Agent Based Code Comprehension Model Using Semantic Knowledge Base

[1]Raj Singh
Research Scholar
Jodhpur National University
Jodhpur (Raj.)

[2]B. D Mazumdar
Assistant Professor
Department of Computer Applications
School of Management Studies
Varanasi, U.P

[3]A. K Vyas
Associate Professor
Department of Mathematics,
Faculty of Engineering and Technology
Jodhpur National University,(Raj.)

***Abstract*- We have proposed an agent based code comprehension model, which will help the novice, programmer and maintenance engineer to understand the code. The programmer use comments within the source proved better understanding than source code without comments. Based on this approach we have developed a model which will use semantic knowledge of the source and help to generate a new knowledge base from the existing knowledge inside the source code and comments. The agents have been used to automate all the process of code comprehension. The BDI (Belief, Desire and Intension) model of agent is implemented using java based jade tools. The different agents extract information at different level and help the novice, programmer and maintenance engineer to understand the large and complex source code. This model will be best suitable for code understanding and cost of understanding the code will be controlled up to large extents.**

***Keywords*- Agents, code comprehension, knowledge base, cognitive model, semantic code.**

I. INTRODUCTION

Software Comprehension is "the process of taking program code and understand it" [3]. defines it as "the activity of understanding existing software systems".[4].defines software comprehension as "the task of building mental models of the underlying software at various abstraction levels, ranging from models of the code itself, to models of the underlying application domain, for maintenance, evolution, and reengineering process. Program comprehension is an important part of software maintenance, especially when program structure is complex and documentation is unavailable or outdated. With the time the definition of software comprehension is enhances, this definition is "a process whereby a software practitioner and novice understand the software artefacts using semantic and syntactic knowledge to build mental model of its relation to the situation". Enormous work has been done on code comprehension, and there is need to

develop a model which can make easy to understand the code comprehension for programmers and novices. We have come across various visualization tools which are used to understand the code.

In software engineering, program comprehension is constantly taken into consideration, and it poses as a serious concern for the developers. When new programmers are assigned to an old code, they often complain about understanding it, and express their views about the code being unintelligible; therefore, software comprehension is very crucial and is especially needed in the occasions when old seasoned programmers leave their projects [5]

In this work we have proposed an agent based model for code comprehension, we have automate the whole process of code understating the code. In addition to this integrated visualization tool can be used to visualize for code analysis. We belief the code comprehension automated model could be more beneficial to novice, programmer and maintenance engineer, when new programmers are assigned to an old code, they often complain about understanding it. The basic model for code comprehension is shown in Figure-1.

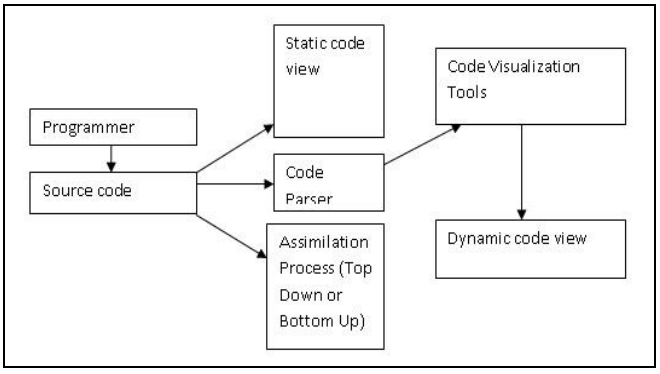

Fig-1 Code comprehension

II. PROBLEM DESCRIPTION

The problem in the most of the companies, sometime employees left, old or legacy system, improper documentation, different programming style used by the programmers. We have identified some of the key issues, we have identified the source code and comments based approach for code comprehension, sometime the programmer or the maintenance engineer forget to update the comments due to which it becomes very difficult for

the novice, programmers and maintenance engineer to understand the code. For this problem of understanding the code, we have used agent based semantic approach. We have used BDI model, our belief that this model could be best suitable for understanding the code. However the integrated visualization tool inside the interface will help to visualization of code for different analysis.

In order to understand code and comments is converted into semantic knowledge, on the basis of this concept the semantics for source code and semantics for comments separately collected into different knowledge bases. On the basis of keyword similarities, we have done searching and comparison from the knowledge bases using agents. Due to agent properties, the social influence and responsive behavior, we think that agent could be best choice for communication between the different levels of code comprehension activity.

III. COGNITIVE MODELS

The top down model is followed when the programmer or the developer is having the knowledge of programming or domain experience. The problem is divided into sub-hypotheses. In this approach we are dividing the problem into sub-goal [6]. In this work we have taken the source code and decided to achieve the different hypotheses. The source code is then converted into different semantic knowledge base. To achieve the goals, we have designed the source code and comments into semantic knowledge bases. On the basis of keyword similarities inside the code and comments, we have compared the comments to code matching ratio i.e. hit and miss ratio. When a particular keyword matches inside the source code and comments from different semantic knowledge bases, then we have used the agents' comparator. This will send a final report for code to comments hit ratio. If there is some code and comments are not matching then we have used a miss ratio report with line of code where it is not matching. The top down model for code comprehension is as shown in Fig-3.

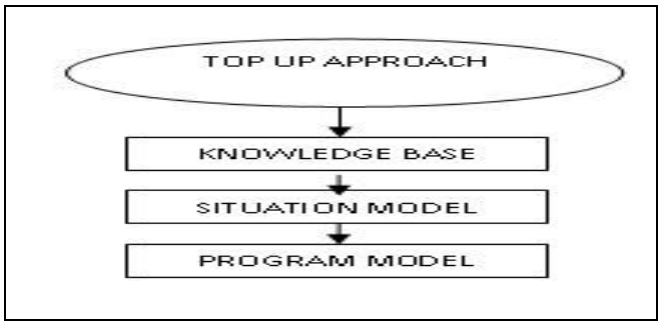

Figure-3 Top down Model for code comprehension

IV. ARCHITECTURE OF PROPOSED MODEL

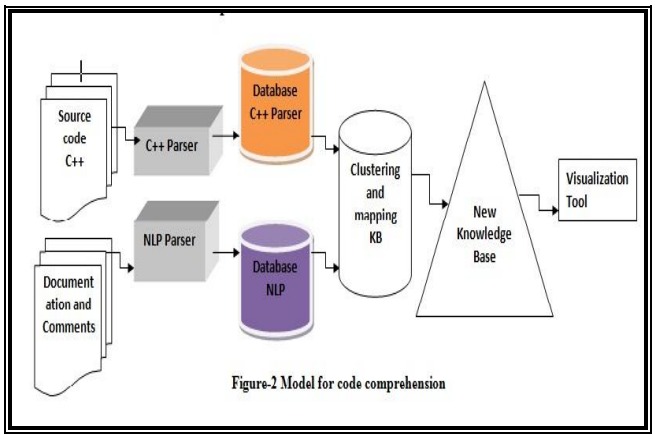

Fig-2 Proposed Model for code comprehension

IV.I PROPOSED AGENT MODEL

We have proposed a new model based on intelligent agents to encode the object oriented semantic knowledge. The agent is having the characteristics like (autonomy, pro-activeness, reactivity and social ability) to understand the others code. The agent model is based on the Belief, Desire and Intension (BDI) architecture to perform the code comprehension task using different agents. The agent based model has been proposed for helping the novice and software maintenance engineer to understand the source code. Figure-2 showing proposed architecture for code comprehension. The agent model is composed of five agents, Agent-1, Agent-2, Agent-3 and Agent-4. ***Agent-1*** **knowledge extractor**, ***Agent-2*** **source code semantic code**, ***Agent-3*** **comments semantic code** and ***Agent-4*** **indicator comparator code.** The remainder agent **(Agent-5)** has been used to just inform the developer before the final report is generated for hit and miss ratio. The remainder agent will highlighted the code which is updated or changed somehow, and also this will highlighted the related comments linked to this keyboard or block of code. as shown in Figure-4.

The belief represents that agent based code comprehension using semantic code for source and comments can improve the code comprehension. The desire or goal represents to improve the code comprehension using the existing knowledge semantic code to understand the code. The intension represents the functionalities of agents or desires. The different agents have been used having unique responsibilities and more intensions. [1]

IV.II WORKING OF AGENT BASED PROPOSED MODEL

The source code is compiled through a parser and comments has designed and identified inside the source code. The knowledge base for C++ source code and comments is extracted into different knowledge bases. The

semantic code is created and on the basis of information retrieval techniques (IR). **Agent-1 knowledge extractor** will collect the both source code and comments semantic code. Then from the Agent-1, the knowledge is extracted, the information message is passed through the Agent-1 for knowledge received. **Agent-2** semantic code for source and **Agent-3** extract comments semantic code, the information message will be done after extraction of semantic code for source code and comments source code through the message passed by Agent-2 and Agent-3 respectively. On the basis of keyboard similarities it will identify the semantic code inside the source code and comments semantics.

It has been observed that, sometimes the problem with the comments is that the software maintainer only see the source code and update the source code only, but sometimes the comments are not updated due to which the understanding of the source code become a problem, that is why the comments are not considered as effective option for code comprehension [6]. The **Agent-4 source code and comments matcher or indicator** is created to identify the commented part which has been mapped to the source code. If during the maintenance, some changes done inside the source code, then the **Agent-4** will response the concept or location where the comments has not been updated, so that the after source code change the comments can also be updated for better documentation and understanding. Agent-4, will give the status inside the code changes and comments not updated. The remainder agent **(Agent-5**) has been used to just inform the developer before the final report is generated for hit and miss ratio. The remainder agent will highlighted the code which is updated or changed somehow, and also this will highlighted the related comments linked to this keyboard or block of code.

The agents interact with each other, and extraction of the semantic code will be done until it received all the source code information from the databases and will be continue unless the agent is passing the confirmation message to each other.

Instructions	Agent-1 Knowledge_extracter	Agent-2 Knowledge_ Source_code	Agent-3 Knowledge_ source_comments	Agent-4 Indicator Comparator_code	Agent-5 Remainder_Agent
Constructor/ Method	Semantic Start_agent() Knowledge_extractor() Exception() Message() Processing() Stop_agent()				
Belief/ Rule		message()	message()	message()	message()
Desire/ Goal	Semantic Knowledge _create()				
Intension/ Plan		Semantic code Knowledge_ Source_code	Semantic code Knowledge_ source_comments	Indicator() Comparator_code()	hit_ratio() miss_ratio()

Table-1 Agent-Prosperities

In the above table the various Agents rules, functioning of each agent has been mentioned. The role of each agent is represented in the agent model. Table-1 is showing belief, desire and intension rules.

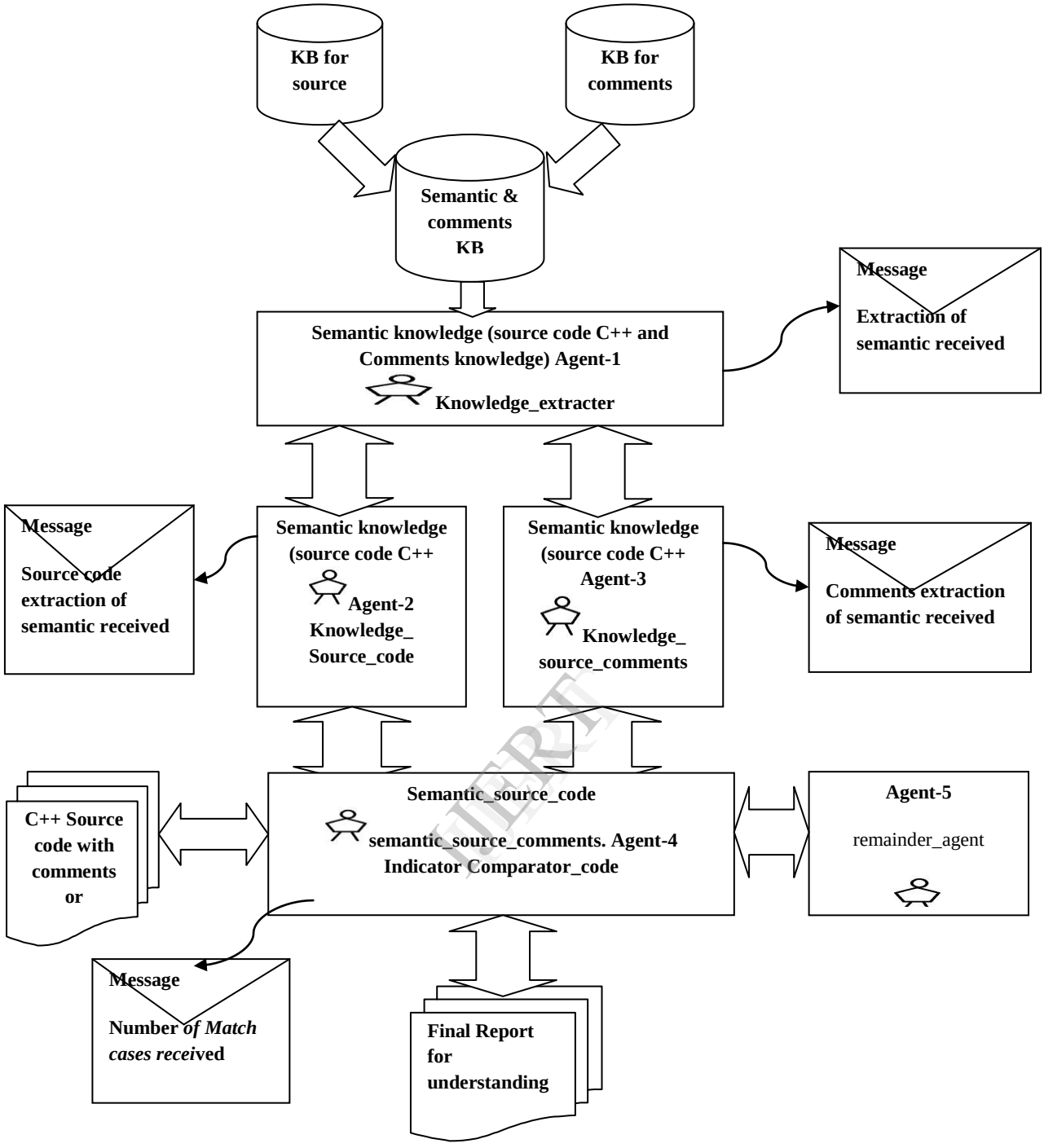

Fig-4 Agent Based Model for Code Comprehension

V. METHODOLOGY

The remainder agent **(Agent-5)** has been used to just inform the developer before the final report is generated for hit and miss ratio. The remainder agent will highlighted the code which is updated or changed somehow, and also this will highlighted the related comments linked to this keyboard or block of code.

Let **S_C** and **C_C** are the two variables for source code and comments code. If 'H' is hit ratio and 'M' is miss ratio. The hit ratio when a particular code matches within the semantic knowledge. Let n is the number of keyword to be search inside the semantic code S_C and comments C_C.

Then the hit ratio, H can be calculated as,

$$H = \sum_{i=0}^{n} (S\_C + C\_C)$$

Where H is the hit ratio and miss ratio M can be calculated as,

M= 1-H

Average of hit and miss cases can be calculated as,

T avg = (H+M)/2

The total number of hit and miss ratio cases will help to understand the code matched and not matched cases. This will help the novice, programmer and maintenance engineer to indentify the cases easily. With the help of visualization tool or highlighted the missed cases i.e. the cases where comments and code semantic not matched in any respect, this task is done at the agent-5 the remainder agent which will get the report of miss cases and highlighted inside the code and comments where there is no match case found.

VI. ALGORITHM AGENT BASED MODEL

Let **S_C** and **C_C** are the two variables for source code and comments code. S_C is source code and C_C (source comment)

S_C=1 and C_C=1 indicates that the source code and source comments semantic knowledge has been received at **Agent-1.**

Step:1
If S_C=1 && **C_C=1** is received **Agent-1 enabled**
Step:2
If Agent-1= 1 Extract the semantic knowledge form **Agent-1** communicate separately the code has been

enabled inside the data marts **Agent-2 (S_C=1) && Agent-3.**
Step:4
If Agent-2 && Agent-3 Match S_C=C_C **then Agent-4 enabled**
else
Agent-5 enabled
end if
end

VII. CONCLUSION AND FUTURE WORK

An agent based comprehension is cost effective. The agent model is composed of five agents, Agent-1, Agent-2, Agent-3 and Agent-4. ***Agent-1*** **knowledge extractor**, ***Agent-2*** **source code semantic code**, ***Agent-3*** **comments semantic code** and ***Agent-4*** **indicator comparator code.** The **remainder agent (Agent-5**) has been used to just inform the developer before the final report is generated for hit and miss ratio. The model contains the extraction, comparison and matching and classification techniques. The implementation of the whole agents will be done using java based jade tool as shown in Figure-6. The future work will be implementation of agent based model in an integrated development environment (IDE) for code comprehension. The Agent based model will help the novice and programmer to understand the code in better ways.

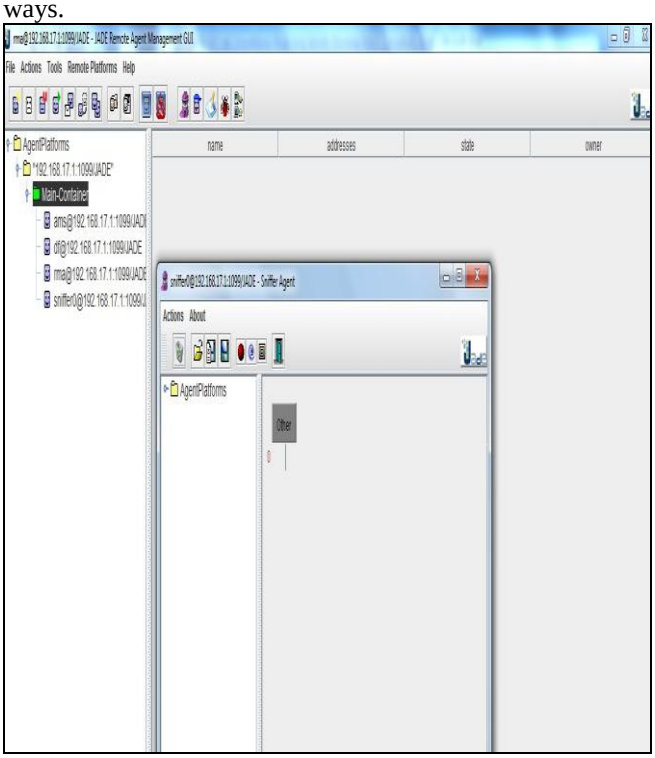

Figure-6 Jade Tool

VIII. REFERENCES

1. Aris.,T.N.M., & Nazeer., S.N., "Object-Oriented Programming Semantics Education based on Intelligent Agents" , IEEE, 2011, pp. 404-407.
2. Cain, J.W., & Mc Crindle, R.C., "Software Visualisation using C++ Lenses", 1998
3. A.V. Mayrhauser and M. Vans, "Program Comprehension during Software Maintenance and Evolution," IEEE Computer, vol. 12, pp. 44-55, 1995.
4. Nouh Alhindawi M.S "SUPPORTING SOURCE CODE COMPREHENSION DURING SOFTWARE EVOLUTION AND MAINTENANCE" Dissertation written by., Al-Balqa' Applied University, Jordan, 2006 B.S., Yarmouk University, Jordan, 2004.
5. Aris., T.N.M "OBJECT-ORIENTED PROGRAMMING SEMANTICS REPRESENTATION UTILIZING AGENTS, Journal of Theoretical and Applied Information Technology 15th October 2011. Vol. 32 No.1 © 2005 - 2011 JATIT & LLS. All rights reserved. ISSN: 1992-8645.
6. Program Comprehension Tools Bibliography: http://www.csc.tntech.edu/~linos/pctinfo.html.