An Agent Coordination Perspective on Software Ontologies with Knowledge Grid

Praveen Desai, Dinesh Acharya, and Ashalatha Nayak, *Member, IACSIT*

***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.**

***Index Terms*—Agents, coalition, knowledge grid, ontology**

I. 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 follow 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 distributed environment in section IV and section V concludes the paper.

II. 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.

III. 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.

Manuscript received March 21, 2012; revised April 29.
Praveen Desai is with the Manipal Institute of Technology, a constituent Institution of Manipal University, Manipal, India.

These agents should fulfil characteristics such as :

- classify attributes, roles, and concepts through ontology in software engineering, project management, respective domains
- identify issues and solutions ontology that rise up during software life cycle
- 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.

A. *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. The use of UML to model the underlying ontology is shown in Fig. 1.

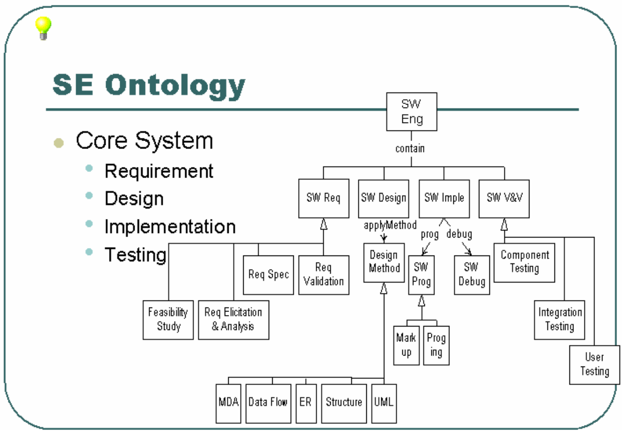

Fig. 1. Illustration of software engineering ontology.

The above mentioned ontology represents the commonly agreed knowledge along with relationships between the concepts and notations, documents and tools[7]. Hence we can propose the platform divided into various segments of knowledge.

Thus we can define five different knowledge levels according to member's cognitive characteristics.

Code Level knowledge helps the team members to share programming skills during development activity described as a set of problem solution pairs.

Component Level knowledge reflects reusable components being developed by the corresponding team members,

Method Level knowledge enables the related team members to reuse the problem solving method described as problem-method pair for process, pattern, algorithms applicable.

Rule Level knowledge defines the knowledge cooperation rules based on work flow execution. These rules may help new team members to cooperate with the team in condition-action-result [CAR] form.

Decision Level knowledge provides the reference for succeeding team members to make their decisions as per Situation-Evaluation-Decision pattern.

The ontology hierarchy based on knowledge levels is shown in Fig. 2.

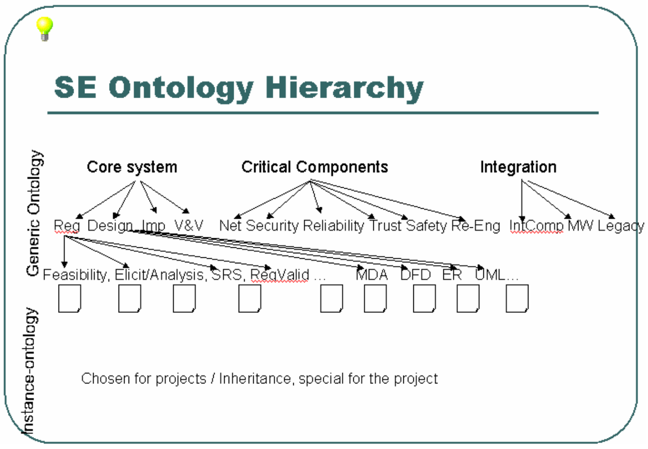
SE Ontology Hierarchy
Generic Ontology
Instance-ontology
Core system
Critical Components
Integration
Req Design Imp V&V
Net Security Reliability Trust Safety Re-Eng
IntComp MW Legacy
Feasibility, Elicit/Analysis, SRS, ReqValid …
MDA DFD ER UML…
Chosen for projects / Inheritance, special for the project

Fig. 2. Hierarchy of software engineering ontology.

B. *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 as shown in Fig. 3.

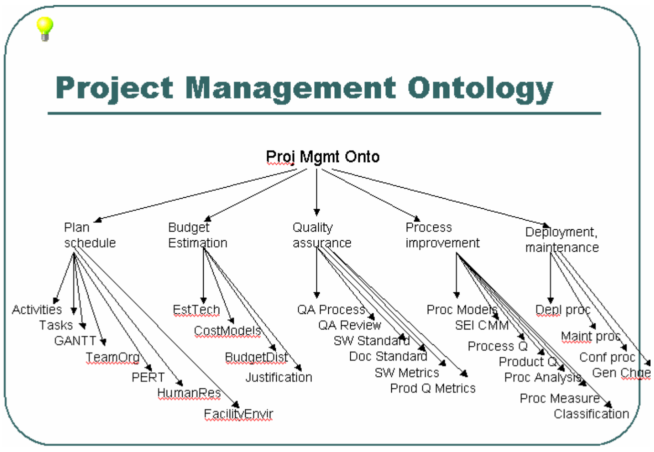

Fig. 3. Example of Project Management ontology.

C. *Challenges and Solutions Ontology:*

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. This is shown in Fig 4.

Any issue raised initially passed through Knowledge grid platform where software agent carries out initial communication with members and classifies the problem or issue. The problem-solution ontology can be defined as an organizational strategic metrics such as task, goal, budget and time.

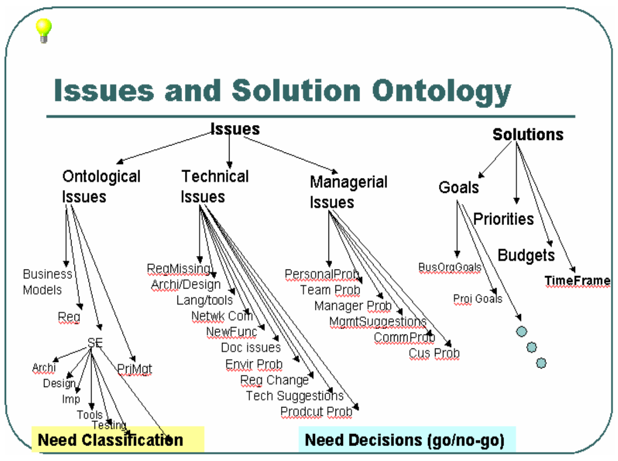

Fig. 4. Example for issue-solution ontology.

D. *Ontology Based on Business Concepts*

A fundamental knowledge of business related domain/area such as financial, logistics, retails etc. is necessary for the success of the software implementation. Hence it is necessary for project architects or managers to work in such industry to understand the process completely. This helps in gaining the fundamental knowledge for the team even when the software developer has little or no knowledge about the domain. Thus the entire business process is represented in this ontology. This is illustrated in Fig. 5.

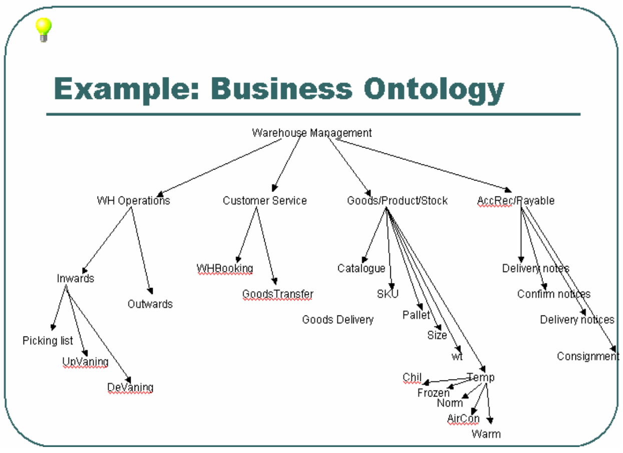

Fig. 5. Example of domain specific business ontology.

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.

IV. Role of Agents in Knowledge Grid

An agent as referring to a component of software and/or hardware which is capable of acting exactingly in order to accomplish tasks on behalf of its user. Given a choice, it is an umbrella term, meta-term or class, which covers a range of other more specific agent types.

A. *Characteristics of Software Agent*

An agent system is essentially a component system exhibiting several of the characteristics. There is six orthogonal characteristics [7] work together to make agent-oriented systems more flexible and robust to Change.

1) Adaptability - The degree to which an agent's behaviour may be changed after it has been deployed.
2) Autonomy- The degree to which an agent is responsible for its own thread of control and can pursue its own goal largely independent of messages sent from other agents.
3) Collaboration - The degree to which agents communicates and works cooperatively with other agents to form multi-agent systems working together on some task.
4) Knowledgeable - The degree to which an agent is capable of reasoning about its goals and knowledge.
5) Mobility - The ability for an agent to move from one executing context to another, either by moving the agent's code and starting the agent afresh, or by serializing code and state, allowing the agent to continue execution in a new context, retaining its state to continue its work.
6) Persistence - The degree to which the infrastructure enables agents to retain knowledge and state over extended periods of time, including robustness in the face of possible run-time failures.

B. *Agent Decision Making Process:*

Interaction is the main driver for agent decision making. The difference between *reasoning* and *decision making* is that reasoning is based on "single thread of control" practice [8]. That is, in reasoning only one decision making entity is present and active. However, in decision making, multiple decision making entities are running simultaneously and independently and the outcome of one's action may affect the others.

An agent's decision making process is summarized as follows:

1) Gathering relevant data/information from the other agents.
2) Organizing and interpreting data/information.
3) Identifying the interaction class and appropriate decision making methods.
4) Building the representation model based on the interaction class, using different classes of games in game theory, utility theory or other uncertainty management theories.
5) Calculating the expected utilities of all possible alternative solutions associated with each class of games, selecting the best one, and using it to select a proper action.
6) Taking action.
7) Evaluating the results.

Basically, each agent should decide firstly its engagement with the other agent belongs to which interaction class (step 3) and secondly try a decision making model for that scenario (steps 4 and 5). The problem of steps 4 and 5, i.e., how different decision making methods can be adopted when an agent is in competitive MAS environment and how to deal with different levels of uncertainty were addressed. Game theory and decision theory are considered to be the fundamental theories to handle an agent's decision making under uncertainty. They provide a powerful tool and a set of mathematical techniques for making decisions about the actions to take when the outcomes of the possible actions are not certain.

The representation of software engineering concepts, software development activities [9], 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.

The Environment is the outside world in a distributed scale.

- This could be invokable applications which are interacting with the system
- A human interface that is part of work place environment
- autonomous activity that implements the tasks without human intervention

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 plan of service agent encapsulates the business logics of how to use the Web service operations to achieve a certain business goal, which dictates how the Web service operations can be combined, synchronized and coordinated. Each plan denotes one of capabilities that the service agent has.

The Execution and Communication component helps the agent to communicate and react with the environment.

- include issues such as trust, reputation, obligations, contract management, and management of large-scale open systems.
- provide implementation methods and middleware, enabling the easy creation of infrastructures for agent-based systems,
- Standardised methods for discovery and communication between heterogeneous services.

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

Beliefs represent the current state of the agent's internal and external worlds.

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 [11].

Agent Coalition or 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 [12] 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.

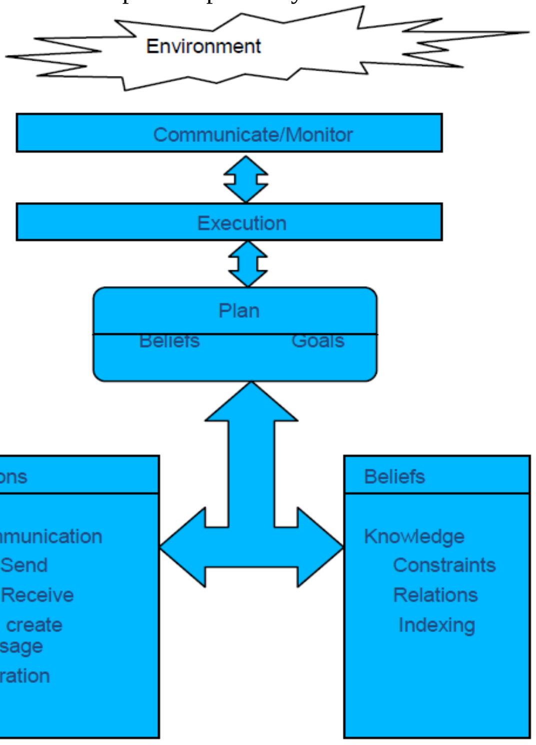

Fig. 6. Service agent model structure.

During the process of service composition [13], the plan of a certain service agent can be one of three statuses that are *Unexplored*, *Exploring* and *Explored*. The status *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.

Dependence Relation [DR]:

An agent is said to be dependent on another if the latter can help to achieve its goal [14]. Considering Agent Services Si and Sj such that Si={ Ai,Bi,Pi} ;Sj={Aj,Bj,Pj}

If PLAN 'i' of Si is dependent on PLAN 'j' of Sj, then dependency relation DR (Si,Pi,Sj,Pj,x)
where x-> parsing of [Goal Pi ∩ Goal Pj].

If Goal Pi = Goal Pj => independent Relation else dependence Relation.

Using DR, the dependence graph [DRG] /directed graph among service agent can be constructed as DRG{ V, E}where V is a vertex or node and E is Edge or a link.
The proposed methodology includes following 3 steps [14], [15], [16].

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(Message Ms =Null)
2: Notify the user that the requirement cannot 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(UserAgent, null, sai, p, x) to sai;
8: Sr=Sr U Request (UserAgent, null, sai, p, x);
9: End for
10:End if
```

Step 2 Tracing Phase

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 cannot 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: Set p.status="Exploring";
8: Choose a minimal cover dependence solution S with the
   minimum length from Dssp;
9: Dssp = Dssp -S;
10: For each Depoi (sai, p, saj, pq, y € S
11: Send the message Request (sai, p, saj, pq, y) to saj;
12: Srp= Srp U Request (sai, p, saj, pq, y);
13: End for
14:Else if(p.status="Explored") AND (p.feasible =true)
15: Send the message Response (sai, p, s, p', x, true, pw') to
S;
16: end if
```

Step 3 Forward Phase: When service agent SA receives the Response Message RSm, it forwards the same to other SA. Thus the communication among peer SA's will be established.

V. Conclusion

Agent-oriented techniques represent an exciting new means of analysing; designing and building complex software systems.Agents collaborate by means of social networks. They can produce the high quality solution at a low cost of communication and addresses the distributed nature of Web service composition.

A formal service agent model is proposed, which integrates the Web service and software agent technologies into a cohesive entity. Based on the service agent model, a agent coalition algorithm in distributed environment for autonomic Web service composition is presented, which formalizes the Web service composition as a graph search problem according to the dependence relations among service agents.

The future work challenges the need of enterprise framework to establish the team based learning.

Acknowledgment

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

References

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

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

[3] G. Antoniol, G. Canfora, G. Casazza, and A. Lucia, “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.

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

[5] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora, “Enhancing an Artifact Management System with Traceability Recovery Features,” in Proceedings *IEEE International Conference on Software Maintenance* (ICSM'04), Chicago, IL, 2004.

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

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

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

[9] H. Zhuge, 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, and M. White, “The Lotus knowledge discovery system: tools and experiences,” *IBM Systems Journal*, vol. 4, 2006.

[11] C. Lin , K. M. Kavi, F. T. Sheldon, K. M. Daley, and R. K. Abercrombie, “A methodology to evaluate agent oriented software engineering techniques,” *Software Agents and Semantic Web Technologies Minitrack*, 2007, IEEE Proc. HICSS-40

[12] B. H. Sellers, “Evaluating the feasibility of method engineering for the creating of agent-oriented methodologies,” M. Pechoucek, P. Petta, and L. Z. Varga (Eds.), *CEEMAS* 05, pp. 142–152, 2005

[13] G. Sakarkar and N. M. Shelke “A New classification Scheme for Autonomous Software Agent,” *IEEE Int. Conf. IAMA’09*, 2009.

[14] G. Sakarkar and S. Upadhya, “A Survey of Software Agent and Ontology,” *International Journal of Computer Applications* (0975 – 8887), vol. 1, no. 7, 2010

[15] F. Zambonelli, N. R. Jennings, and M. Wooldridge, “Multi-Agent Systems as Computational Organizations: The Gaia Methodology,” In B. Henderson-Sellers and P. Giorgini (Eds.), Agent-oriented methodologies, 2005, ch. 6, Hershey, PA: Idea Group.

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

Praveen Desai obtained his Master’s degree from National Institute of Technology, Surathkal and Engineering degree from Mangalore University. He started his career as an IT professional before joining academics. He is pursuing his research interest in Machine Learning. Currently he is working for Manipal Institute of Technology, a constituent Institution of Manipal University, Manipal ,India.

Mr. Desai is an active member of professional bodies such as Indian Society for Technical Education [ISTE], IEEE and International Association of Computer Science and Information Technology [IACSIT], Singapore. He has published research papers in International conferences. He is an invited speaker as well as a well-known resource person.