Information & Management 40 (2003) 419–429

www.elsevier.com/locate/dsw

Workflow- and agent-based cognitive flow management for distributed team Cooperation

Hai Zhuge[*]

Knowledge Grid Group, Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, PR China

Received 25 June 2001; received in revised form 5 January 2002; accepted 24 May 2002

Abstract

Human cognition and work processes are two inseparable parts of human problem solving, however, the cognitive process is neglected in most research on knowledge-intensive team Cooperation. This paper presents an approach for modelling these two parts of the process for use by a distributed co-operative team. We define a new notion of cognitive flow to reflect the dynamic cognition processes of a team. The mechanism is built to model, control and manage the cognitive flow process. A solution for applying this approach to a distributed team during software development is presented. The experiment showed that the approach can improve the problem solving ability of a team.

Keywords: Agent; Cognition; Cooperation; Knowledge management; Teamwork; Workflow

1. Introduction

People usually solve large-scale problems as a team where all the members make a co-operative effort to achieve a common goal. The process, however, may need the team members to be decentralised in order to make use of site-specific resources. Team members then must Cooperate in either a loosely coupled way, like a research or a robot football team [30], or a tightly coupled way, like a production line, depending on the application requirements.

Knowledge management plays an important role in promoting the team's work effectiveness. It concerns the management aspect (e.g. the organisational learning, organisational behaviour and organisational culture) and technical support [3,4]. The scope of this paper is on tightly coupled Cooperation teams and their technical support.

Workflow, the automation of a business process in whole or part, is a useful tool for modelling and managing a business process. It can enforce tightly coupled co-operation between team members according to a pre-defined logical process between activities (tasks). A workflow management system (WfMS) is a system that defines, manages and executes workflow specification through the execution of software whose order of execution is driven by a formal representation of the workflow logic [15,16,31,32]. The workflow establishes the logical execution ordering between team members' activities, and therefore, it helps define the intra-enterprise or inter-enterprise business process [27]. Time modelling, reusability, exception handling, distribution, adaptability and formal modelling of workflow have been discussed in several articles (e.g. [5,20,21,24,35]) and these features,

[*] Fax: +86-10-62567724.
E-mail address: zhuge@ict.ac.cn (H. Zhuge).

together with resource-sharing, should be incorporated into the current WfMS to provide better support to the distributed team.

The implementation of every task in a workflow is an interaction between team members and the support environment. In knowledge-intensive teamwork, the members' cognitive abilities and their Cooperation determine the efficiency and quality of their performing the team task. Thus, the team members actually cooperate at three levels, from low to high: work Cooperation, resource (information, knowledge and services) sharing, and cognitive Cooperation. At the *work Cooperation level*, team members implement their tasks according to the team's workflow definition. At the *information sharing level*, team members communicate to share information based on a pre-defined sharing paradigm. At the *cognitive Cooperation level*, team members learn from each other, make abstractions, make across-problem analogies, and use past experience and skills to solve new problems [6,7,33]. The current workflow model and WfMS do not include such a cooperation level.

Internet techniques can help a globally distributed team cooperate, but team members are likely to be changed more frequently. Those members who leave take away information, knowledge and experience that they acquired, and the replacement team members have to spend time learning the co-operative rules and problem solving methods and accumulating the information resources and experience from scratch. This addition will usually slow down the work of the team [2].

Another challenge is the contradiction between globalisation and co-ordination. A globally distributed team has the advantage of making use of global resources, but the complexity of the co-ordination as well as the costs of communication and transporting products will also increase. The co-ordination involves, task planning including such considerations as time difference and establishing a mutual-understanding. These require a distributed team development environment to support cognitive Cooperation as well as work Cooperation.

Similar problems apply in multi-agent Cooperation for distributed artificial intelligence [1,8,9,14,17,29] and decision-making in group decision support systems [10], as well as a collaborative knowledge base [12] and collaborative design [23]. They focus on the computational principles, the operational models, and the support systems for interaction and co-ordination between team members, and the application of these principles and models. The dynamic decision, Cooperation, and creative user interfaces were investigated in [28,34]. However, previous research has neglected the evolution of the team's cognitive ability during Cooperation and the management of this evolution to enhance the problem solving ability of the team.

This paper proposes an approach that aids the cognitive Cooperation in a workflow-based team so that its work efficiency and problem solving ability can be enhanced. Consequently, two issues need to be investigated: (1) the process modelling and the evolution process of the team's cognitive information and (2) the management mechanism to help in Cooperation.

2. Agent-based workflow: modelling active human Cooperation

Conventional workflow models are activity-based. All the activities (tasks) and their order of execution are pre-defined at build-time. Any performer has to follow these activities into the order defined by the work-list of the related WfMS. However, this model is not in line with real people-centred team works [18,39] and performers' active characteristics are not considered. To overcome such shortcomings, we incorporate agents into the traditional model to form an *agent-based workflow model*. This consists of a set of agents, a set of *inter-agent workflows*, and a set of *intra-agent workflows* for every agent. The process semantics of both the inter-agent workflow and the intra-agent workflow are defined by the traditional workflow semantics. Every agent represents one or more team member, and it can perform the activities in the inter-agent workflow according to the agent's internal work process definition.

An actively co-operative team can be modelled by a set of agents, a team manager, and a set of pre-defined inter-agent workflows (denoted as *InterAgentWFS*) corresponding to different team tasks, represented as $AgentTeam = \langle \{Agent_1, \ldots, Agent_n\}, TeamManager, InterAgentWFS \rangle$. The team manager is also an agent who is responsible for the membership management at the build-time of the team. When receiving a new task, the team manager is responsible for planning the new workflow that implements the new team-task, deploying sub-tasks to the member agents, and monitoring the execution of the inter-agent workflow in run-time. The type of a team-task T is implemented through the co-operative work of the member agents according to the corresponding team workflow definition, represented as $AgentTeam(\mathrm{T}) = \langle\{Agent_1, \ldots, Agent_n\}, InterAgentWF(\mathrm{T})\rangle$, i.e. the inter-agent workflow varies with the type of the task received. The inter-agent workflow concerns multiple flow types, such as control flow, data flow, and material flow between agents. These should be reflected in the workflow definition database of the WfMS in order to manage multiple types of flows. During the execution of the inter-agent workflow, some sequential activities can be simultaneously active in the same time period (e.g. the activities in the production line), and this forms an *active activity segment*. In this case, the WfMS should be able to monitor the propagation of the active activity segment.

Every agent performs its activities to accomplish its task *t* (a part of the team task T) according to its intra-agent workflow, i.e. $Agent_{\mathrm{i}}(t_{\mathrm{i}}) = \langle ActivitySet(t_{\mathrm{i}}), Intra\text{-}agent, WF(t_{\mathrm{i}})\rangle$. The conventional workflow model does not provide such a workflow to the activity performer. The intra-agent workflow provides flexibility in performing the activities on the work-list. The activities of different agents concern different inter-agent workflow. The intra-agent workflow only affects control flows like the traditional workflow. Agents' activities are executed according to the control flow and the condition defined by the intra-agent workflows. Any continuous activities may be regarded as a single atomic activity in the intra-agent workflow. Any constraint on the intra-agent workflow progress, if any, is initiated by the inter-agent workflow. The start of an activity depends on the execution order and the status of the inter-agent workflow in which it participates.

The agent in this model has a communication interface, a reasoning mechanism, a set of event condition action (ECA) rules, and an intra-agent workflow. It receives information (demands) through its input interface. The output is a set of behaviours related to the input and results from its reasoning mechanism, which executes with the ECA-rule set in simulating the active activities such as negotiation, decision, and exception handling. The condition and activity in the ECA rules include a time constraint. The member agents can be managed in either a loosely or a tightly coupled way. In the *loosely coupled way*, each agent is autonomous and can serve more than one team. Each agent can plan to perform its tasks in an optimal way. In the *tightly coupled way*, each member agent can only belong to one team and perform the assigned team task. The team is authoritatively managed, only the team manager has the right to communicate with other teams. Besides participating in the inter-agent and the intra-agent workflows, agents can further cooperate at the cognitive level.

3. Cognitive flow in a co-operative team

Cognitive flow occurs during the passing of team members' cognitive information generated during the knowledge-intensive Cooperation process through a definite media. It records the cognitive information of the current working team member, passes a team member's cognitive information to another team member (receiver) according to a definite process logic, shares its content with the receiver, and accumulates the receiver's cognitive information. The complete cognitive flow passing through all the team members (human or agent) during a team Cooperation process constitutes a *cognitive flow network* (CFN). Every node of the CFN represents the generation of a team member's cognitive information during the team member's task implementation process. The output of each cognitive node is a flow that depends on the team member's cognitive ability and his or her input (i.e. the experience of the relevant team member). We call a cognitive node *active* only when the corresponding team member is working on it, otherwise it is inactive. An inactive cognitive node is re-activated when the corresponding team member starts working on it according to the process logic of the CFN. The propagation of the active node(s) in a CFN can keep pace with the execution of the corresponding inter-agent workflow process.

The relationship between the cognitive flow and the cognitive node of a CFN can be defined by referring to the workflow network. Fig. 1 shows the constitution of the input and output cognitive flows under the context of the agent-based workflow. $\mathrm{CN_i}$ denotes the internal cognitive process of an agent participating in an inter-agent workflow. Its input

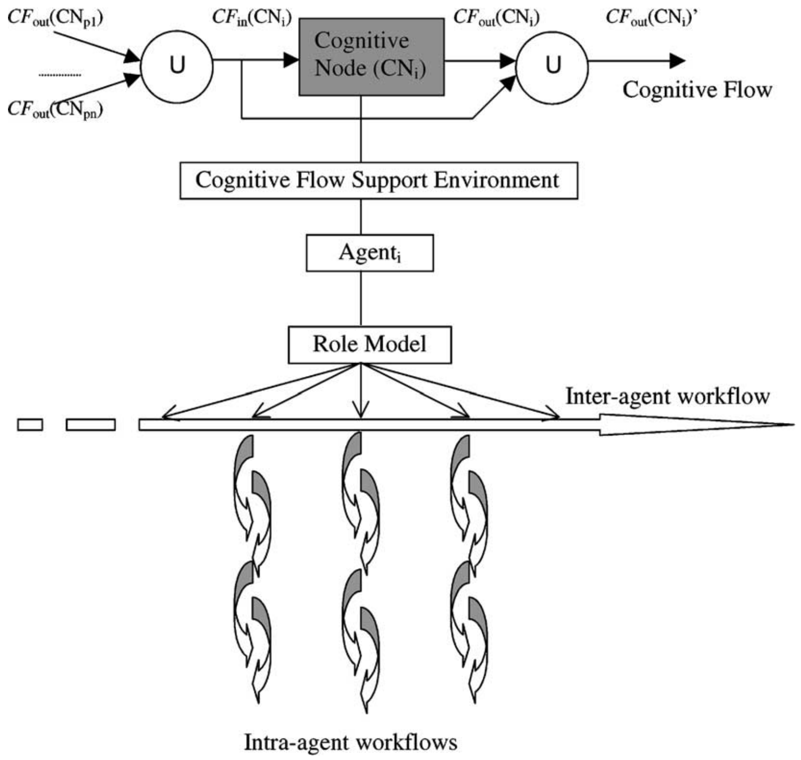

Fig. 1. Relationship between cognitive flow, cognitive node, intra-agent workflow, inter-agent workflow and participants.

cognitive flow is the 'union' of the flows received from its predecessors, it is represented as: $CF_{in}(CN_i) = CF_{out}(CN_{p1}) \cup \cdots \cup CF_{out}(CN_{pn})$.

The cognitive output of CN_i (i.e. $CF_{out}(CN_i)$) is generated by the corresponding team member during his or her work period. The cognitive flow must accumulate the cognitive information generated by the previous nodes, so the final output of CN_i, is the 'union' of the input $CF_{in}(NC_I)$ and the output $Cl_{out}(NC_I)$, i.e. $CF_{out}(CN_i)' = CF_{out}(CN_i) \cup CF_{in}(CN_i)$.

For a newly created CFN, the cognitive input of the first cognitive node $CF_{in}(CN_0)$ will be initialised by the co-operation rules of the development team. After the first run of the CFN, the output of the end cognitive node of the last run will be rolled back as the input of the first cognitive node of the current run of the CFN.

The union of cognitive flows can be a simple merging, a summarisation, or a classification of all relevant flows. A team consisting of members with different knowledge structures will produce different cognitive co-operation effects, so team members should be well organised, e.g. into a hierarchical structure where people at a high level or high rank are responsible for solving more general problems, and people at the same level can share cognitive information more effectively and easily.

There are four major differences between cognitive flow and the workflow. First, cognitive flow reflects the cognitive Cooperation between team members, while workflow only reflects the work Cooperation in the business process. Second, the cognitive flow content is generated from the team members' task implementation process during the execution of the workflow process; it cannot be pre-designed, but the workflow reflects the business process and may be pre-designed by its designer. Third, the cognitive flow carries the cognitive information of the team, while the workflow reflects either the data dependence relationship or the logical execution dependence relationship between task implementation activities. Fourth, the cognitive flow will increase its content during its execution, while workflow just reflects the control of the activities.

The representation of a cognitive flow should have four basic information processing capabilities:

(1) *Accumulation*: it should be able to accumulate the knowledge of all the team members during the problem solving process and retain their knowledge generated during the solving of previous problems.
(2) *Classification*: it should classify the knowledge in different problems and with different team members. This classification reflects the human cognitive information refinement ability.
(3) *Abstraction*: it should reflect the cognitive information at different abstraction levels. With abstraction, human can simplify a complex problem and solve it first at the high abstraction level, then mapping it onto low levels.
(4) *Analogy*: it should establish analogy associations between the related contents. With analogy, people can solve a new problem by using their prior problem solving experience.

All the team members are both writers and readers of the flow content, so the content description of the cognitive flow should be understandable by all members. The implementation of the cognitive flow consists of two levels: (1) the constrained natural language for communication between the human team members and their agents and (2) the communication language and paradigm between agents. Establishing creativity in the agents' interfaces is very helpful in promoting Cooperation between human and software agents. Since different agents may have different software support systems, the cognitive flow should be based on a common information exchange language. Several languages for this communication have been suggested [13,19]. The Semantic Web (http://www.semanticweb.org) enables the cognitive flow to be expressed and transmitted across the Internet.

4. Cognitive flow management

A team's problem solving ability varies with the knowledge structures of its members. A candidate should be evaluated according to the following four aspects when selecting him or her as a member.

(1) *Specialisation knowledge of the problem field* (including basic concept, theory, method, experience and skill): Evaluation can be carried out by examining and reviewing the candidate's experience.
(2) *Knowledge of related fields*: This is the basis of making associations when solving problems. Evaluation can be carried out by reviewing the candidate's experience.
(3) *Creativity*: This evaluation is for determining whether the candidate can make use of past experience. Evaluation can be carried out by assigning the candidate a set of pre-designed problems beyond his or her major specialised knowledge area.
(4) *Co-operative spirit*: This can be tested and evaluated by asking the candidate a set of well-designed questions on the area.

A team's Cooperation degree can be measured by assessing the following factors:

(1) the match between the special knowledge required for solving the problem and the team's overall special knowledge;
(2) the degree of similarity between team members' knowledge;
(3) the average creativity and co-operative spirit of the team members.

Good coverage of related fields can improve the team's problem solving ability and help members learn from one another. Once a team is formed, its knowledge structure will evolve continuously, because every member will study new knowledge and will change during the problem solving process. The evaluation of the team's degree of Cooperation is dynamic, so the evaluation approach reported in [38] can be used.

Each team member is responsible for not only the generation of the cognitive flow but also its generalisation. The generalisation of each team member's cognitive information depends on the following three types of input:

1. Cognitive information generated while performing the current task.
2. Cognitive flow output of the direct predecessor(s). The input of the first cognitive node can be the roll back cognitive output of the end cognitive node of the last run of the CFN or the initial input when the first run of the CFN. The initial input reflects the cognitive status of the team, which is

accumulated through learning from each other and from typical problem solving examples.

3. Generalised cognitive flow coming from the direct predecessor(s). The generalised cognitive input of the first team member will be the generalised cognitive output of the end cognitive node of the last run or the pre-designed initial input at the first run. The generalised cognitive flow can extend its problem solving region. It can also refine the cognitive flow to avoid unlimited expansion of the cognitive flow content.

Before accomplishing the current problem solving task, the output cognitive flow of the end cognitive node of a CFN should be saved in a team cognitive flow repository for reference while solving future problems. A team cognitive flow repository is a set of cognitive flow frames with the saved date. The team members can query the required cognitive information according to various versions: the problem, the member, the revision, and the date on which the problem was solved. The out-of-date content in the repository will be updated after the termination of the problem solving process. The team cognitive flow repository is located at the site where the team-ware could provide support.

With the team cognitive flow repository and the member cognitive flow repository, a development team can adapt to a change of its members because the related cognitive information of the members who leave are recorded in the team members' cognitive flow repository. A new team member can become experienced enough to play the role of the absent member by learning the cognitive information from the individual and team repository.

Three Internet-based communication approaches like the e-mail-based, blackboard-based, flow-based and hybrid approaches can be used to implement the cognitive flow [36]. Recently, we have proposed an Internet-based knowledge grid platform to realise global knowledge sharing [41].

5. A case study: cognitive Cooperation in distributed team software development

5.1. Case description and related work

Distributed team software development is a kind of paradigm that focuses on work co-operation and resource sharing between distributed team members. Such a development can make use of resources at different sites. The team should be supported by a distributed and across-platform computing environment. At the system level, the team environment should support version control, workspace and release management, build management, process management, etc.

The first generation environments are file-based version control tools (e.g. Sun Microsystems *TeamWare* and Microsoft *Visual SourceSafe*). The second-generation environments shift the focus from the file to the project level. They employed a project repository that supports parallel development, team co-ordination, and process management. Besides sharing codes and documents, it should further allow team members to share technical skills (establishment of IT skills standards has been proposed [22]), to learn from each other, etc.

Reasons for incorporating cognitive flow include:

(1) *The Software development process is cognitive*: Software development is a knowledge-intensive human activity process, and software development can be improved by recognising related knowledge structures [25]. Team members can improve their problem solving ability during the development process; they can do this by cognitive co-operation between the team members.
(2) *Understand and utilise dynamic cognitive Co-operation during development*: Team members' cognitive information (knowledge, experience, methods, strategy, and skills) about the software development is generated and accumulated during the development process; cognitive co-operation between them cannot be pre-designed.
(3) *A distributed team requires an effective and low cost communication method*: Ordered communication can reduce the communication cost and better reflect the real work process in project development.
(4) *A development team should be supported by a way to collect and distribute experience*: Each team member should be able to use the experience of other team members and the overall experience accumulated during previous project development efforts. With cognitive flow, the team members can avoid redundant work and adapt to any change of membership.

(5) *Different cultural backgrounds of the geographically distributed team members require high-level Cooperation*: Direct communication between team members with different cultural backgrounds will be more difficult but a high-level cognitive Cooperation mechanism can help cross this barrier.

To incorporate cognitive flow into the management of software development can meet these requirements. The cognitive flow content only includes a general description of the cognitive information that can be further refined and stored in a central repository (i.e. the team repository).

5.2. Solution description

An interactive platform for collaboration and the agent-based software engineering were discussed in [11,26]. Our intention was to provide an assistant mechanism that could work with any development environment to promote Cooperation between team members. Our solution is to provide the development team with a set of agents, each of which has an individual cognitive repository and serves one developer. Every developer performs the team's development task according to pre-defined workflow under time constraint. With the help of an agent, every developer can input cognitive information into the cognitive flow and get the relevant cognitive information from the cognitive flow during the development process. In simplification, we provide an automatic control mechanism to distribute the control right to the team members. The agent of every developer is mainly responsible for performing six tasks:

(1) Collect the developer's experience by recording the problems encountered and the solutions made during the development process, i.e. the agent will learn from its host during this process. The agent can only provide the developer with an active window interface to accept the input content of the flow frame.
(2) Summarise and generalise the collected experience (knowledge acquisition and mining).
(3) Communicate with other agents by using the inter-agent common language (e.g. KQML or CBL), e.g. assist its host to ask other team members' questions in their common language. This can help cross the culture barrier.
(4) Learn experience from other agents. This enables the agent to assist its host intelligently.
(5) Help to find the solution to the problem of the developer.
(6) Teach new members after their recruitment. This will help to retain the team's ability.
(7) Assist a developer to maintain his or her individual cognitive repository: generalising the contents, establishing links between similar contents, etc.

5.3. Cognitive flow in software development

In software development, the general cognitive flow can be classified as the following levels:

(1) *Concept level*: It concerns domain concepts, design concepts, and software concepts. Team members can share these concepts at this level. The semantic of a concept and the relationship between concepts are described in ontology.
(2) *Process level*: Team members can share some software processes they participate in.
(3) *Code level*: Here team members share their programming skills. The content of this level is the programming skills that result in a set of problem–solution pairs.
(4) *Component level*: This reflects the cognitive information about the components being developed by the team members; it can help team members use the component (or reuse its subcomponents). This level has two items. The first is the 'category' to which the component belongs. It has a category name that defines the component generalisation. The second item is the 'dependent components and categories,' which enables the team members who want to use the component to identify its dependent components. It is a set of component and category name pairs. The corresponding analogy link content points to the similar component name and related versions.
(5) *Method level*: This allows team members to reuse the problem solving method. The content of this level is a set of problem–method pairs, where the method can be either a set of general problem solving steps or algorithms. The corresponding analogy link points to similar problem–method pairs with related versions and optional methods.

(6) *Rule level*: This records the development and the pre-designed cognitive co-operation rules. With workflow execution, the development rules become richer. The team members can then share the development rules. Rules should be generalised for supporting software development. Co-operation rules define the Cooperation between team members. These rules are very useful for new team members.
(7) *Strategy and evaluation level*: This reflects the strategies made during the development process and the evaluation of their application. It provides a reference for the team members in planning their strategies. The content is a set of ⟨*situation*, *strategy*, *evaluation*⟩, where the *evaluation* is the degree of satisfaction of the strategy. Such an evaluation can help later team members avoid making unsuccessful choices when a similar problem is faced. The analogy link of this level points to the similar situation–strategy pairs with the related versions and the optional strategies to the same situation.

Reuse (or sharing) of cognitive information at different levels can be complete, partial [37], or a kind of heuristic that can help other team members find a short cut to accomplish their tasks.

5.4. Experimental comparison

The purpose of this experiment was to examine the relationship between cognitive Cooperation and problem solving ability in small teams. The experiment concerns an instructor and two small teams: one (A) works with the cognitive flow and workflow while another (B) only works with the workflow. Team A has the same number of members as B. The instructor was responsible for determining the match degree between the students' specialisation and the assigned task (a match can be obtained by selecting suitable students or projects), checking the quality of task accomplishment, and counting the duration of project development. The first step was to compare the problem solving abilities of the two teams. We used the time duration (D) to solve a set of problems completely to some satisfactory quality as a measure of the team's problem solving ability. The second step was to compare the change of the problem solving abilities of the two teams after solving the same set of problems.

Different members of a team will play different roles when solving different problems. The matching degree (denoted as MD, MD $\in [0,1]$) between every member's problem solving ability and the role required by the problem was an important factor that affected the team's problem solving ability. MD $= 0$ and 1 represented "complete mismatch" and "complete match", respectively. The larger MD means a better match between the team member and the role of the problem. The match degree of the team (MDT) is a function (we used the average) of all the matches of its members, such that MDT $\in [0,1]$.

The experiment was carried out by assigning two student-teams (A and B) with the same groups of projects (i.e. projects 1–4, projects 5–8, and projects 9–12), and then recording the time to complete each project. Each team had five student members, and every member was responsible for developing a component of the assigned project. The development duration of a project was the sum of the duration spent for task division, developing all the components, and component composition. The experimental data of the three groups of projects is shown in Table 1. The following results were obtained:

Table 1
Experiment for comparing two student-team's problem solving abilities

Stage	Project	MDT	D(A) (h)	D(B) (h)
1	1	0.25	8.5	8.3
	2	0. 5	6	6.5
	3	0.75	5	5
	4	1.0	4	4
	Average	0.625	5.875	5.95
2	5	0.25	7	7
	6	0.5	5.5	6
	7	0.75	5	5
	8	1.0	4	4
	Average	0.625	5.375	5.5
3	9	0.25	6	6.5
	10	0.5	5	5.5
	11	0.75	4.5	5
	12	1.0	3.5	4
	Average	0.625	4.75	5.25
4	Total average	0.625	5.33	5.57

(1) The larger MDT causes the shorter duration. The team member with the smaller MD needs more time to communicate with the other members than with the bigger MD.
(2) D(A) varying range is larger than that of D(B) with a change of MDT.
(3) D(B) is more stable than D(A) when developing the first group of projects.
(4) D(A) becomes more stable and D(B) stays almost unchanged when carrying out the second and the third group of projects. This shows that team A has learning ability.
(5) The total average data shows that D(A) is smaller than D(B).

The problem solving abilities of team A and B can be measured by $1/D$(A) and $1/D$(B), respectively, so the problem solving ability of team A is stronger than that of team B.

After finishing developing a new project, the knowledge addition of any member of team A consists of the knowledge: (1) *learned from the development process*; (2) *learned directly from the other team members*; (3) *generated from the knowledge reasoning* based on the heuristic information obtained by communicating with other members during the process. But the knowledge addition of any team member of team B after finishing the development of a new project only has the first part of the knowledge available to team A. If we give team B a period of time to allow its team members to learn from each other after finishing the project, team B can get the second part of the knowledge, but the third part of knowledge is generated during the development process, and it cannot be obtained afterwards.

Problem solving ability consists of problem solving knowledge (including basic knowledge, high-level skills, systematic understandings, and inspiration and creativity required by domain problem solving) and the mechanism of using that knowledge. A co-operating team's problem solving ability could be more than the simple sum of all the team members' individual problem solving abilities. A part of the team's problem solving ability is generated during the co-operative work process. A team member can learn from his or her own work process, learn existing knowledge from the other team members, and generate new knowledge and new approaches after obtain some heuristic information from other team members. This is why a well-organised co-operative team can work better and wiser than a disorganised team. An entire problem solving process should be managed at two levels: (1) *work*, including task planning, resource scheduling, and workflow management and (2) *cognitive*, including the learning of the relevant problem solving knowledge, skills, rules, methodologies, etc.

The scale of team members is another important factor that affects the team's problem solving ability. Larger problems require larger teams. The cognitive co-operation of a large team is more complex than that of a small team. We only focused on small teams with three to seven members. The experiment shows that the problem solving ability of the team with cognitive Cooperation was stronger than that without cognitive Cooperation.

The proposed approach provides a practicable way to manage the software development ability of the team during problem solving process. The case study shows that the approach has four advantages: (1) it helps to generate cognitive information during the development process; (2) it prevents the case that the developers forget the cognitive information generated during the process; (3) it prevents the case that the developers are not reluctant to contribute the cognitive information after finishing the development; (4) it can avoid the case that developers are not reluctant to spend extra time to recollect the cognitive information.

6. Conclusion

This paper presented an agent-based workflow model, proposed a flow-based cognitive co-operation approach for distributed team co-operation, and applied the approach to team software development. The main contribution involved three factors.

1. The cognitive flow model and management approach provides a new way to improve distributed team Cooperation. It can realise low-cost and ordered cognitive information sharing, accumulation, and inspiration during knowledge-intensive co-operative problem solving.
2. The approach provides a way to promote the current WfMS from work to cognitive level

Cooperation by incorporating agent mechanism and cognitive flow into the workflow model and upgrading the management mechanism.

3. The approach provides a way to retain a team's problem solving capability while changing team members.

An experiment in team software development shows that the combination between cognitive flow and workflow can increase a small-scale team's development ability. This also shows that new problem solving abilities could be generated during a Cooperation process. Incorporating the corresponding management and utilisation mechanism into current teamware would enhance the teams' problem solving capabilities. Currently, a main part of the cognitive flow support environment has been implemented [40,41], and is available for use at http://kg.ict.ac.cn.

Acknowledgements

The author thanks the Chairman of the Editor Board and the anonymous referees for their helpful comments on the earlier version of this paper. The research work was supported by the National Science Foundation, the National Basic Research Plan, and the National Hi-tech R&D Plan of China.

References

[1] R.I. Brafman, M. Tennenholtz, Modelling agents as qualitative decision makers, Artificial Intelligence 94, 1997, pp. 217–268.
[2] F.P. Brooks, The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley, Reading, MA, 1995.
[3] M. Cushman, L.A. Franco, J. Rosenhead, Learning from partners in the construction industry: a feedback approach to cross-organisational learning, in: Proceedings of the Eighth International Conference in Multi-Organisational Partnerships and Co-operative Strategy, Bristol, UK, 12–14 July 2001.
[4] P.F. Drucker (Ed.), Harvard Business Review on Knowledge Management, Harvard Business School Press, Boston, MA, 1998.
[5] A. Geppert, D. Tombros, K.R. Dittrich, Defining the semantics of reactive components in event-driven workflow execution with event histories, Information Systems 23 (3/4), 1998, pp. 235–252.
[6] B.R. Gaines, Knowledge management in societies of intelligent adaptive agents, Journal of Intelligent Information Systems 9 (3), 1977, pp. 277–298.
[7] A.K. Goel, Design, analogy and creativity, IEEE Expert 12 (3), 1997, pp. 62–70.
[8] B.J. Grosz, S. Kraus, Collaborative plans for complex group action, Artificial Intelligence 86, 1996, pp. 269–357.
[9] C.C. Hayes, Agents in a nutshell—a very brief introduction, IEEE Transactions on Knowledge and Data Engineering 11, 1999, pp. 127–132.
[10] W. Huang et al., Effects of group support system and task type on social influences in small groups, IEEE Transactions on System, Man and Cybernetics 27, 1997, pp. 578–587.
[11] N.R. Jennings, On agent-based software engineering, Artificial Intelligence 117, 2000, pp. 277–296.
[12] P.D. Karp, V.K. Chaudhri, S.M. Paley, A collaborative environment for authoring large knowledge bases, Journal of Intelligent Information Systems 13 (3), 1999, pp. 155–194.
[13] S.O. Kimbrough, S.A. Moore, On automated message processing in electronic commerce and work support systems: speech act theory and expressive felicity, ACM Transactions on Information Systems 5 (4), 1997, pp. 321–367.
[14] S. Kraus, Negotiation and Cooperation in multi-agent environments, Artificial Intelligence 94, 1997, pp. 79–97.
[15] P. Lawrence (Ed.), Workflow Handbook 1997, Wiley, New York, 1997.
[16] F. Leymann, D. Roller, Workflow-based applications, IBM Systems Journal 36 (1), 1997, pp. 102–122.
[17] V.R. Lesser, Co-operative multiagent systems: a personal view of the state of the art, IEEE Transactions on Knowledge and Data Engineering 11 (1), 1999, pp. 133–142.
[18] D.A. March, W.J. Kettinger, J.D. Rollins, Information orientation: people, technology and the bottom line, Sloan Management Review Summer (2000) 69–80.
[19] J. Mayfield, Y. Labrou, T. Finin, Evaluation of KQML as an agent communication language, http://www.cs.umbc.edu.
[20] P. Muth et al., From centralized workflow specification to distributed workflow execution, Journal of Intelligent Information Systems 10 (2), 1998, pp. 159–184.
[21] J. Puustjarvi, H. Tirri, J. Veijalainen, Reusability and modularity in transactional workflows, Information Systems 22 (2/3), 1997, pp. 101–120.
[22] R. Rada, IT skills standards, Communications of the ACM 42 (4), 1999, pp. 21–37.
[23] S. Ram, V. Ramesh, Collaborative conceptual schema design: a process model and prototype system, ACM Transactions on Information Systems 16 (4), 1998, pp. 347–371.
[24] M. Reichert, P. Dadam, Adept-flex—supporting dynamic changes of workflows without losing control, Journal of Intelligent Information Systems 10 (2), 1998, pp. 93–129.
[25] P.N. Robillard, The role of knowledge in software development, Communication of the ACM 42 (1), 1999, pp. 87–92.
[26] T.J. Rogers, R. Ross, V.S. Subrahmanian, IMPACT: a system for building agent applications, Journal of Intelligent Information Systems 14 (2/3), 2000, pp. 95–113.
[27] T. Rose, Visual assessment of engineering processes in virtual enterprises, Communications of the ACM 41 (12), 1998, pp. 45–52.

[28] B. Shneiderman, Creating creativity: user interfaces for supporting innovation, ACM Transactions on Computer–Human Interaction 7 (1), 2000, pp. 114–138.
[29] M. Tambe et al., Building agent teams using an explicit teamwork model and learning, Artificial Intelligence 110, 1999, pp. 215–239.
[30] M. Tambe, Towards flexible teamwork, Journal of Artificial Intelligence Research 7, 1997, pp. 83–124.
[31] WfMC, The workflow reference model, http://www.wfmc.org.
[32] WIDE Consortium, WIDE workflow development methodology, http://dis.sema.es/projects/WIDE/Documents/.
[33] H. Zhuge, J. Ma, X.Q. Shi, Analogy and abstract in cognitive space: a software process model, Information and Software Technology 39, 1997, pp. 463–468.
[34] H. Zhuge, Conflict decision training through multi-space co-operation, Decision Support Systems 29, 2000, pp. 111–123.
[35] H. Zhuge, T.Y. Cheung, H.K. Pung, A timed workflow process model, Journal of Systems and Software 55, 2001, pp. 232–243.
[36] H. Zhuge, X. Shi, Communication cost of cognitive co-operation for team development, Journal of Systems and Software 57, 2001, pp. 227–233.
[37] H. Zhuge, Inheritance rules for flexible model retrieval, Decision Support Systems 22 (4), 1998, pp. 379–390.
[38] H. Zhuge, X. Shi, A dynamic evaluation approach for virtual conflict decision training, IEEE Transactions on Systems Man and Cybernetics 30, 2000, pp. 374–380.
[39] H. Zhuge et al., A federation-agent workflow simulation framework for virtual organization development, Information and Management 39 (4), 2002, pp. 325–336.
[40] H. Zhuge, et al., KGCL: A knowledge-grid-based co-operative learning environment, Lecture Notes in Computer Science, in: Proceedings of the First International Conference on Web-Based Learning, Hong Kong, August 2002, Springer, Berlin.
[41] H. Zhuge, A knowledge grid model and platform for global knowledge sharing, Expert Systems with Applications 22 (4), 2002, pp. 313–320.

Hai Zhuge is a professor at the Institute of Computing Technology, Chinese Academy of Sciences. His current research interests include: semantic grid, knowledge flow management, problem-oriented model base systems, component reuse, cognitive-based software process model, inter-operation model for group decision, and web-based workflow model. He is now the leader of the China Knowledge Grid project Vega-KG. He is the author of one book and over 40 papers appeared mainly in leading international conferences and the following international journals: *IEEE Transactions on Systems, Man, and Cybernetics*; *Information and Management*; *Decision Support Systems*; *Journal of Systems and Software*; *International Journal of Co-operative Information Systems*; *Expert Systems with Applications*, *Knowledge-based Systems*; *Information and Software Technology*; *Journal of Software*; and *Lecture Notes in Computer Science*.