$$

We can consider the experience of the members of the project (Exp) as an effective factor in complexity of the requirements. Although the experiences of the team members by some complex methods can be measured, but the only relevant and important factors are whether the team member has worked on a similar project or not, and whether the team member has worked with this team before or not. The coefficient of experience (Exp) shall be extracted from table (7). These amounts will be calculated for every member of the team and the average will be used as coefficient for team project. In measurement of complexity for team project from equation (8) computing can be made.

Table 6. Coefficient experience of team-project

Complexity coefficient	Experience
١	Clerk who has had experience in this company and in similar projects
٢	Newly employed clerk in this company who has had experience in this line activity in other projects
	A clerk who has work in this company but has had no activity in similar projects
٣	Newly employed clerk who has had no activity in similar projects

$$TC = (Exp \times TS) \qquad (8)$$

We also need one agent for calculating team-project complexity with two inputs and 1 output figure (8).

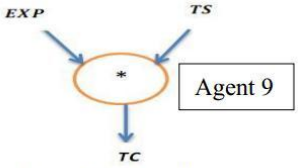


Figure 8. Agent9

9.2 Stakeholder Complexity

As it was presented in comple ; very important factors, will be reviewed in 6 levels which appe r complexity will be computed as equation 9.

Table 7 Stakeholders complexity

Coefficient Complexity		
Complexity=0	**Complexity=1**	**Factor**
same	different	**Language**
same	different	**Industrial Domain**
same	different	**Rule business**
same	different	**Stakeholder location**
limit	high	**Number of stakeholder**

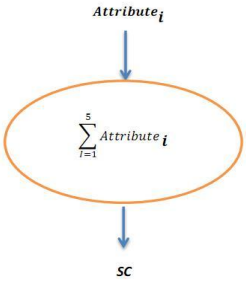


Figure 9. Agent10

Now for calculation of human resource complexity including both team-project complexity and stakeholder complexity which exhibited in equation 10 below:

So for calculating human resource complexity we need 3 agents as shown in figures 8, 9, 10 indicating agent 9 and 10 are work in parallel and reporting the result to agent 11.

$$HRC = (TC + SC)$$

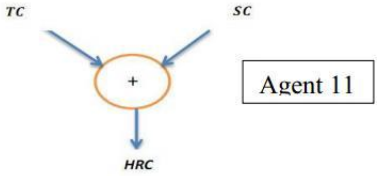


Figure 10. Agent11

10. Requirement Complexity Metric (RCM)

Finally considering the above formulas, system complexity in requirements phase by using the equation (11) can be calculated.

$$RCM = ((IOC \times RC \times HRC) + DC) \times PRC)$$

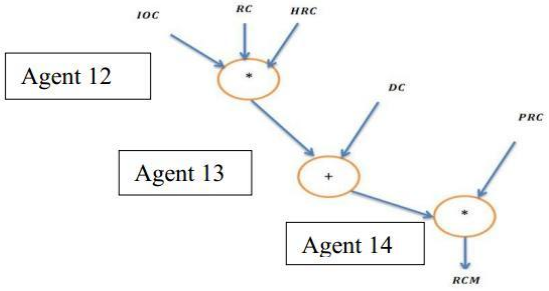
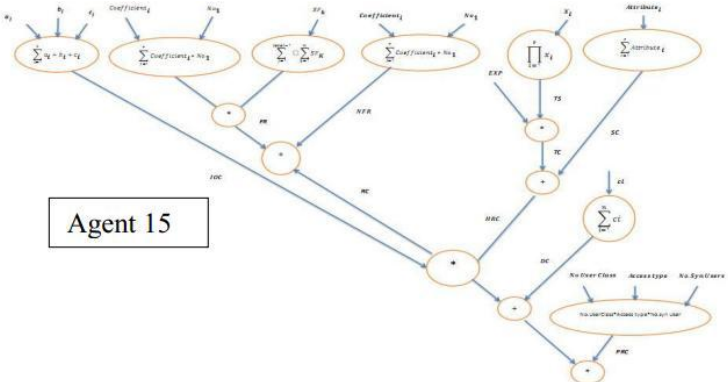


Figure 11. Agents12-14

There are 15 agents in a multi agent model which has been exhibited in figure (12).



11. Conclusion

So far most of efforts for software complexity calculations are done after code development phase. This causes wasting time, cost and resources. Also because of producing not optimized codes we need an extra reproduction phase. To better manage
software development, we have proposed a multi agent system for fast software complexity estimation based on requirement engineering. This system can provide a high speed optimized software development methodology which decreases unnecessary requirements and errors in different software development phases.

12. Reference

[1]Westfall, L., "Software Requirements Engineering: What, Why, Who, When, and How", Journal of Software Quality Professional, Vol. 7, No. 4, 2006, pp. 17-26

[2] Brooks, F. 1995. Mythical man-month: essays on software engineering, 20th anniversary edition. Addison-Wesley Professional. Gause, D., and G. Weinberg. 1989. Exploring requirements, quality before design. New York: Dorset House Publishing.

[3] Wiegers, K. E. 2004. In search of excellent requirements. Process Impact Web site. See URL: http://www.processimpact.com.

[4]*Ashish Sharma*, D.S. Kushwaha: "A Complexity measure based on Requirement Engineering Document", Journal of computer science and engineering, volume 1, issue 1, may 2010.

[5]Keshavarz, G., & Modiri, N., & Pedram. M. "Metric for Early Measurement of Software Complexity". International Journal on Computer Science and Engineering (IJCSE). Vol. 3, No.6, 2011.

[6]Naji, H. "Solving Complex Computational Problems Using Multiagents Implemented in Hardware". Journals of computing in Science & Engineering. Vol 10, 54-63, 2008.

[7]Rao, A., & Georgeff, M. "BDI Agents: From Theory to Practice". 1st International Conference, Multi Agent system (ICMAS95), MIT press, pp. 312-319,1995.

[8]Sheng, Yu., Shijie, Zh., "A survey on metric of software complexity", 2nd IEEE International Conference on Information Management and Engineering, 2010, pp.352-356.

[9]sylvia, B., and milliman, ph., "Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics", Ieee Transactions On Software Engineering,Vol.2, No. 2, 1980, PP: 96-104.

[10] Maurice Howard Halstead, Elements of Software Science, Elsevier, Amsterdam Netherlands, 1977.

[11]T.McCabe, " A Complexity Measure" , IEEE Transaction On Software engineering. IEEE, Washington, 1976, pp. 308-320.

[12] Henry, S., & Kafura, D. Software structure metrics based on information flow. IEEE Transactions on Software Engineering, 7(5),510–518, 1981.

[13]Kushwaha, D.S., & Misra, A.K. "Improved Cognitive Information Complexity Measure: A Metric that establishes program comprehension effort, ACM SIGSOFT Software Engineering, Vol. number. 5, 2006.