Journal of Theoretical and Applied Information Technology
20th December 2013. Vol. 58 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

COMPUTATIONAL PROBLEM SOLVING ARCHITECTURAL DESIGN BASED ON MULTI - AGENT

[1]MARYAM RAJABI, [2]TEH NORANIS MOHD ARIS, [3]MD. NASIR SULAIMAN
[1, 2, 3] Department of Computer Science, Faculty of Computer Science and Information Technology, 43400 UPM Serdang, Selangor, Malaysia
E-mail: [1]maryam.rajabi2020@gmail.com, [2] nuranis@fsktm.upm.edu.my, [3] nasir@fsktm.upm.edu.my

ABSTRACT

The application of problem solving methods is seemed to be difficult for novice students in computer programming field. Hence, majority of them prefer to go straight to the last stage to collect information by analyzing source code. Indeed, introducing an efficient solution for this problem will help them to figure out programming problems properly as well as saving time. Nevertheless, computational problem solving systems are not as applicable as enough to be contributed to complex problems craving intelligent analysis. So, intelligent agents tie with problem solving methods to conquer the mentioned issue. Here, a new system mapped by prometheus design tool (PDT) has been introduced. Likewise, the textualized problem to be given to the system and then problem analysis chart (PAC), input process output (IPO) chart, flowchart and algorithm will be produced. The designed problem solving system in this work comprises five agents, namely GUI, PAC, IPO, flowchart and algorithm agents interacting with the environment by percepts and actions. Additionally, there exists extraction, transformation and module number generation processes covering with three scenarios: 'Extract Scenario, 'Transform Scenario' and 'Generate Module Number Scenario'. The system specification, the architectural design and the detailed design are produced based on the analysis overview diagram and the scenario diagram.

Keywords: *Computational Problem Solving, Problem Solving Method, Intelligent Agents, Prometheus Design Tool (PDT)*

1. INTRODUCTION

Nowadays, the major problem which novice students in computer programming field are faced, is applying taught problem solving concepts to unfamiliar given problems. There is such a place where problem solving techniques come to help them. Extracting required information from problems by utilizing these methods has turned into a hot topic these days.

Moreover, with the rapid growth in technology, the need of preparing capable programmers drew significant attention to itself more than before. On the other hand, most of the novices in computer programing field prefer to consider the examples of source codes and change them based on the problem posed in their assignments [1], [2]. To ease large-scale understanding of agent applications there exists an urgent requirement for frameworks, methodologies in addition to toolkits that assist the effective progress of agent systems [3]. That is because one of many tasks for which usually agent systems were invented could be the integration between heterogeneous software programs and independently developed agents [4], [5]. At this point, the agents indicate how problem solving techniques are visualized. Here, it is predicated that agent-based models are to decrease programming bugs or errors in the procedure of developing programs [2], [6].

1.1 Intelligent Agent

In this study, an intelligent agent is surely considered as an entity which is of some intelligence. Agent on the side of the users can perform tasks, owing to the autonomous and reactivity of agents in addition to their mobility [5]. Likewise, intelligent agents are is capable of playing on the behalf of the users, thus, they are very likely to dependent on roles which assist designers as well as coders in how to model the intelligent agents. This contributes to complete tasks highly more than those developed for, which is most probably to what happens in the real life, where people discover ways to perform things and also develop their knowledge [4], [7].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Moreover, agents are viewed as the most important paradigms which, not only improve the current techniques in order to conceptualize, designing and also implementing software devices, but also they may solve the legacy software integration problem as well as agents which are flexible problem solvers [5].

Agents are autonomous encapsulating invocation [8]. Although, an object provides methods caused externally, an agent does not provide any control point with external entities [3]. Thus, an agent can be act autonomously since it can operate with no interference of human-being or others and it also controls on their performances and inner state.. An agent is views as social -since it has cooperation with people or other agents which experience their tasks. An agent can be reactive since it –recognises its surrounded environment and in a timely fashion provides responses for these changeswhich – happen in the environment. In addition, an agent can be proactive since it is capable of displaying goal-directed behaviour through by subtracting initiative although it is simply show reaction in response to its surrounded environment. It might be -logical, always acting to obtain experiences for achieving its goals. It never prevents meeting their -aims, and it can learn how to adapt itself to be suitable for - its environment and the -requests of its users [9]. Hence, it can be noted that agents -require the particular computational apparatus in order to –enable run-time decisions - regarding the scope and the nature of their interactions so that it can initiate interactions not foreseen on design time [10].

1.2 Computational Problem Solving

Multi-agent problem solving systems within scientific computation are becoming increasingly complex and they include numerical models of the real life. Therefore, we need mostly to take them into consideration as strategies which provide scientific computing systems aimed at resolving problems cooperatively [4].

Here, before the development of any source code, computational problem solving need to be viewed as the opening step . However, the problem is that the novice students have problems with how to understand problem statements and how to transform them into computational problem solving techniques [6]. Environments designed for problem solving in data mining, computer language C and other languages includes 1) L.E.C.G.O. [11] 2) ONTOIAS [12] 3) JELIOT 3 [13] 4) RAPTOR [14]:

L.E.C.G.O. is considered as an open problem-solving computer learning environment which has been designed for supporting students during learning programming and C programming language. The design of L.E.C.G.O. results in the synthesis and combination of three models: a) the learning model b) the subject matter model, and finally c) the learner model [11].

ONTOIAS (Ontology-supported Information Agent Shell) as an environment used for multi agent technique [12] includes the four significant modules of information agents, such as information searching using OntoCrawler, extracting information via OntoExtractor, information classifying with OntoClassifier, and information presenting/ranking with OntoRecommander. ONTOIAS provides many users with tremendous information integration as well as recommendation ranking [12].

Jeliot 3 is considered as a program visualization tool. Its main is to learn programming in Java for novices. The user interface of Jeliot3 is classified into two -important panes: (1) the code editor as the code - visualized and (2) Visualization pane such four interconnected areas as method, expression evaluation, constant, instance and then array areas of -different visualization components. Jeliot 3 makes supports the agents to visualize pseudo-code language [13].

Another tool i.e. RAPTOR is highly relevant to the source code level. RAPTOR as a visual programming environment help students develop their algorithm envision and also keep away from syntactic baggage. In other words, RAPTOR contributes to students to construct flow charts and the tool which visualize them through the content of variables and arrays[14].

1.3 The Prometheus Design Tool

In this paper, we use PDT specifically developed to support the Prometheus methodology in order to design our system [3]. Prometheus design methodology is an intelligent agent development which gives the ability to handle all development phases is respectively specification, design, implementation and debugging [15]. PDT is based on Java programming language and hence is viewed as platform independent. The Prometheus methodology comprises three following phases [16]: (1) The system specification phase (2) The architectural design phase (3) The detailed design phase. Figure 1 show the notation used in PDT.

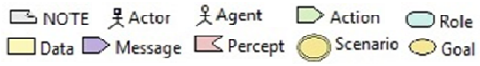


Figure 1: Entity Notation.

Journal of Theoretical and Applied Information Technology
20th December 2013. Vol. 58 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. STEPS OF DESIGN USING PROMETHEUS

2.1 System Specification

It is common to briefly capture the initial ideas for a system in a few paragraphs. Then the mentioned descriptions must be explained in detail to implement a suitable basis for system design and development.

Our Problem Solving Comprehension System is described as a system with pre-programming phase which requires five following agents: extracting simple text formed problems using the GUI agent, analysing the problem using PAC agent, developing the Input-Process-Output (IPO) using IPO agent, drawing the program flowcharts using flowchart agent, writing algorithms using algorithm agent.

The GUI agent is the interface by which the user can interact with other agents. The methodology comprises of extraction, transformation and module number generation processes. At the beginning, the problem transformed to text form should be given to the system by the user. Here, the GUI agent is the interface by which the user interacts with other agents. As a matter of fact, it starts from a text document produced by the PAC agent. The IPO agent is able to extract the needed information from the PAC agent, the flowchart agent can also take information out of the IPO agent and finally the algorithm agent can obtain the needed information from the flowchart agent respectively. Indeed, such agents are at an intersection with each other via sending data including keywords, Input-Process-Output, I-P-O module number and process. Finally, the output of this model is shown in the form of Algorithm. It should be noted that these problem solving stages are related to each other.

2.1.1 Analysis Overview

Centered at top of the first page should be the system specification development which starts with the identification of the external entities (see actors) which interact with the system in some ways. The main scenarios along with interaction will happen are as follow.

This can be performed through PDT using the 'Analysis Overview Diagram'. In Figure 2 we identified 'User' as an actor and GUI, PAC, IPO, FLOWCHART, ALGORITHM as the agents which have interactions with the system. We connect them to the three keyscenarios associated with the system functionality.

Afterward, we refine this diagram through recognising the percepts which for each scenario are inputs -, and the actions which are produced by the system -. They relate them to -the -suitable actor and agents as -indicated in Figure 2. On the other hand, messages have been used to display the results of each step. As an example, user enters a problem as a percept (input) to the system and the system -act the extraction on the given problem. Therefore, the analysis overview diagram illustrates the interaction between the system and environment in the guise of percepts and actions (output) as well as messages. Percepts in our system are User Problem, keywords, Input-Process-Output, Flowchart Process and also Algorithm Process. Likewise, used actions in our system are Extract, PAC transform, IPO transform, Flowchart transform, and ALGORITHM transform.

Our proposed design possesses five messages, namely: 'Keywords', 'Input', 'to flowchart sequence', 'to Algorithm sequence' and 'algorithm'. The agents used in our design are GUI agent that extracts simple text formed problems, PAC agent that analyses the problem, IPO agent that develops the IPO and produce module number, flowchart agent that draws the program flowchart, algorithm agent that writes algorithm. 'Problem statement' includes different sample problems in computer programming that covers: Java fundamental programming, arrays and strings.

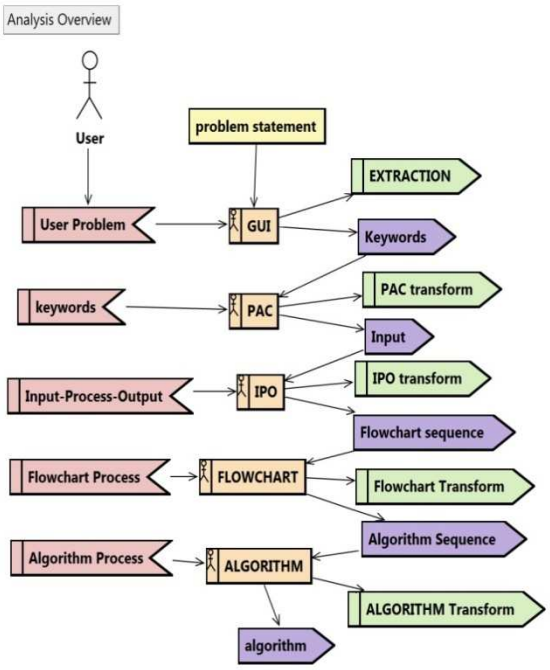


Figure 2: Analysis Overview Diagram.

Journal of Theoretical and Applied Information Technology
20th December 2013. Vol. 58 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2.1.1 Scenarios Diagram

A scenario diagram demonstrates the various scenarios existing in the system. In fact, a series of steps make up a scenario. Each step includes the functionality performing that step, the name of the step, its type (one of action, percept, and goal, scenario or other) and optionally, the information are used and produced by that step. The scenario diagram of our system is in Figure 3.

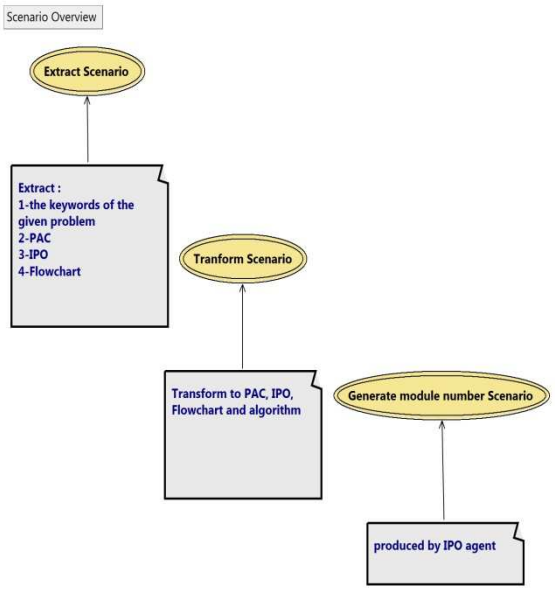


Figure 3: Scenario Overview Diagram.

There are three scenarios in the scenario overview diagram which are 'Extract Scenario', 'Transform Scenario' and 'Generate module number Scenario'. Every scenario has its own steps to be applicable in the system. 'Extract Scenario' consists of extracting keywords from the given problem, PAC, IPO as well as Flowchart. 'Transform Scenario' will transform keywords to PAC, PAC to IPO, IPO to Flowchart and flowchart to algorithm. 'Generate module number Scenario' defines the steps of producing module number by IPO agent. These primary goals, data and roles determined are applied to transfer information into other aspects of the design in an automatic way.

2.1.2 Goal Overview

Figure 4 represents the goals of the problem solving system in which Sub-goals are considered as 'AND' branches. This is the goal based on which the scenario is defined. The name of the goal can be adapted. If preferred, the same goal can be related to multiple scenarios, although this cannot be often the case at the most abstract level of the Analysis Overview Diagram. The goals which areoriginated from the scenarios, can be automatically placed into the 'Goal Overview Diagram', in which goal hierarchies promote how to describe the developed application In order to identify some sub-goals embedded in each goal, it is of necessary to ask the question "how can we acoplish this goal? There exists a typically significant iteraction between the developemt of scenario and the development of goal hierarchy as long as the developer think that the application is adequately described.

Our main goal system is 'pre-programming prep' which is consisted of two sub goals (each sub goal is considered as a goal compared to its sub goals), namely: 'Extract' and 'Transform'. 'Transform' goal aimed at achieving four sub-goals as follow: 'PAC', 'I-P-O', 'FLOWCHART', and 'ALGORITHM'. Two of the above-mentioned sub goals enjoy their own sub goals too. Notably, 'Generate module number' goal is related to I-P-O goal.

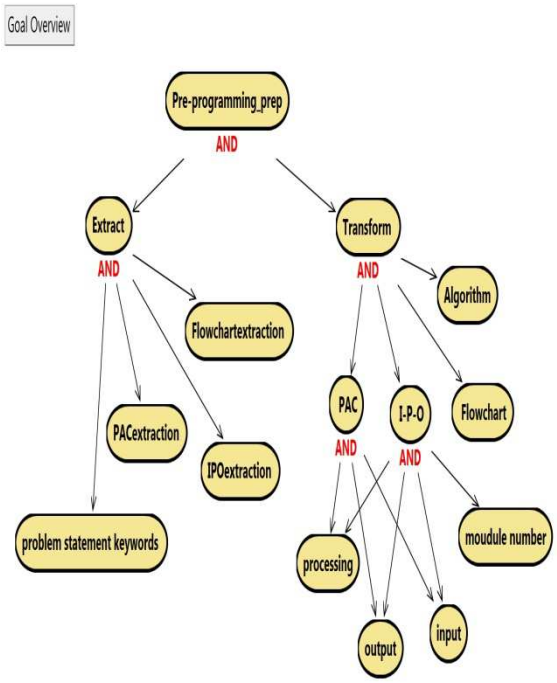


Figure 4: Goal Overview Diagram.

2.1.3 System Role Overview

At this stage, goals are categorised into cohesive units. In fact, their roles are determined as -rather small and easily specified chunks of agent functionality. Then, the percepts and actions also takes roles appropriately to accomplish their goals. This is performed by the System Roles Diagram. For example, Figure 5 represents that the role of 'GUI' is to achieve the goal to extract. To meet this goal, the role needs the inputs (user problem).

Journal of Theoretical and Applied Information Technology
20th December 2013. Vol. 58 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

It should perform the action of extraction on the given problem**.**

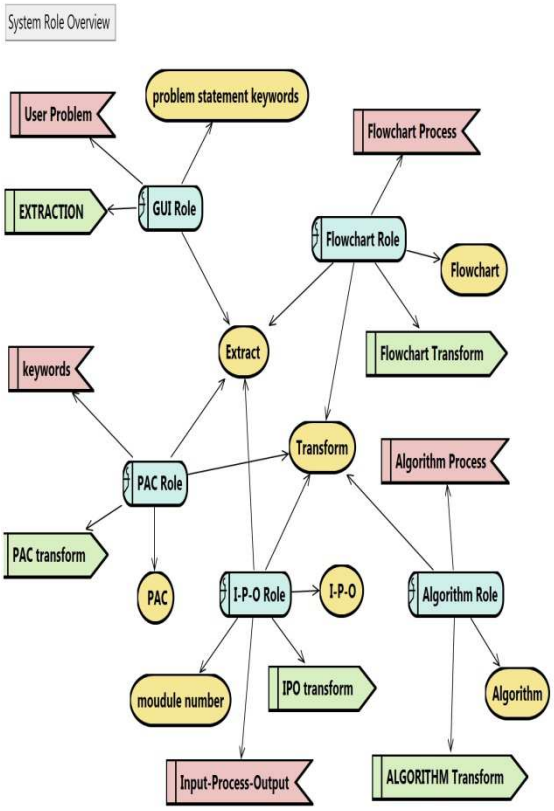


Figure 5: System Role Overview Diagram.

2.2 Architectural Design

The next stage is related to the architectural design in which the internal composition of the device is specified. Here, the significant task is to make decisions on the types of agents (as selections of roles) that could happen as a way to recognise the identified targets and scenarios. Decisions on grouping of roles to agents are made in the 'Agent-Role Grouping Diagram'.

2.2.1 Data Coupling Overview

The Data Coupling Agent Acquaintance diagrams help the designer learn visualising the relationships of roles to data. As seen in Figure 6, in our design each role has connected to its relevant data. As an example, GUI role couples with problem statement data.

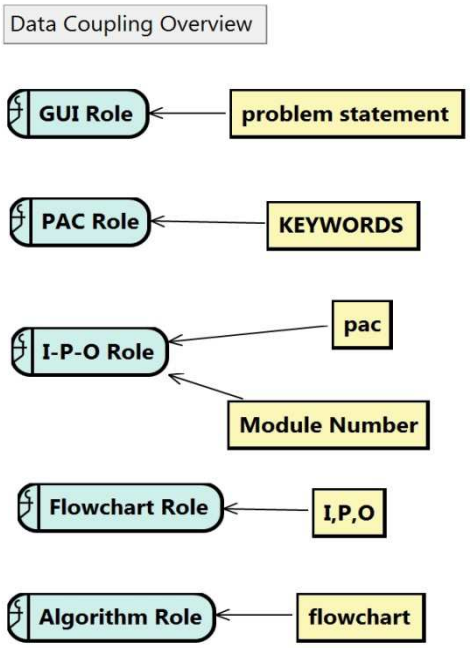


Figure 6: Data Coupling Overview Diagram.

2.2.2 Agent Role Grouping Overview

In this stage we have to define the relationships between agents and roles. It will help the designer to manage which roles are hence to be done by every agent. As understood from above mentioned explanations, one agent is probably associated with more than one roles. However, in our system, each agent is linked to its own specific role.

2.2.3 System Overview Diagram

System Overview Diagram overview the architecture of internal system - In other word, it captures the system's overall (static) structure which brings all the items together. The system overview diagram is viewed as the most -significan product of the design process. It is highly related to agents, data, external input and output together. It also represents how the agents communicate with each other -. To complete this overview, it is needed -that the interactions among the agents are defined and any shared data are added. To put it simple, designer has surely designed its system before reaching this stage but this diagram includes all the needed data. Hence, it will be much easier to refine and review the system design. As far as agents are concerned, they have their own action, percepts and data.

For instance, we can see that observing the 'GUI' agent receives problem (percept) from user and read data from 'problem statement' and then provides an extract action.

Journal of Theoretical and Applied Information Technology
20th December 2013. Vol. 58 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

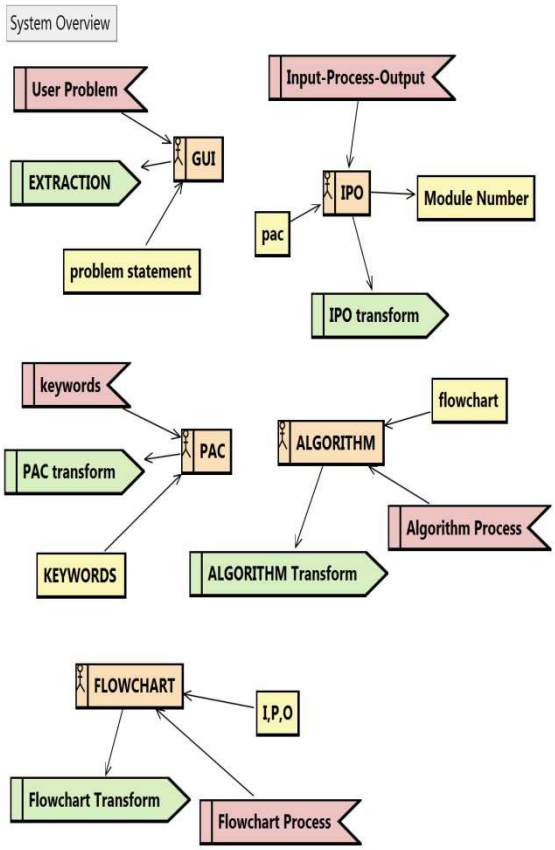


Figure 7: System Overview Diagram.

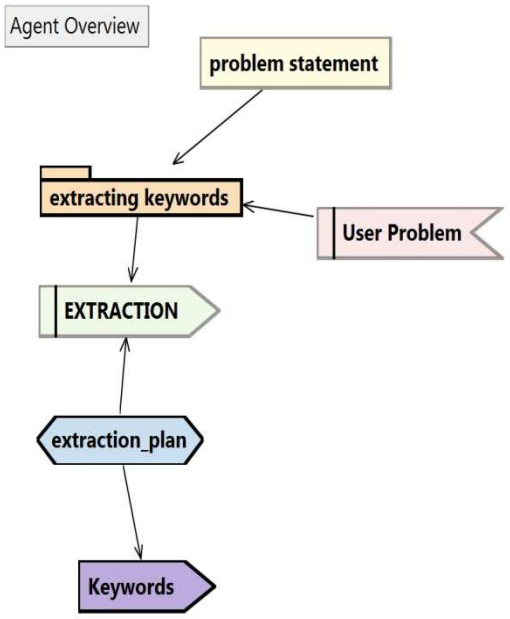
2.3 Detailed Design

Detailed design is the spot where the details of merely every agent's internals usually are developed and defined focusing capabilities, data, occasions, plans and process. Diagrams are utilized as a going stone between interaction protocols and ideas. The detailed design includes:

- Developing the internals of agents, in conditions of capabilities (and, in some cases directly in conditions of events, ideas and data). This can be done using real estate agent overview diagrams and also capability descriptors.
- Develop the information on capabilities considering other capabilities and also events, plans in addition to data.

This is done using capability understanding diagrams and a variety of descriptors. A key target is to develop plan sets to achieve goals and ensure appropriate coverage.

All the entities associated with the agent in the system overview diagram are transferred to the agent overview diagram, such as the individual messages from protocols related to the agent. Entities in an agent/capability overview diagram propagated, form part of the interface to the internals of the agent/capability are represented as



"faded" icons. These interface entities must be associated with internal capabilities or plans which are defined to utilise or generate them. The designer here needs to make sure that all percepts, the actions, messages, and data access is considered. For instance, the capability of 'sending data' is able to control the percept 'DATA' and adapts data in the 'problem statement'.

2.3.1 GUI Agent Overview

Extracting keywords capability receives the User problem percept and read data from problem statement and then problem will be extracted by the Extraction action.

The "extraction plan" is linked to a message to generate and display keywords, when keywords extraction complete successfully, a message will display the keywords.

Figure 8: Gui Agent Overview Diagram.

2.3.2 PAC Agent Overview

PAC agent overview has one capability named 'producing PAC'. 'Keywords' as a percept provided by the GUI agent goes through PAC agent and then the inputs, outputs and processes are extracted from the input text. 'transform_PAC_plan' is a plan associated with 'KEYWORDS' data. The mentioned plan is linked to three messages to display the following results: Input, Process, and Output.

Journal of Theoretical and Applied Information Technology
20th December 2013. Vol. 58 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

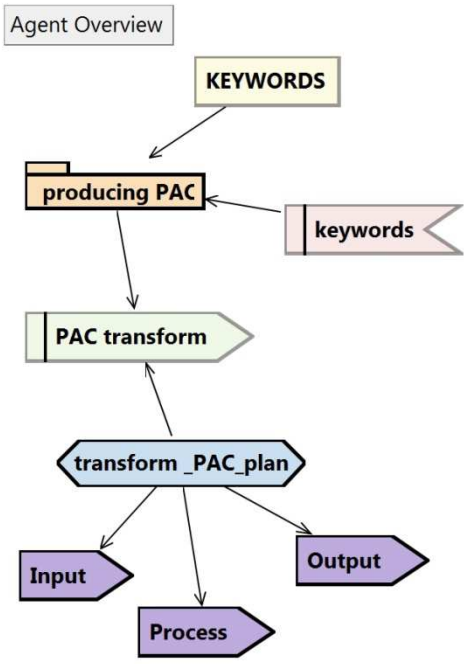


Figure 9: Pac Agent Overview Diagram.

3. DISCUSSION

As regard to complex problems, multi-agent systems are considered as a perfect solving agent since they show the -charecteristics of flexibility, intelligence and solving complex issues -based on -allocated knowledge along with –capabilities updated [17].

Remarkably, designing various constituents of a multi-agent system (MAS) can be a demanding task. In fact, prometheus design tool provides us with refinement model using iterative steps. As a rule, there exist three significant design stages: system specification in which the actors, the inputs, outputs, can utilize scenarios and the goals of the system are -recognised; Architectural design in which agents, roles, communication protocols and the overview of the internals of the system are determined; and detailed design where each agent's internals are -explained to a level which can be immediately implemented. The detailed design would be at a conceptual level and implementation independent, so that it allows the systems to be applied in the choice platform[18].

Apparently, novice students are unable to join the individual statements and constructs associated with IPO chart, flowchart, algorithm into valid programs. Therefore, it can be said that this study aimed at investigating the designed environments for computer programing and possibility utilizing of multi agent systems in problem solving to help them. For the sake of simplicity, iterations and various phases which usually take place in the development process have not explained. The function of problem solving comprehension module is applied for extraction, transformation and also module number generation. The textualized problem goes through detection process to detect words. Then, the word extraction process extracts keywords at the next step. Lastly, the extracted keywords are transformed to its standard form in order to be displayed.

4. CONCLUSION AND FUTURE WORK

This article represents an agent-based system aimed at assisting novice students in computational problem solving. The major reason why agents are suitable for such category is their flexibility in communicating with surrounding environment through percepts and actions. The problem solving system consists of five agents, namely, GUI, PAC, IPO, FLOWCHART, and ALGORITHM. The agents have been defined to accomplish different tasks such as extraction, transformation and module number generation on the way to be achieved system goals. The whole design has been implemented with Prometheus methodology.

As the next step of our future plan, we intend to write a system code using the Java Agent Development Framework (JADE) software since Jade is a java-based framework; even those who have initial information about agent theory are able to construct JADE agent-based systems without facing critical problems. In addition, software developers can program, test and debug agent-based applications easily by the help of the JADE software. The next plan is to develop and improve the problem solving skill of the designed system via covering a wide range of problem statements such as looping and advanced Java Object-Oriented features.

5. ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Teh Noranis Mohd Aris and Associate Prof. Dr. Md. Nasir Sulaiman for their guidance. I would also like to thank Kementerian Pengajian Tinggi (KPT) for the support given under the Fundamental

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Research Grant Scheme (FRGS) University Putra Malaysia, project code number 02-12-10-1000FR.

REFRENCES:

[1] L. Padgham, J. Thangarajah, and M. Winikoff, "Prometheus design tool". *In AAAI*, 2008, pp. 1882–1883.

[2] T.N. Mohd Aris, "Object-Oriented Programming Semantics Representation Utilizing Agents". *Journal of Theoretical and Applied Information Technology*, vol. 31, 2011, pp. 10-20.

[3] L. Padgham, J. Thangarajah, and M. Winikof, "The Prometheus design Tool-A conference Management System case study", *Lecture Notes in computer Science*, 2008, pp. 197-211.

[4] Tamer F. Mabrouk, Mohamed M. El-Sherbiny, Shawkat K. Guirguis, and Ayman Y. Shawky, "A ulti-Agent Role-Based system for Business Intelligence", *Innovations and advances in computer Sciences and Engineering*, 2010, pp. 203-208.

[5] F. Bellifemine, G. Caire, and D. Greenwood, "agent technology overview". *Willey series in agent technology*, 2007, pp. 3-27.

[6] M. RAJABI, T.N. MOHD ARIS, "A Multi-Agent System for Computational Problem Solving -A Review", *Proceedings of International Conference on Advances in Computer and Information Technology (ACIT)*, 2013, pp. 147- 151.

[7] G. Cabri, L. Ferrari, L. Leonardi, "Role-based Approaches for Agent Development", *Proceedings of the 3rd Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, New York, USA, Vol. 3, 2004, pp. 1504-1505.

[8] J.J. Odell, H.V.D. Parunak, and M. Fleischer, "Modeling agent organizations using roles" *Software and Systems Modeling*, vol. 2, no. 2, 2003, pp. 76–81.

[9] L. Padgham, and M. Winikoff, "Developing Intelligent Agent Systems: A Practical Guide". *John Wiley and Sons*, Vol. 13, 2005.

[10] N.R. Jennings, "On Agent-Based Software Engineering", *Artificial Intelligence*, 117, 2000, pp. 227-296.

[11] M. Kordaki, "A drawing and multi-representational computer environment for beginners' learning of programming using C: Design and pilot formative evaluation", *Computers & Education*, Vol. 54, No. 1 , 2010, pp. 69-87.

[12] S.Y. Yang, "OntoIAS: An ontology-supported information agent shell for ubiquitous services" , *Expert Systems with Applications*, Vol. 38, 2011, pp.7803–7816.

[13] R. Bednarik, M. Joy, A. Moreno, N. Myller, and E. Sutinen. "MULTI_AGENT EDUCATIONAL SYSTEM FOR PROGRAM VISULIZATION", *Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)*, 2005, pp.1-6.

[14] M.C. Carlisle, T.A. Wilson, J. W. Humphries, and S.M. Hadfield, "RAPTOR: A Visual Programming Environment for Teaching Algorithmic Problem Solving", *SIGCSE*, Vol. 05, 2005, pp. 176-180.

[15] S. DeLoach, L. Padgham, A. Perini, A. Susi, and J. Thangarajah, "Using three AOSE toolkits to develop a sample design", *International Journal of Agent-Oriented Software Engineering*, Vol. 3, No. 4, 2009, pp. 416–476.

[16] L. Padgham, M. Winikoff, D. Poutakidis, " Adding debugging support to the prometheus methodology", *Journal of Engineering Applications in Artificial Intelligence* , Vol. 18, No. 2, 2005.

[17] M. Morandini, D.C. Nguyen, A. Perini, A. Siena, and A. Susi, "Tool-supported Development with Tropos: The Conference Management System Case Study", *In Agent-Oriented Software Engineering VIII*, Springer Berlin Heidelberg, 2008, pp. 182-196.

[18] H. Sun, J. Thangarajah, L. Padgham, "Eclipse-based Prometheus Design Tool", *In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, Vol 1, 2010, pp. 1769-1770.