PERGAMON

Expert Systems with Applications 23 (2002) 23–30

Expert Systems with Applications

www.elsevier.com/locate/eswa

Short communication

A knowledge flow model for peer-to-peer team knowledge sharing and management

Hai Zhuge*

Knowledge Grid Group, Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, People's Republic of China

Abstract

To realize effective knowledge sharing in teamwork, this paper proposes a knowledge flow model for peer-to-peer knowledge sharing and management in cooperative teams. The model consists of the concepts, rules and methods about the knowledge flow, the knowledge flow process model, and the knowledge flow engine. A reference model for coordinating the knowledge flow process with the workflow process is suggested to provide an integrated approach to model teamwork process. We also discuss the peer-to-peer knowledge-sharing paradigm in large-scale teams and propose the approach for constructing a knowledge flow network from the corresponding workflow. The proposed model provides a new way to model and manage teamwork processes. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Knowledge grid; Knowledge management; Semantic web; Teamwork; Workflow

1. Introduction

Knowledge has become the most precious property of any commercial or academic institution. Knowledge management plays the key role in upgrading the competitiveness of a team. Knowledge management concerns innovating, spreading, sharing, and using of knowledge. Research on knowledge management concerns the management aspect including organizational learning, personal management, cultural, etc. (Drucker et al., 1998), and the technical aspect including models, support tools and environments (Zhuge, 2002). This paper focuses on the technical aspect and on the application background of the Internet-based teamwork.

Teamwork process can be regarded as a co-operative human problem-solving process with the support environment. Such a co-operation exists at both the knowledge level and the work level. At the knowledge level, team members can learn knowledge from each other and can further make abstractions and analogies between problems, and use past experiences and skills to solve new problems (Zhuge, Ma, & Shi, 1997). Learning from human problem-solving processes is an important strategy in developing teamwork support environments.

The Internet enables a team to be globally distributed. The globalization of applications causes not only the rising of the communication cost between the distributed team members but also the increasing of changing partners (Zhuge & Shi, 2001). The member change of a team leads to two effects: (1) knowledge drain happens with the capable members leaving their posts, and (2) member recruitment causes different experienced members to work together. These new challenges require a distributed teamwork environment to support team knowledge management.

People have investigated multiple types of flows (e.g. the material flow, the energy flow, the message flow, control flow, etc.) and the rules they follow in respective domains. Unfortunately, a knowledge flow existed in teamwork processes has not been paid enough attention. This knowledge flow reflects the knowledge level cooperation in teamwork, which has an important influence on the effectiveness of teamwork. This paper investigates: the principle and mechanism of the knowledge flow that can carry, share and accumulate knowledge when it goes through from one team member to another; the rules it follows; the related management mechanism for realizing the ordered knowledge sharing in a distributed team; and, the approach for coordinating the knowledge flow process with the workflow process so as to provides a better way to model the distributed teamwork process.

2. Related works: the semantic web, the workflow, and the knowledge grid

The current Web enables information exchange between

* Fax: +86-1062567724.
E-mail address: zhuge@ict.ac.cn (H. Zhuge).

PII: S0957-4174(02)00024-6

distributed users. But the current HTML-based web pages cannot reflect machine-understandable semantics. The main intent of the semantic web is to make the Web resources machine-understandable. The Semantic Web currently focuses on the markup languages such as Resource Description Framework (RDF), Ontology Inference Layer (OIL), and DARPA Agent Markup Language (DAML) (Fensel et al., 2001; Heflin & Hendler, 2001; Hendler, 2001; Klein, 2001; Maedche & Staab, 2001). The XML-based RDF defines the machine-understandable semantics of web resources by using the object-attribute-value model (Klein, 2001). The RDF schema (RDFS) enhances the representation ability of the RDF through providing the means to define the vocabulary, the class-based structure and the constraints for expressing the metadata about the Web resources. OIL is an extension of the RDFS through the well-defined syntax in XML based on the document type definition (Fensel et al., 2001). DAML is an ongoing project that aims at enabling the markup and manipulation of complex taxonomic and logical relationships between objects on the Web (Hendler, 2001). The Semantic Web provides the across platform media basis to exchange knowledge between distributed team members.

Workflow is a technique that can be used to realize work co-operation between team members according to a definite logical process. A workflow management system (WfMS) is a system that completely defines, manages and executes the workflow specification through the execution of software whose execution order is driven by a computer representation of the workflow logic (Lawrence, 1997; Leymann & Roller, 1997; WfMC). As a high-level tool, workflow can be used to integrate distributed and heterogeneous application processes into a unified process with the support of low-level distributed object techniques. Workflow can establish the logical dependence order relationship between team members' works (activities). The characteristics of time modeling, reusability, exceptional handling, agent-based workflow, and formal model of workflow have been discussed (Geppert, Tombros, & Dittrich, 1998; Zhuge, Cheung, & Pung, 2001; Zhuge et al., 2002). These characteristics can be incorporated into the WfMS so as to support a better teamwork. But the current workflow model and WfMS does not support knowledge level cooperation. In fact, the implementation of each task of a workflow is an inter-operation between human team members and the support environment. Team members' knowledge and cognitive ability play an important role in teamwork.

The purpose of the Knowledge Grid is for sharing and managing globally distributed knowledge resources in an efficient and effective way (Zhuge, 2002). A Knowledge Grid engine is proposed to realize knowledge operation and intelligent use of knowledge. The Internet users can use the Knowledge Grid Operation Language KGOL built in the engine to create their Knowledge Grids, to put knowledge to them, to edit knowledge, to partially or wholly open their Grids to any other Grids, and to get the required knowledge from either a special Knowledge Grid or the worldwide Knowledge Grids distributed on the Internet as a universal resource view. A three-dimensional knowledge space is proposed to uniformly specify knowledge. A Knowledge Grid platform has been implemented and is available for use (http://kg.ict.ac.cn). A Knowledge Grid enables people to conveniently share knowledge with each other when they work on the Internet. A Knowledge Grid can be established on either the Semantic Web or the high-performance Grid computing environment (http://www.gridforum.org).

3. Knowledge flow

3.1. Knowledge flow and its field

Knowledge flow is invisible, but it works with any co-operative team no matter whether people intentionally make use of it or not. We can imagine the following scenario of the knowledge flow working with a cooperative team. Team members are linked with various types of 'knowledge transmission belts' like the production line. Any team member can put knowledge onto a proper belt, which then automatically conveys the knowledge to the team member who requires it. Any team member can get the required knowledge from the 'transmission belt' linked to him when performing his task. These transmission belts together with the team members constitute a knowledge flow network (KFN). People can raise the effectiveness of teamwork by properly designing the network and controlling its execution process.

One advantage of the knowledge flow is that it can avoid unnecessary knowledge passing between team members (Zhuge et al., 2002), because different team members may perform different types of tasks and require different types of knowledge. Another advantage is that people do not need to spend time and make effort (the required knowledge is not easy to be accurately acquired in large-scale knowledge repository) in searching for the required knowledge from a centralized knowledge repository during task performing process as most traditional knowledge base system did.

Definition 1. A *knowledge flow* (denoted as KF) is a process of knowledge passing between people or knowledge processing mechanism. It has three crucial attributes: *direction*, *content*, and *carrier*, which, respectively, determine the sender and the receiver, the sharable knowledge content, and the media that can pass the content.

We use an arrow to denote the direction of a knowledge flow. The carrier can be based on the Internet or a local network. The sharable knowledge content implies that the knowledge is understandable by all team members. Our strategy is to make use of the Internet and the Semantic Web as the knowledge carrier. Knowledge content can be specified by a knowledge space (Zhuge, Chen, Feng, & Shi, 2002), where each point determines knowledge of a certain

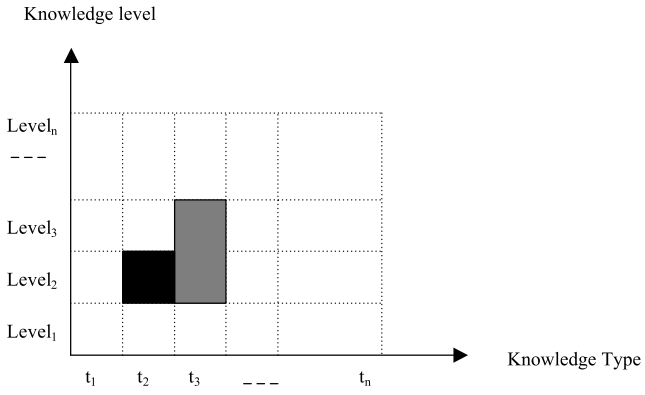

Fig. 1. The fields of knowledge flow defined in knowledge space.

type, at a certain knowledge level, and at a certain location. Such a knowledge specification meets the following needs: (1) people working at different positions require knowledge of different levels, and (2) people working for different types of tasks require different types of knowledge.

Definition 2. The field of a knowledge flow KF is a two-dimensional region in a knowledge space, defined by a type-field (TF) and a level-field (LF), denoted as Field(KF) = ⟨TF, LF⟩, where TF = ⟨t|t is a knowledge type⟩ and LF = ⟨level|level is a knowledge level⟩.

Two rectangles in Fig. 1 represent two knowledge fields. The union of two type-fields $TF_1 \cup TF_2$ is a set union of them such that the order of the knowledge types of each at the knowledge type axis can be kept. Similarly, the following set operations: ' ∪ ', ' ∩ ', ' − ' can be carried out between any two TFs and between any two LFs.

Characteristic 1. Let $\langle TF_1, LF_1\rangle$ and $\langle TF_2, LF_2\rangle$ be the fields of two knowledge flows KF_1 and KF_2, we have the following items hold:

1. $\langle TF_1, LF_1\rangle \cup \langle TF_2, LF_2\rangle = \langle TF_1 \cup TF_2, LF_1 \cup LF_2\rangle$;
2. $\langle TF_1, LF_1\rangle \cap \langle TF_2, LF_2\rangle = \langle TF_1 \cap TF_2, LF_1 \cap LF_2\rangle$;
3. $\langle TF_1, LF_1\rangle - \langle TF_2, LF_2\rangle = \langle TF_1 - TF_2, LF_1 - LF_2\rangle$; and,
4. $\langle TF_1, LF_1\rangle \subseteq \langle TF_2, LF_2\rangle$ if and only if $\langle TF_1 \subseteq TF_2, LF_1 \subseteq LF_2\rangle$.

3.2. Knowledge flow network and its characteristics

Knowledge node (KN) is the stop (the sender or receiver) of a knowledge flow. It corresponds a team member (or a role) and reflects the knowledge generation and requirement during the team member's task implementation process. The output of a KN is a knowledge flow that depends on the corresponding team member's cognitive ability and the input knowledge flow. A KN can be implemented as a mechanism that incorporates a personal knowledge repository and an agent for helping team members to process knowledge. We call a KN *active* only when the corresponding team member works on it. Otherwise, the KN is *inactive*.

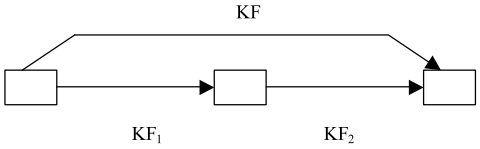

Fig. 2. An example of repeat knowledge flow paths.

An inactive KN can be *re-active* again when the corresponding team member works on it again.

Definition 3. A knowledge flow passing through team members during a teamwork process constitutes a KFN. It consists of a set of KNs that reflects the participation of team members and the knowledge flows between them.

The following definitions are for answering to the question of what is a good KFN.

Definition 4. A KFN is called connected if there exists a flow path between any two KNs.

Definition 5. A KFN is called complete on a task if it is connected and such that its KN set is the mapping image of all the members or their roles for performing the task.

A complete KFN can eliminate knowledge isolation between team members.

Definition 6. For all the complete KFNs of a team for performing a task, a KFN is called the smallest complete if it has the least number of knowledge flows between KNs.

A smallest complete KFN can not only eliminate knowledge isolation but also achieve an effective team knowledge sharing, so it is a criterion for design a KFN.

Characteristic 2. A smallest complete KFN does not include any repeat knowledge flow paths between any two KNs.

Fig. 2 shows an example of the repeat knowledge paths.

3.3. Knowledge flow representation

The representation of knowledge flow should have the following five features. (1) *Information accumulation*, it should be able to accumulate knowledge during the current task performing period and can keep knowledge for later use. (2) *Classification*, it should be able to classify knowledge according to different projects and different team members. (3) *Abstraction*, it should be able to reflect knowledge at different abstraction levels and to refine the content. (4) *Analogy*, it should be able to establish analogy associations between the related contents. (5) *Version management*, it can manage the evolution process of the knowledge flow. The knowledge space presented in (Zhuge, 2002) meets the needs of the earlier requirement. Considering the across platform requirement of the knowledge flow, we adopt the Extensible Markup Language (XML) to represent the knowledge flow as follows.

⟨KnowledgeFlow⟩
  ⟨Sender⟩SenderName⟨/Sender⟩
  ⟨Receiver⟩ReceiverName⟨/Receiver⟩

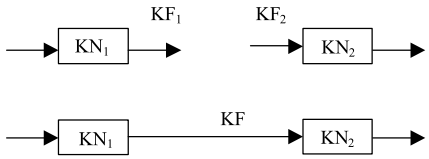

Fig. 3. Sequential Connection between two knowledge flows.

⟨Content⟩⟨Content$_1$⟩KnowledgeDescription⟨/Content$_1$⟩
…
⟨Content$_n$⟩KnowledgeDescription⟨/Content$_n$⟩
⟨/Content⟩
⟨/KnowledgeFlow⟩

In the earlier representation, the sender and the receiver define the direction of the knowledge flow, the content portion specifies knowledge content where the similar content can be specified for analogous purpose and the field of the knowledge flow in a knowledge space can be specified. An example for describing the method-level knowledge is given later.

⟨Content$_i$⟩ ⟨Problem⟩ProblemDescription⟨/Problem⟩
⟨Solution⟩SolutionDescription⟨/Solution⟩
⟨Field⟩⟨LF⟩Method⟨/LF⟩
⟨TF⟩Coding ⟨/TF⟩
⟨/Field⟩
⟨/Content$_i$⟩

4. Knowledge flow process model

A knowledge flow process concerns four types of connections between knowledge flows: the sequential connection, the join connection, the split connection and the broadcast connection.

Definition 7. Sequential Connection of two knowledge flows (KF_1 and KF_2) forms one knowledge flow (denoted as $KF_1 \cdot KF_2$) such that the following two items hold:

1. $Field(KF_1 \cdot KF_2) = Field(KF_1) = Field(KF_2)$ if $Field(KF_1) = Field(KF_2)$; and,
2. $Field(KF_1 \cdot KF_2) = Field(KF_2)$ if $Field(KF_1) \subseteq Field(KF_2)$.

The sequential connection of two knowledge flows causes the output flow of one KN connects with the input flow of another KN as shown in Fig. 3.

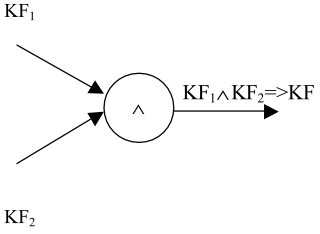

Fig. 4. Join-connection between two knowledge flows.

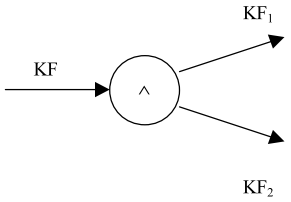

Fig. 5. The split of a knowledge flow.

Definition 8. Join-connection of two or more knowledge flows forms one knowledge flow, denoted as $KF_1 \wedge KF_2 \wedge \ldots \wedge KF_n = > KF$, such that $Field(KF_1 \wedge KF_2 \wedge \ldots \wedge KF_n = > KF) = Field(KF_1) \cup Field(KF_2) \cup \ldots \cup Field(KF_n) = \langle LR_1 \cup LR_2 \cup \ldots \cup LR_n, TR_1 \cup TR_2 \cup \ldots \cup TR_n \rangle$ holds.

Fig. 4 shows the join-connection of two knowledge flows.

Definition 9. A knowledge flow KF can be split into two or more knowledge flows, denoted as $KF = > KF_1 \vee KF_2 \vee \ldots \vee KF_n$, such that $Field(KF = > KF_1 \vee KF_2 \vee \ldots \vee KF_n) = Field(KF_1) \cup Field(KF_2) \cup \ldots \cup Field(KF_n) = \langle LF_1 \cup LF_2 \cup \ldots \cup LF_n, TF_1 \cup TF_2 \cup \ldots \cup TF_n \rangle$ holds.

Fig. 5 shows that a knowledge flow is split into two knowledge flows.

Definition 10. A knowledge flow KF can be broadcast to multiple knowledge flows, denoted as $KF = (KF_1, KF_2, \ldots, KF_n)$, such that $Field(KF = (KF_1, KF_2, \ldots, KF_n)) = Field(KF_1) = Field(KF_2) = \ldots = Field(KF_n)$ holds.

Fig. 6 shows that a knowledge flow is broadcasted to two knowledge flows.

A knowledge flow can accumulate the knowledge generated by the previous KNs during its passing process. Fig. 7 describes the constitution of the input and output knowledge flows of a KN (KN_i). The input knowledge flow is the join connection of the output knowledge flows of its predecessors: $KF_{out}(KN_{p1}) \wedge \ldots \wedge KF_{out}(KN_{pn}) = > KF_{in}(KN_i)$. The final output of the knowledge node KN_i, $KF_{out}(KN_i)'$ is constituted by the join connection of the input $KF_{in}(KN_i)$ and the output $KF_{out}(KN_i)$, represented as $KF_{out}(KN_i) \wedge KF_{in}(KN_i) = > KF_{out}(KN_i)'$.

For a newly created KFN, the knowledge input of the first KN, $KF_{in}(KN_0)$ will be initialised by the co-operation rules of the team. After the first run of the KFN, the output of the end KN of the last run will be the input of the first KN of the current run. After finishing to perform a task, the generated knowledge can be stored in a knowledge space for later use (Zhuge, 2002).

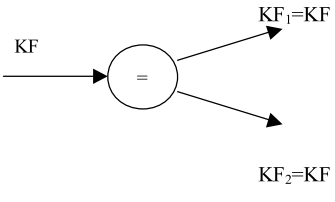

Fig. 6. Broadcast of a knowledge flow.

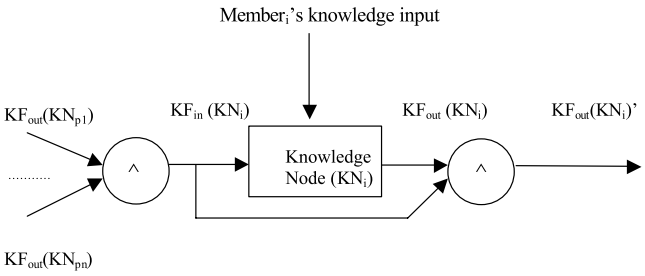

Fig. 7. Input and output of a KN.

There are two major differences between the knowledge flow and the workflow. First, the knowledge flow content is generated from the team members' task implementation process during the execution of the workflow process and cannot be pre-designed. While, a workflow reflects the domain business and is pre-designed by its designer. Second, the knowledge flow carries knowledge of the team members, while a workflow reflects either the data dependence relationship or the execution dependence relationship between activities (tasks). Despite the two differences, a knowledge flow process can be integrated with a workflow process to form a uniform teamwork process.

5. A reference model for integrating knowledge flow with workflow

Team members are the determinant factor of both the knowledge flow and the workflow. Knowledge flow and workflow can be integrated through the people or their roles participated in the two processes. A team member can participate in the work of one or more activities (task) nodes so can contribute knowledge to one or more KNs through roles.

A reference model for integrating knowledge flow and workflow consists of three levels as shown in Fig. 8: the knowledge flow level, the team member (or role) level, and the workflow level. The role model reflects the organization architecture of the team. The following three mappings establish the relationships between the knowledge flow level, the role level, the workflow level, and the human team members.

1. Mapping-1 deploys the roles onto the KNs.
2. Mapping-2 deploys the roles onto the workflow activities.
3. Mapping-3 deploys the roles onto team members.

6. Knowledge flow engine

Knowledge flow engine is to execute, control, schedule, and monitor the knowledge flow process during a teamwork process according to the knowledge flow process definition. Its components and function is similar to that of the workflow engine described in (http://www.wfmc.org). To avoid content redundancy, we herein just present different points between them as follows.

Problem-list. The knowledge flow engine should provide a mechanism that enables a team member to list the problems encountered during working on a task. The engine first checks the personal knowledge repository to find the solutions to these problems, if not available, then announcing its peer(s) to contribute their knowledge for solving the problems.

Ontology Service. The ontology of a particular domain establishes a common understanding between people. It usually contains a hierarchy of domain concepts and describes each concept's crucial properties through an attribute-value. The WordNet is a kind of ontology at the conceptual level. People have developed the assistant tools for the creation and management of ontology. The function of the ontology service is to explain knowledge when misunderstanding happens during knowledge flow passing and sharing process.

Implement the Connection between Knowledge Flow. The knowledge flow engine should have the mechanism for implementing the connection operations between knowledge flows.

Version Management of Knowledge. The mechanism for

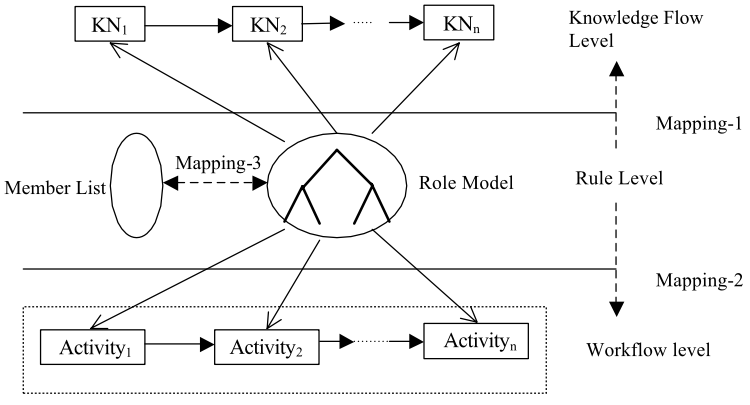

Fig. 8. A reference model for integrating knowledge flow and workflow.

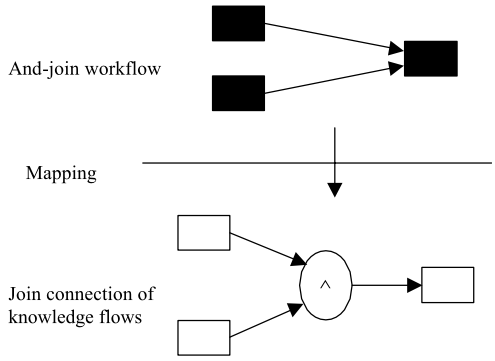

Fig. 9. Example of mapping from workflow into knowledge flow.

avoiding knowledge loss in case of flow passing exception and for deleting out-of-date knowledge.

Knowledge Generalisation. A KN is responsible for not only the generation of knowledge but also the generalisation of knowledge. Each team member's knowledge generalisation depends on three kinds of input: (1) the knowledge flow generated during performing the current work; (2) the knowledge flow output of the direct predecessor(s); and, (3) the generalised knowledge flow of the direct predecessor. The generalised knowledge input of the first team member will be the generalised knowledge output of the end KN of the last execution of the flow or the pre-designed initial input when it first executes. The generalised knowledge flow can extend its problem-solving region. It can also refine a knowledge flow so as to avoid unlimited expansion of the knowledge flow content.

7. Peer-to-peer knowledge sharing in large-scale organization

A large-scale organization usually has a hierarchical structure with multiple middle-layers. The team members are called peers if they work at the same level of the organization hierarchy and on the same type of tasks. Knowledge sharing means by making use of the knowledge existed in a team for the purpose of quickly solving problems. Knowledge sharing between peers is much more effective than that between non-peers. This is because of the following three reasons.

1. Peers work on the same types of tasks so their experiences are more valuable to share with each other for solving their respective problems.
2. Peers have similar knowledge structures so can understand with each other easily when sharing knowledge.
3. More common interests exist between peers so can effectively share knowledge.

For example, two programmers can better share programming knowledge with each other than a manager and a programmer do. According to the above reasons, we have the following peer-to-peer knowledge sharing principle.

Principle. A peer-to-peer knowledge sharing requires the receiver of a knowledge flow to be the peer of the sender.

Organization innovation is one of the key issues of knowledge management. Different from traditional large-scale organization structures, a successful information-based large-scale organization tends to have fewer middle layers. In some domains, organizations even tend to have no middle layer like orchestras (Drucker et al., 1998). So peer-to-peer knowledge sharing is also a criterion of structuring a large-scale organization.

The workflow process of an organization reflects its structure. A large-scale hierarchical organization needs a hierarchical workflow process to describe its behavior. Hence, we can get the KFN of a hierarchical organization by mapping the workflow process into the knowledge flow process. The following are five mapping rules.

Mapping rule1. Mapping the activity nodes of the workflow into the KNs of the KFN.
Mapping rule2. Mapping the edges (control flow) of the workflow into the knowledge flows of the KFN.
Mapping rule3. Mapping the 'and-join' and the 'or-join' connection between the activities of the workflow into the 'join' connection between knowledge flows.
Mapping rule4. Mapping the 'and-split' and the 'or-split' connection between the activities of the workflow into the 'split' connection between knowledge flows.
Mapping rule5. Link the output knowledge flow of the end KN with the input knowledge flow of the initial KN.

Fig. 9 shows an example of mapping from the and-join workflow connection into the join connection between knowledge flows. We show an example of mapping from a hierarchical workflow into a hierarchical knowledge flow in Fig. 10.

8. A brief case study

Distributed team software development is a software development management paradigm that focuses on work co-operation and resource sharing between distributed team members during development process. The current team development research works and environments only focus on technique aspects. Human cognitive characteristics are seldom addressed. Reasons for incorporating the knowledge flow into the distributed team development process include the following four aspects: (1) *Software development process is a knowledge-intensive process*. The team members can improve their work not only with the support of the software tools but also through cognitive co-operation among team members. (2) *Cognitive co-operation cannot be pre-designed*. The team members' knowledge (experience,

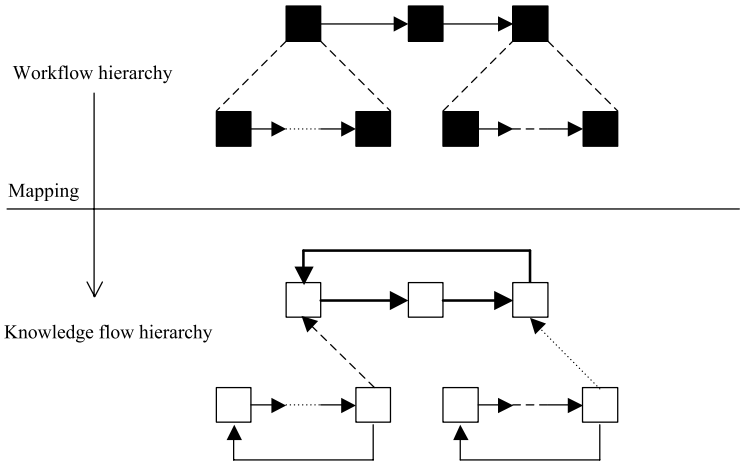

Fig. 10. Mapping from a hierarchical workflow into a hierarchical knowledge flow.

method, decision, and skill) about a software development is generated and accumulated during the development process, and the cognitive co-operation among them cannot be pre-designed. So a knowledge flow is needed to dynamically reflect the cognitive co-operation process. (3) *A distributed team requires an effective and low cost communication*. An ordered communication can reduce the communication cost and can better reflect the real work process of a project development. (4) *A development team should be supported by an experience accumulation mechanism*. Each team member can use the experience of its predecessor and the experience of the team accumulated during developing previous projects, so the team can be able to avoid redundant work and adapt to the change of team members.

The knowledge flow can satisfy the above requirements. Five knowledge levels can be classified from low to high according to people's cognitive characteristic.

1. *Code level knowledge*. This is the lowest abstraction level that helps the team members to share programming skills with each other during development process. The content of this level is the programming skills described as a set of problem–solution pairs.
2. *Component level knowledge*. This level reflects knowledge about the components being developed by the corresponding team members, it can help team members to reuse components.
3. *Method level knowledge*. This level enables the related team members to reuse the problem-solving method for solving their problems. The content of this level is described as problem-method pairs, where a method can be the process, the pattern, or the algorithm for solving a problem.
4. *Rule level knowledge*. This level records the development rules generated during the development process and the pre-designed knowledge co-operation rules. With the workflow execution, the development rules will become richer and richer. The development rules enable the succeeding team members to share the development rules. Rules should be then generalised for supporting a general software development. Co-operation rules determine the co-operation among team members. These rules are very useful for the newly joined team members to know how to co-operate with the other team members. Rules at the rule level can take the condition-action-result form as follows: RuleId: ON ⟨*Condition*⟩DO ⟨*Action*⟩RESULT⟨*ResultRecord*⟩ and ⟨*SuccessRate*⟩, where ⟨*Condition*⟩ can be logic 'and' or logic 'or' of several conditions, ⟨*Action*⟩ can be sequential *n* actions, ⟨*ResultRecord*⟩ records the *success* or *failure* of applying the rule, and ⟨*SuccessRate*⟩ = successful-application-times/total-application-times.
5. *Decision and evaluation level knowledge*. This level reflects the decisions made during the development process and the evaluation of these decisions. It provides the reference for the succeeding team members to make their decisions. The content is a set of triples: ⟨situation, decision, evaluation⟩. The evaluation reflects a kind of satisfaction degree about the decision. Such an evaluation can help the succeeding team members to avoid unsuccessful decisions when a similar situation is facing.

Knowledge sharing at different knowledge levels can be a complete reuse, a partial reuse, or just a kind of heuristic information, which can help the other team members to accomplish their development tasks.

9. Discussion

The Internet enables a team to be globally distributed. If team members do not communicate with each other, a team member cannot avoid to work on the problem that the related members have solved before. The Internet-based communication is the basis of the knowledge level co-operation between distributed team members. The Internet-based communication approaches can be classified as three types: the email-based, the blackboard-based, and

the flow-based (peer-to-peer). The flow-based approach has the lowest communication costs (Zhuge & Shi, 2001). This is because of the following three reasons. First, a team member is only allowed to communicate with whom have direct work dependence relationship with him/her. Any team member can know the related predecessors' knowledge from the input knowledge flow, so unnecessary communication and frequent information exchange can be avoided. Second, the knowledge flow executes in the order coinciding with the corresponding workflow process logic, so information confusion can be avoided. Third, the general and historical knowledge is also available from the knowledge flow, a newly joined team member can enrich his/her knowledge through learning from the knowledge flow, so the team can adapt to the change of team members. Besides, it is not an overburden for most team members to summarise and input their knowledge (e.g. problem–solution pairs) within a reasonable short period during his/her work period. Because the time duration for knowledge input can be divided into two parts: one is the generation duration, another is the in-key duration. The generation of a team member's knowledge naturally carries out with his/her thinking process during work period, it does not take any extra time. To avoid taking extra time for recollection and loss information, each team member should in-key the newly generated knowledge as soon as possible. Besides, teamwork mainly aims at big projects, which usually need to take quite a long work period, e.g. several months. While, the knowledge input usually takes a short period comparing with the whole period.

10. Conclusion

The proposed model has three major advantages. First, it realizes knowledge level co-operation between distributed team members. Such a co-operation can be optimized by properly designing the KFN. This advantage results in the enhancement of the team's problem-solving ability and the teamwork effectiveness. Second, the combination of the knowledge flow and the workflow can better model the teamwork process than the single workflow-level modeling approach. Third, it enables a distributed team to be able to loosely depend on the ability of its members. Usually, the efficiency and quality of teamwork depend on the ability of the team, which further depends on the ability of every team member and the knowledge-level cooperation between team members. The proposed model enables a team to stabilize its knowledge level in case of changing team members.

The proposed approach can be applied to any distributed human-computer process where the involved behavior need the co-operation between human and the support work environment, especially in the globally distributed teamwork applications where the team members cannot communicate with each other face to face and member recruitment often occurs.

Acknowledgements

This work was supported by the National Science Foundation, the National Basic Research Plan, and the National Hi-tech R&D Plan of China.

References

Drucker, P. F. (ed.) (1998). *Harvard business review on knowledge management*, Boston, MA: Harvard Business School Press.

Fensel, D., et al. (2001). OIL: an ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, *16* (2), 38–45.

Geppert, A., Tombros, D., & Dittrich, K. R. (1998). Defining the semantics of reactive components in event-driven workflow execution with event histories. *Information Systems*, *23* (3/4), 235–252.

Heflin, J., & Hendler, J. (2001). *A portrait of the semantic web in action*, *16* (2), 54–59.

Hendler, J. (2001). Agents and the semantic web. *IEEE Intelligent Systems*, *16* (2), 30–37.

Klein, M. (2001). XML, RDF, and relatives. *IEEE Internet Computing*, *March/April*, 26–28.

Lawrence, P. (1997). *Workflow handbook*, New York: Wiley.

Leymann, F., & Roller, D. (1997). Workflow-based applications. *IBM Systems Journal*, *36* (1), 102–122.

Maedche, A., & Staab, S. (2001). A ontology learning for the semantic web. *IEEE Intelligent Systems*, 72–79.

WfMC, The workflow reference model, http://www.wfmc.org

Zhuge, H. (2002). A knowledge grid model and platform for global knowledge sharing. *Expert Systems with Applications*, *22* (4).

Zhuge, H., & Shi, X. (2001). Communication cost of cognitive co-operation for team development. *Journal of Systems and Software*, *57* (3), 227–233.

Zhuge, H., Ma, J., & Shi, X. (1997). Analogy and abstract in cognitive space: a software process model. *Information and Software Technology*, *39*, 463–468.

Zhuge, H., Cheung, T. Y., & Pung, H. K. (2001). A timed workflow process model. *Journal of Systems and Software*, *57* (3), 231–243.

Zhuge, H., Chen, J., Feng, Y., & Shi, X. (2002). An agent-workflow-federation approach for virtual organization development. *Information and Management*, *39* (4), 325–336.

Hai Zhuge is a professor at the Institute of Computing Technology, Chinese Academy of Sciences. He was an associate professor at the Institute of Software, Chinese Academy of Sciences. He received the PhD in computer science from Zhejiang University, China, in 1992. He visited the National University of Singapore, the City University of Hong Kong, and the Tsinghua University several times as a senior visiting scholar or research fellow from 1994 to 2000. His current research interests include: knowledge management, grid and the semantic web, 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 principle investigator of the China Knowledge Grid project (Vega-KG) as well as three national grants. 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 Cooperative Information Systems; Expert Systems with Applications, Knowledge-based Systems; Information and Software Technology; and, Journal of Software.