2012 International Conference on Information and Computer Networks (ICICN 2012)
IPCSIT vol. 27 (2012) © (2012) IACSIT Press, Singapore

A study of an agent coalition algorithm with software ontologies in Knowledge Grid

Praveen Desai [1], Dinesh Acharya[2] and Ashalatha Nayak[3]

[123] Manipal Institute of Technology, Manipal University, Manipal

Abstract. The effectiveness of a team work not only depends on individual's knowledge; depends on cooperation and precise communication among them. The tradeoffs of remote communication as compared with face-to-face communication are a major issue for geographically dispersed team work. However, the challenges such as what task to be completed, what problems have been raised and clarified, clarity in project plan , availability of resource person and to make just-in-time decision are the major concerns of a software life cycle. Consequently, these issues cause project delay as well as anxiety among team members. The ease of communication can be achieved by aggregating the collective knowledge about the project, the domain knowledge and skills of managing project into a common resource platform with the help of intelligent agents and allow them to share the repository called Knowledge Grid. In this paper, we present the challenges in distributed team environment and ontology involved in software life cycle along with distributed agent algorithm.

Keywords: Knowledge Grid, ontology, agent

1. Introduction

In today's knowledge era, a defining characteristic is our reliance on vast, complex and intertwined information networks. Such networks enable exchange, analysis, and control of information on a scale and of a quality that has never been emphasized [1].These information networks support the critical infrastructure that is responsible for much of the productivity behind our economic growth. Also ensures advances that we contemplate in areas such as medicine, engineering and communication.

As our reliance on these networks grows, so does our vulnerability. The internet enabled us to communicate globally. However the globalization causes not only the rising of communication cost, also the increasing overhead of managing teams when knowledge drain happens [2].There are multiple types of flows which have been investigated viz. energy flow, message flow, control flow. Each of them follows the rules in their respective domain.

This paper investigates: a mapping between software engineering and knowledge grid in section II, followed by different ontologies in software systems in section III, implementation approaches of agents in section IV and section V concludes the paper.

2. Software systems and Knowledge Grid

A distributed team work environment requires team knowledge management. A knowledge flow exists in team work processes and this knowledge flow reflects the knowledge level cooperation in team work, which in turn defines the effectiveness of team work. Distributed software development team focuses on work co-operation and resource sharing between members during software development life cycle and knowledge flow should reflect cognitive cooperation process dynamically. Hence each team member can use experience of predecessor accumulated during previous projects and avoid redundant work. With the advent of the networks [3], the system specification is done in one geographic area and the design in some other place. The entire software development process has distributed resources such as five generic up-level ontologies and a knowledge based [KB] issues and solutions ontology. An issue and solution pair criteria is based on organizational goals, priorities, cost and timeliness. As a result following challenges to be addressed.

a.different terminologies and protocols about principles of software engineering
b.variation in understanding of problem domain
c.various styles of training, project management skills
d.lesser accountability about the project and the implication that it is somebody else's fault
e. redundancy and wastage of time

The purpose of the Knowledge Grid is for sharing and managing globally distributed knowledge resources in an efficient and effective way. The Knowledge Grid is a sustainable human machine interconnection environment that enables people or agents to effectively generate, capture, publish, share, manage and promote knowledge[4], to process any type of resource through machines, and to transform resources from one form to another. It provides appropriate on-demand services to support, innovation, teamwork, simulation, problem solving, and decision making by using sharable knowledge. It incorporates epistemology and ontology to reflect human cognition, exploits social, biological, ecological and economic principles, and adopts the techniques for the future interconnection environment.

3. Ontologies in software systems

In order to develop ontological concepts in software engineering, it is necessary to identify various knowledge domains involved in the process to facilitate the optimum knowledge transfer process.[5] A new paradigm with the use of intelligent agents will help along with identified ontologies. These agents should fulfill characteristics such as

1. classify attributes, roles, and concepts through ontology in software engineering, project management, respective domains

2. identify issues and solutions ontology that rise up during software life cycle

3. effectively communicate with developers and classify queries and provide autonomous answers

Knowledge sharing at different levels may lead to complete or partial reuse, or just a kind of heuristic information which in turn help other team members to accomplish their development task.[6] Based upon the activity, following ontology's can be defined in multisite development activity.

3.1 Ontology on Software Engineering concepts

The software engineering discipline covers aspects of software development such as business function and logic, security and fault tolerance as well as legacy systems. Since each project differs from one another, only a subset of ontology is required. This allows generating a subset ontological knowledge pertaining to software engineering. This leads to instance ontology which is specifically meets a particular project need.

3.2 Ontology on Project Management concepts

Every organization has specific approach towards execution of projects, however it is necessary to have consistent knowledge when discussing the project matters. Hence there should be a generic and specific ontology.

3.3 Ontology on Challenges and solutions concepts

In today's complex scenario, the issues will increase and becomes harder to solve the project progress and lack of just in time solution increase issues like a balloon eventually burst causing project failure. Based on historical data, the project failure is due to requirements -process-product segments. It is necessary to understand how IT can be aligned with business instead of focusing only on technology. The ambiguity of definitions related to the business models, technical terms of software engineering or project management. Therefore all software issues can be classified into 3 major issues namely; Technical-Managerial and Ontological segments.

The above mentioned ontology helps in standardizing business operations and workflow, vocabulary and concepts used across the geography. Thus it incorporates internationally agreed workflow, process, objectives where each organization can customize for their business needs. It helps in no common understanding or unified understanding of the same domain in same project and all team members will benefit by this knowledge from software agents.

4. Role of agents in Knowledge Grid

The representation of software engineering concepts, software development activities[8], software models, processes, issues as well as software documentation using generic and specialize ontology representation will help to provide cognitive, clear, precise concepts and ideas, knowledge and classified issues. The ontology defines the concepts, principles, ideas, knowledge and domain assumptions explicitly, hence allows the complete interpretation and common understanding by teams. These ontologies can be transformed to a software development resource using resource description framework.

Software agent consults knowledge specified in ontology as well as in knowledge base. The ontology is a computer readable description of knowledge. It describes classes of objects such as components, documents, projects et al and their attributes, relationships and processes and their respective instances are stored in databases. Such enumerated knowledge used by agents for getting answers from user queries, making decisions, conveying results autonomously.

The service agent model is composed of four fundamental elements that are *Beliefs*, *Goals*, *Actions* and *Plans*.

Beliefs represent the current state of the agent's internal and external worlds. *Goals* are the set of goals that the service agent wants to achieve. *Actions* are the set of actions that the service agent is able to perform. *Plans* are the set of plans that the service agent has. Each plan is a partially ordered set of activities that is executed in a unit of action.

Through the execution and monitor component, the agent can communicate with the environment and react to the environment. The set of beliefs is the knowledge base for the service agent, which denotes the knowledge about itself and the environment. The knowledge of service agent is classified into three categories that are basic knowledge, constraint knowledge and social knowledge. In order to represent these three categories of knowledge, the beliefs set of service agent is divided into three sub-models that are world model, constraint model and acquaintance model respectively [14].

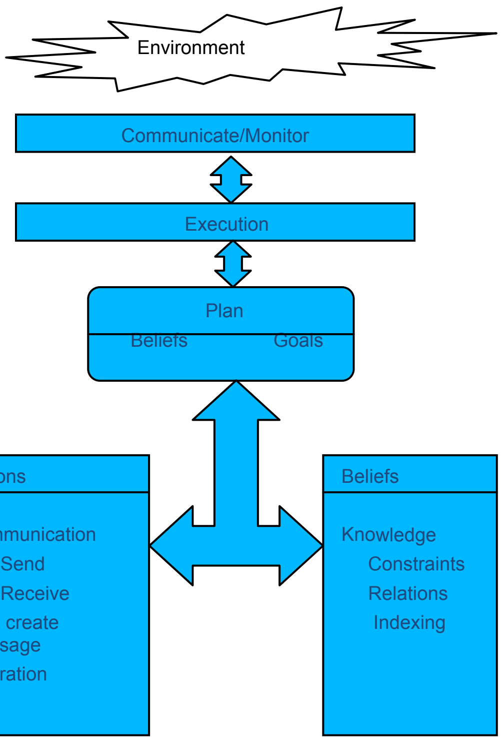

Fig1: Service Agent Model structure

Agent Coalition/coordination Algorithm is based on the decision making of the autonomous service agents and addresses the distributed nature of internet based services. The service agent encapsulates the business logic of how to use the internet platform to achieve a certain business goal, which describes how the web based service operations can be combined, synchronized and coordinated.

Each plan denotes one of the capabilities that the service agent has. The plan is defined as tuple(*Os, Ra, Goal*), where:

(1) *Os* is a set of Web service operations.
(2) *Ra* is a set of relations among Web service operations
in *Os*, *Ra* = *DfCf*, where *Df* and *Cf* are data flows set and control flows set, respectively.
(3) *Goal* is the business goal that the plan achieves,

which is denoted as tuple (*Inputs*, *Outputs*), where *Inputs* and *Outputs* denote the input parameters and output parameters of the plan respectively. During the process of service composition, the plan of a certain service agent can be one of three statuses that are ―*Unexplored*, ―*Exploring* and ―*Explored*.

The status of ―*Unexplored* means that the plan has not been searched, the status of ―*Exploring* means that the plan is being searched and the status of ―*Explored* means that the search for the plan has been finished.

The algorithm includes following three steps.

Step1 Initialization

Once the user submits the service requirement, a user agent is created. Following, the user agent sends a message *Broadcast*(*UserAgent, ri, r*o) to all service agents to notify that a new task arrives. When the service agent *sai* receives the message *Broadcast*(*UserAgent, ri, r*o). The service agent *sai* checks whether there is a plan *p* whose output parameters can help the user agent to achieve the service requirement. If it is true, it sends a message denoted as *Provide*(*sai, p, r*o)∩ *GetGoalOutputs*(*p*)) to the user agent with the aim to tell the user agent that service agent *sai* can provide the parameters set *r*o n *GetGoalOutputs*(*p*) for the user agent by the output parameters of its plan *p*.

```
---------------------------------------------------------------------
1: If(Ms=Null)
2: Notify the user that the requirement can not be achieved;
3: Else
4: Choose the minimal cover solution S with the minimum length
from Ms;
5: Ms=Ms-S;
6: For each (sai, p, x) €S
7: Send the message Request(UserAent, null, sai, p, x) to sai;
8: Sr=Sr U Request(UserAent, null, sai, p, x);
9: End for
10:End if
---------------------------------------------------------------------
```

Step 2 **Backtrack search**

This step is to construct the dependence graph based on the dependence relations among service agents. The user agent checks whether the set of the minimal cover solutions is empty. If it is true, then notify the user that the requirement can not be achieved, otherwise, a minimal cover solution *S* with the minimum length is chosen to search by sending request messages.

Once the service agent *sai* receives *Request*(*s, p', sai, p, x*) message, following algorithm is executed.

According to the status of the plan *p*, two cases are distinguished. One is the status of ―*Unexplored*, which means it is the first time that the service agent receives the request message about the plan *p* and the search for the plan has not been carried out before. The other is the status of ―Explored, which means that the search for the plan p has been finished.

```
------------------------------------------------------------------------
// When service agent sai receives Request(s, p', sai, p, x)
------------------------------------------------------------------------
1: Rrp=Rrp U Request(s, p', sai, p, x);
2: If(p.status="Unexplored")
3: If((GetGoalInputs(p)Ø ri) and (Dssp= Ø))
4: Set p.status="Explored";
5: Set p.feasible=false;
6: Send the message Response(sai, p, s, p', x, false, null) to s;
7: Else if(GetGoalInputs(p) € ri)
8: Set p.status="Explored";
9: Set p.feasible=true;
10: Send the message Response(sai, p, s, p', x, true, pw) to s;
11: Else
12: Set p.status="Exploring";
13: Choose a minimal cover dependence solution S with the minimum length from Dssp;
14: Dssp = Dssp -S;
15: For each Depoi(sai, p, saj, pq, y € S
16: Send the message Request(sai, p, saj, pq, y) to saj;
17: Srp= Srp U Request(sai, p, saj, pq, y);
18: End for
19:Else if(p.status="Explored") AND (p.feasible=true)
20: Send the message Response(sai, p, s, p', x, true, pw') to S;
21: Else
22: Send the message Response(sai, p, s, p', x, false, null) to S;
23: End if

----------------------------------------------------------------------------------
```

5. Conclusion

The Knowledge Grid based methodology is the study of the fundamental principles, strategies and methods for the development and maintenance of the Knowledge Grid as a human-machine environment. Also, the role of software agent as its intelligence in software systems. This paper enriches the content and focuses on the basics, role of agents in software systems, need of knowledge grid as basis of knowledge, and dynamicity is the nature of knowledge sharing with the help of efficient communication between remote teams.

6. Acknowledgement

The authors would like to thank parent institution for their support and necessary guidance along with an anonymous reviewers for their constructive comments and suggestions during the conference.

7. References

[1] Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., and Merlo, E.,"Recovering Traceability Links between Code and Documentation", IEEE Transactions on Software Engineering,2002.

[2] Antoniol, G., Canfora, G., Casazza, G., and Lucia, A.,"Identifying the Starting Impact Set of a Maintenance Request: A Case Study", in Proceedings 4th European Conference on Software Maintenance and Reengineering, Zurich,Switzerland,2000.

[3] Clelang-Huang, J., Settimi, R., Duan, C., and Zou, X., Marcus, A., Maletic, J. I., and Sergeyev, A., "Recovery of"Utilizing Supporting Evidence to Improve Dynamic Traceability Links Between Software Documentation and Requirements Traceability", in Proceedings International Source Code", International Journal of Software Engineering Requirements Engineering Conference (RE'05),2005.

[4] Crestani, F., Lalmas, M., Van Rijsbergen, C. J., and Campbell, I., "Is this document relevant?...probably: a survey of probabilistic models in information retrieval", ACM Computing Surveys, 1998.

[5] De Lucia, A., Fasano, F., Oliveto, R., and Tortora, G., "Enhancing an Artefact Management System with

Traceability Recovery Features", in Proceedings IEEE International Conference on Software Maintenance (ICSM'04), Chicago, IL, 2004.

[6] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T.K., and Harshman, R.,"Indexing by Latent Semantic Analysis", Journal of the American Society for Information Science,1990.

[7] Frakes, W., "Software Reuse Through Information Retrieval", in Proceedings 20th Hawaii International Conference On System Sciences, Kona, HI, 1997

[8] Homes, R. and Murphy, CG.Using Structural Context to Recommend Source Code Examples. In Proc. of Int'l Conf. on Software Engineering,2005.

[9] Zhuge H., X. Sun, J. Liu, E. Yao and X. Chen,"A Scalable P2P Platform for the Knowledge Grid", IEEE Transactions on Knowledge and Data Engineering,vol.7, 2007.

[10] W.Pohs, G.Pinder, C. Dougherty, M. White,"The Lotus knowledge discovery system: tools and experiences", IBM Systems Journal (4),2006.

[11] S. Wang, W. Shen and Q. Hao, "An Agent-based Web service Workflow Model for Inter-enterprise Collaboration", Expert Systems with Application, 2006.