

# A Novel Knowledge Modeling Framework *POTMe* and Its Application for Collaborative Problem Solving

Jian JIANG

Department of Mechanical Engineering, Nanjing University of Science and Technology,

Nanjing, 210094, China

Email: 83905118@163.com

**Keywords:** Knowledge Modeling Framework, Collaborative Problem Solving, Knowledge-dependency, Temporal Reasoning

**Abstract.** Collaborative efforts are inevitably demanded when solving a complex problem, which is often plagued with incomplete, ill-structured, and poor quality information. Therefore, a global knowledge modeling framework is required for cross-domain knowledge coordination. However, current knowledge modeling techniques pay little attention to temporal knowledge that is indispensable for navigating cross-domain collaborative task executions and their collaboration. In view of this challenge, a novel knowledge modeling framework, named *POTMe*, is investigated in this paper. This knowledge modeling framework provides a structured application context by articulating knowledge dependencies and temporal dependencies among problem-solving activities. The main contribution of this paper is twofold. First, this novel knowledge modeling framework provides a structured problem solving context. It enhances the consistency of global knowledge coordination among collaborative problem-solving activities. Second, temporal knowledge modeling is incorporated into this knowledge modeling framework. It is reified by a temporal reasoning rule, and aims at evaluating cross-domain task collaboration in an incorporated executive environment.

## Introduction

Complex problem solving is a cognitively difficult task and often requires collaborative efforts [1][2][3][4][5]. New product development [2], distributed software engineering [3][4], and disease diagnosis and remedy [5] are typical examples of complex problem solving that require collaborative endeavors. The related problem solving processes often consist of human-engineered activities [3][6]. Robillard [3] believed that a human-engineered activity could be characterized by three traditional engineering activities, i.e., 1) stakeholders define their needs and requirements, 2) engineers design the product based on stakeholders' definitions, and 3) producers build the product based on engineers' designs. These activities are often accompanied by cross-domain knowledge sharing and collaboration. The first and second activities could be characterized by product conceiving behaviors. The third activity could be featured by product producing behavior based on the product blueprint.

In collaborative problem solving, qualitative problem-defining process equal to a group of product conceiving behaviors, while quantitative parameter computing and verifying process could be treated as a group of product producing behaviors. Generally, they are two different application stages. The first stage mainly concentrates on preliminary knowledge discovery with qualitative problem cognition [3][5][7][8]. The second stage mainly focuses on verifying the problem specification derived from the first stage, through concrete parameter computing activities [6][9][10][11]. In practice, a complex problem and its collaboration solving are often initiated by incomplete, ill structured and poor quality information. It is always a challenging endeavor

to coordinate task executions and their collaboration, if there is no a knowledge modeling framework for navigating collaborations of task executions [1][2][6][12]. A knowledge modeling framework always aims at reducing conflicts and producing better solutions for collaboration problem solving, through articulating the knowledge-dependency relations among task executions and their collaboration[13][14]. It plays as a global mental model for knowledge sharing by all the teammates engaged in collaborative problem solving, through which numerous human thoughts and actions could be coordinated in a compatible way [3][15].

In the past few years, some knowledge modeling frameworks, such as Generic Tasks Structures [16], Role-Limiting Methods [17], CommonKADS [18], MIKE [19], Components of Expertise [20], have been proposed in the knowledge management community. Generally, current knowledge modeling techniques [16][17][18][19][20] pay little attention to temporal knowledge. In practice, collaborative problem solving parallels a group of knowledge-intensive task executions. To satisfy an expected deadline of a task execution or collaboration, temporal knowledge should be incorporated into a knowledge modeling framework for scheduling cross-domain knowledge coordination [21][22][23][24]. For example, such time interval descriptions as “before we calculate the total weight value of a new product, we should get every part’s weight value in advance”, and “function  $A$  could only be triggered after it receives a parameter  $b$  produced during function  $B$ ’s execution” indicate two typical temporal relations. The first temporal relation is initiated by an inherent knowledge-dependency relation that the weight of a new product is the total weight of all parts. The second temporal relation is also initiated by a cross-domain knowledge dependency that function  $A$  demands a parameter  $b$  produced by function  $B$ , otherwise it could not be triggered. Furthermore, as mentioned in [9], “scheduling deals with the assignment of jobs and activities to resources and time ranges in accordance with relevant constraints and requirements.” Here, knowledge dependency relations among task executions are typical “relevant constraints and requirements” in scheduling application. Therefore, to incorporate temporal knowledge into a knowledge modeling framework is helpful for enhancing cross-domain knowledge coordination among task executions. In this regard, current knowledge modeling techniques as presented in [16][17][18][19][20] mainly aims at set up a structured organization for articulating problem-solving components at knowledge-level, rather than scheduling application of knowledge-based collaboration [9].

In view of this observation, a novel knowledge modeling framework, named **POTMe**, is investigated in this paper. This framework not only articulates problem-solving components of a complex problem for cross-domain knowledge coordination, but also incorporates task execution-related temporal knowledge into its specification for further scheduling application.

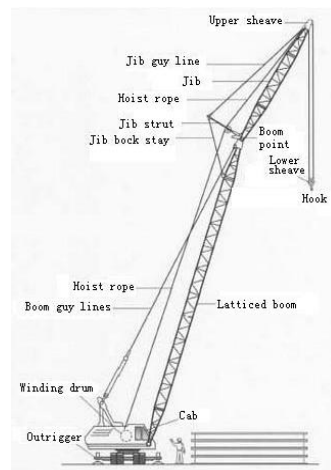
The remainder of this paper is organized as follows. In Section2, a motivating example is put forward for highlighting the core research issues to be explored in this paper. In Section3, a **POTMe** framework is proposed based on problem-solving context analysis. In Section4, temporal knowledge modeling with regard to **POTMe** framework is investigated. In Section5, related works and comparison analysis are presented to evaluate the feasibility of our proposal. Finally, the conclusions and our future work are presented in Section6.

## A Motivating Example

Here, a new product development is presented for highlighting the key topics that will be investigated in this paper. Fig.1.a illustrates a framework of an expected mobile crane product, whose functional units are indicated by a set of ontology definitions. In practice, product-defining process is often initiated by incomplete, ill structured and poor quality information derived from stakeholders’ requirements. Generally, to satisfy the stakeholders’ requirements, designers often

make decisions in an empirical manner by using their personal knowledge or experience achieved from the past related produce development. Typical, it has been widely cited that most managers/designers often refer to solutions of the similar problems as the first step for their new design tasks [25][26]. Therefore, a solution of similar project is very helpful for a new project. For instance, an existing product of a truck crane as illustrated in Fig.1.b would be referred to, if its truck crane's boom, chassis, turning platform or outrigger were similar to the expected mobile crane. If so, the assembly and operation regulations enacted among these functional units could also be referred to for new mobile crane development.

Here, Fig.1.a serve as a knowledge modeling framework for incorporating stakeholders' requirements and existing knowledge assets into designers' perception of expected product [1][15]. It provides a helpful global mental model to be shared by the participant teams or members for conceiving the expected product, as well as for their later design and manufacturing activities [1].



(a). An expected crane product



(b). A referred crane product

Fig.1. An expected crane product and its available analogue for its development.

Moreover, to gain and retain business advantages in a competitive business arena, design and manufacturing activities should be effectively orchestrated for timely responding to market requirements. Here, scheduling application is often navigated by knowledge-dependency relations specified by the knowledge modeling framework as illustrated by Fig.1.a. For example, in Fig.1.a, there are certain assembly relations between hoist rope and latticed boom. Here, the length of a hoist rope is calculated only after the outline dimension of latticed boom and its jib are determined. If the start time and the end time of a task execution engaged in this collaboration could not be specified in a quantitative way, its uncertain working status would greatly influence other task executions that have knowledge and temporal dependencies with it. It typically indicates a kind of execution-related temporal knowledge. Therefore, temporal knowledge modeling should be incorporated into a comprehensive knowledge modeling framework in a compatible way.

Accordingly, two interesting topics could be drawn out from the example to highlight the motivations of this paper.

(1) To incorporate stakeholders' requirements, designers' perceptions and past experience into new concrete product design and manufacturing process, a well structured knowledge modeling framework is demanded. It plays as a global shared mental model for cross-domain collaboration among product conceiving, product design and product manufacturing activities.

(2) To orchestrate problem-solving activities and their cross-domain collaboration for satisfying certain deadline, temporal knowledge modeling should be incorporated into a structured

knowledge modeling framework in a compatible way. It is often reified by a group of temporal-dependency relations among task executions. Here, temporal knowledge modeling is indispensable for evaluating new product development from scheduling application perspective.

Initiated by the first topic, a novel knowledge modeling framework, named *POTMe*, is firstly investigated in Section3. In Section4 and Section4, some issues with regard to *POTMe*'s temporal knowledge modeling would be discussed initiated by the second topic as listed above.

## **A Knowledge Modeling Framework *POTMe***

Now, accessing to a wide range of information is easy. However, the increasing size of complex problems makes it more and more difficult to absorb all pertinent information in a timely manner [25][26][27][28][29]. With the goal of gaining comprehensive insights, the following issues should be taken into consideration for modeling a complex problem in knowledge level.

(1)How to deal with the complexity of a complex problem object for its collaborative solving?

In the past decade, object-oriented decomposition technology has been proved as an effective method for degrading the complexity of problem specification, no matter for qualitative problem cognizing or for quantitative problem solving [30][31][32][33]. Taking advantage of this technology, a complex problem could be decomposed into a set of domain problem ontology in a hierarchical way for further cognizing and solving [27][28][30][34][35]. Once a complex problem is specified by a set of domain problem ontologies, all the domain-specific knowledge could be absorbed from different perspectives for collaborative problem solving

(2)How to specify knowledge-dependency relations in collaborative problem solving?

In [36], ontologies are treated as "agreements about shared conceptualizations". Here, the shared content between two domain problem ontology typically indicates knowledge-dependency relations among task executions engaged in collaborative problem solving. In practice, attributes are key issues for specifying an ontology [35]. The associations among attributes of domain problem ontologies essentially state knowledge-dependency relations for collaborative problem solving. Therefore, taking advantage of object-oriented decomposition techniques, we not only disclose latent domain problem ontologies encompassed in a complex problem, but also specify knowledge dependency among domain problem ontologies for collaborative problem solving [35]. Here, we believe that domain problem ontologies and their attributes are always fulfilled by attribute-driven task executions, and collaboration among task executions is often initiated by knowledge dependency among attributes of domain problem ontologies

(3)How to schedule task executions and their collaboration?

Collaborative problem solving is often enabled by a group of knowledge-intensive task executions. Moreover, a task execution is often promoted in a collaborative context initiated by knowledge-dependency relations as we discussed above. Accordingly, temporal scheduling of task executions and their collaboration should be compatible with knowledge-dependency relations specified among task executions [28][34][37][38][39]. How to incorporate temporal knowledge into the collaborative context for promoting collaborative problem solving is a key issue that will be investigated in this paper.

(4)How to promote task execution with domain knowledge support?

As mentioned in [40], domain knowledge does strongly depend on particular task at hand. In practice, execution-specific domain knowledge is often reified into a group of problem-solving methods. If problem-solving methods could be discovered or determined in advance, task scheduling would be successfully put into effect. Otherwise, task executions of collaboration problem solving would be in a logjam state, if we cannot find an effective solving proposal for task execu-

tions. Therefore, specifications of task-oriented problem solving methods should also be incorporated into task execution and collaborative context [31][33][41][42][43].

In view of these observations, a knowledge modeling framework, named **POTMe**, is presented in this section for promoting collaborative problem solving. This framework not only incorporates problem cognition-related knowledge into its specification, but also incorporates task execution-related knowledge, especially temporal knowledge, into its comprehensive application context. In its application logic, it covers all the key knowledge assets associated with **P**roblem **O**ntology definition, **T**ask scheduling, and **M**ethod discovery.

**Definition1.** **POTMe** framework is formally represented as a mapping from a six-dimensional space of  $\{P, A, KR, T, M, TR\}$  to collaborative problem solving. The notations are depicted as below:

- **Complex problem,  $P = \{SubP_1, \dots, SubP_n\}$ .**  $P$  consists of a set of domain problem ontology that will be assigned to a group of tasks for problem solving.

Here, please note that a domain problem ontology is essentially a problem that are not yet solved at this stage. To highlight its current state, we will use  $SubP_i$  to indicate a domain problem ontology in this paper.

- **Attributes,  $A$ .** For each domain problem ontology,  $SubP_i$ , there is an  $A_i$  uniquely associated with  $SubP_i$ . Here,  $A_i$  stands for a set of attributes. It consists of all the attributes of  $SubP_i$ , i.e.,  $A_i = \{a_{i-1}, \dots, a_{i-m}\}$ , in which  $a_{i-x}$  stands for a attribute or a set of attributes associated with a unique task execution.

- **Knowledge-dependency Relations,  $KR = \{KR_{i-j} | i = 1, \dots, n; j=1, \dots, n\}$ .**  $KR_{i-j}$  stands for a directed knowledge-dependency relation between  $SubP_i$  and  $SubP_j$ . In this paper, it is specified by the associated relations between  $A_i$  and  $A_j$ , i.e.,  $KR_{i-j} = A_i \times A_j$ . If  $i \neq j$ ,  $KR_{i-j}$  indicates a cross-domain knowledge-dependency relation between two different domain ontologies of  $SubP_i$  and  $SubP_j$ , that is,  $A_j$ -around calculating process depends on  $A_i$ -around calculating process. If  $i=j$ ,  $KR_{i-j}$  indicates an internal knowledge-dependency relation among  $a_{i-x}$  inside  $SubP_i$ , in which  $a_{i-x} \in A_i$  is held.

Here,  $P$ ,  $A$ , and  $KR$  indicate basic knowledge assets for qualitative problem cognition. Initiated by knowledge dependency, collaborative problem solving could be formalized into  $(SubP_i.A_i)KR_{i-j}(SubP_j.A_j)$  in logic through articulating the relations among attributes of domain problem ontologies, where  $i = 1, \dots, n; j=1, \dots, n$ . It is essentially navigated by *Knowledge-dependency Relations*  $KR$ . To verify domain problem ontologies and their attributes, attribute-driven task specifications and task execution-related knowledge assets should be taken into consideration for further application.

- **Tasks,  $T$ .** For each domain problem ontology,  $SubP_i$ , there is a  $T_i$  uniquely associated with  $SubP_i$ .  $T_i$  stands for a task domain that consists of a set of task specifications for calculating  $SubP_i$ 's attributes  $A_i$ , i.e.,  $T_i = \{t_{i-1}, \dots, t_{i-m}\}$ , in which  $t_{i-x}$  stands for a attribute-driven task execution.

In practice, a task execution may fulfill a group of  $a_{i-x}$  (See our case study in Section5). Here, for better understanding the mapping relation between  $T_i$  and  $A_i$ , we suppose that there is just one uniquely task  $t_{i-x}$ ,  $t_{i-x} \in T_i$ , for fulfilling  $a_{i-x}$ ,  $a_{i-x} \in A_i$ , that is, there are just  $m$  task to be assigned for respectively achieving the  $m$  attributes contained in  $A_i$ 's definitions. Furthermore,  $t_{i-x}(a_{i-x})$  is used to indicate a task execution for fulfilling  $a_{i-x}$ .

- **Methods,  $M$ .** To promote  $T_i$ 's executions in knowledge level, a  $M_i$  is uniquely associated with  $T_i$ 's executions for fulfilling  $SubP_i$ 's attributes  $A_i$ , i.e.,  $M_i = \{m_{i-1}, \dots, m_{i-m}\}$ , in which  $m_{i-1}$  stands for a task-oriented solving method.

Here, for brevity and without the loss of generality, for the mapping relations between  $T_i$  and  $M_i$ , suppose that there is just one solving method  $m_{i-x}$ ,  $m_{i-x} \in M_i$ , associated with  $t_{i-x}$ 's execution.

• *Temporal-dependency Relations*,  $\mathbf{TR} = \{KR_{i-j}: TR_{i-j}|i = 1, \dots, n; j=1, \dots, n\}$ .  $TR_{i-j}$  stands for a directed temporal-dependency relation between  $T_i$  and  $T_j$ , that is,  $T_j$ 's execution depends on  $T_i$ 's calculating process. If  $i \neq j$ ,  $TR_{i-j}$  indicates a cross-domain temporal-dependency relation between two different task executions of  $T_i$  and  $T_j$ . If  $i = j$ ,  $TR_{i-j}$  indicates an internal temporal-dependency relation among  $t_{i-x}$  inside  $T_i$ , in which  $t_{i-x} \in T_i$  is held. In application logic,  $TR_{i-j}$  is initiated by  $KR_{i-j}$  specification, that is, if there is no knowledge-dependency relation between two task executions, no temporal-dependency would be taken into consideration in their executions.

$\mathbf{T}$ ,  $\mathbf{M}$ , and  $\mathbf{TR}$  indicate task execution-related knowledge assets for quantitative problem solving and collaboration. Moreover,  $\mathbf{TR}$  subscribes to Allen's [22] representation of standard time and relations. Has-Earliest-Start-Time, Has-Latest-Start-Time, Has-Earliest-End-Time, and Has-Latest-End-Time as specified in [9] are typical temporal specification for  $\mathbf{TR}$ 's knowledge modeling.

In Definition1, for collaborative problem solving, its knowledge modeling process consists of two knowledge-based application levels for heuristic knowledge discovery. The first application level focuses on problem cognition. Knowledge exploration mainly aims at recognizing the ontology and their attributes engaged in a problem domain, as well as the associations among the attributes. The second application level focuses on task executions for fulfilling and verifying problem specification derived from the first level. Knowledge exploration mainly aims at conceiving or selecting a suitable method for fulfilling required functions or attributes based on certain temporal discipline.

Current knowledge modeling techniques [16][17][18][19][20] often pay attention to problem analysis and knowledge discovery. They often take little consideration of temporal knowledge in their application. In practice, temporal knowledge is demanded in a wide range of disciplines. Coordination among task executions and their collaboration often benefit from certain temporal knowledge modeling [22][24][44]. In our research, issue of temporal knowledge is incorporated into a comprehensive knowledge modeling framework as defined in Definition1. In the following sections, we mainly focus on investigating how to incorporate  $\mathbf{TR}$ 's temporal knowledge modeling into **POTMe** framework, by knowledge-dependency analysis.

Here, we will conclude this section with some discussions around the example presented in Section2, for demonstrating **POTMe**'s typical application from requirement engineering perspective [45]. In Section6, it would be investigated in detail to demonstrate the deployment of **POTMe** framework.

#### (1) What is *Complex problem P*?

For the motivating example presented in Section2, an expected truck crane illustrated by Fig.1.a indicates a *complex problem P*. In its ontology specifications, *Boom*, *Hydraulic System*, *Turning Platform*, *Chassis*, *Outrigger*, *Power System*, *Electrical Operation System*, and *Attachments* are typical domain problem ontologies.

#### (2) How to specify *Attributes A*?

Here, we take domain problem ontology of *Boom* as an example to specify its typical attributes (i.e.,  $\mathbf{A}_{\text{Boom}}$ ).

- What is the type of boom construct: a latticed boom or a trunk boom?
- If boom's profile is a trunk type, what is its profile in geometrical shape: a hexagonal boom profile, an oviform boom profile or an all-round octagonal boom profile?
- How many sections encompassed in the boom: three sections, four sections, five sections, or six sections?
- Is there a jib structure? If so, how many sections encompassed in the jib: one sections or two

sections?

- What is the style of its telescoping system: single cylinder plus wire ropes or double cylinders plus wire ropes? Synchronous extension or asynchronous extension?
- What is its lifting capacity curve?

According to these analyses,  $A_{Boom} = \{Trunk, Oviform\ profile, 4\text{-Sections}, 2\text{-Section folding jib}, Synchronous\text{-Extension}, 50t\}$  indicates a list of typical attributes associated with domain problem ontology of *Boom*.

(3) How to specify *Knowledge-dependency Relations KR*?

Fig.1.a typically illustrates assembly and operation relations among domain problem ontologies. **KR** is always initiated by these assembly and operation relations. For instance, boom's lifting capacity is a typical attribute associated with domain problem ontology of *Boom*. It is often calculated based on the parameters of outrigger's extension span, weight of the counterbalance, and power parameters of the power system (e.g., rated power, rated torque), etc. These parameters are derived from the attributes of domain problem ontology of *Outrigger*, *Turning Platform*, and *Power System*. It typically indicates a knowledge-dependency relation between *Boom* and  $\{Outrigger, Turning\ Platform, Power\ System\}$ .

(4) How to specify *Tasks T*?

Typically, we take *Tasks*  $T_{Boom}$  initiated by *Boom* as an example. Here, let  $Boom.A_{Boom} = \{Trunk, Oviform\ profile, 4\text{-Sections}, 2\text{-Section folding jib}, Synchronous\text{-Extension}, 50t\}$ .  $T_{Boom}$ 's enactment is often promoted by assigning its task items to suitable teams or person for fulfilling these attributes. Here, the collaboration among task executions is navigated by the attributes' association specified by **KR**.

(5) How to discover valid *Methods M*?

In [9][11][34], some strategies are presented for method discovery related to knowledge modeling. Here, task-oriented method discovery is often promoted by decision-making based on past experiences or by knowledge exploring from existing repository [1][5][29]. As attribute-driven task executions, method discoveries are also initiated by *Attributes A*. In our case study, it is reified into function mapping or attribute mapping. For example, *Boom* could be used as an index for absorbing similar functional unit from existing product spectrums. Moreover, an attribute derived from  $\{Trunk, Oviform\ profile, 4\text{-Sections}, 2\text{-Section folding jib}, Synchronous\text{-Extension}, 50t\}$  could also be used as an index for method discovery. Through function and attribute mapping, related implementation knowledge such as design methods and manufacturing techniques could be recruited for facilitating expected boom's design and manufacturing.

(6) How to specify *Temporal-dependency Relations TR* based on *Knowledge-dependency Relations KR*?

Temporal knowledge modeling around **TR** is one of the key topics. It will be investigated in Section 4.

## Temporal Knowledge Modeling around **TR** inside **POTME** Framework

### 1.Context Mapping from **KR** to **TR**

In Definition1, **KR** indicates a context related to problem definition, and **TR** specifies a context related to task execution. They are two different domain-specific contexts. However, as  $TR_{i-j}$  is initiated by  $KR_{i-j}$  specification in application logic in its definition, context mapping between **KR** and **TR** is firstly discussed for promoting their coordination.

**Definition2.** Let  $A_{i-j} = \{a_{i-x} \mid a_{i-x} \in A_i \wedge \exists a_{j-x} \in A_j: (a_{i-x} \ a_{j-x}) \in KR_{i-j}\}$ . In this situation, knowledge co-

ordination between  $T_i$  and  $T_j$  is initiated by  $t_{i-x}(a_{i-x})$ 's executions, in which  $a_{i-x} \in A_{i-j}$  and  $A_{i-j} \subseteq A_i$  are held.

In Definition2,  $t_{i-x}(a_{i-x})$  plays as a knowledge producer for  $T_j$ 's executions, and  $T_j$  plays as a learner (i.e., knowledge consumer) in  $T_i$  and  $T_j$ 's cross-domain collaboration. They share the knowledge assets produced by  $A_{i-j}$ -around task executions. Therefore, cross-domain knowledge coordination among task executions is defined as below.

**Definition3.** For  $T_i$  and  $T_j$ , their knowledge coordination process ( $KC_{i-j}$ ) could be formalized into  $KC_{i-j} = \{(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) | a_{i-x} \in A_{i-j}, a_{j-x} \in A_j, (a_{i-x}, a_{j-x}) \in KR_{i-j}\}$ .

In application logic,  $KC_{i-j}$  is essentially promoted by knowledge dependency  $KR_{i-j}$  specified between  $T_i$  and  $T_j$ . On the other hand, its concrete execution is navigated by certain temporal-dependency specification. Therefore, it integrates  $KR_{i-j}$ 's context and  $TR_{i-j}$ 's context into an incorporated application environment.

**Definition4.** For  $t_{i-x}(a_{i-x})$  and  $t_{j-x}(a_{j-x})$ , to highlight their producer-consumer relation initiated by  $a_{i-x}$ ,  $t_{i-x}(a_{i-x})$  stands for a knowledge producing process, and  $t_{j-x}(\sim a_{i-x})$  stands for a knowledge consuming process engaged in  $t_{j-x}(a_{j-x})$ 's execution, in which  $t_{i-x}(a_{i-x}) \in T_i$ ,  $t_{j-x}(a_{j-x}) \in T_j$ , and  $(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) \in KC_{i-j}$ . Here,  $a_{i-x}$  would be treated as a *process parameter* for promoting  $t_{i-x}(a_{i-x})$  and  $t_{j-x}(a_{j-x})$ 's knowledge coordination.

In this paper, if there is just one element contained in  $A_{i-j}$ , we believe that there is a *single learning path* for cross-domain knowledge coordination between  $T_i$  and  $T_j$ . If there is more than one element contained in  $A_{i-j}$ , we believe that there is a *multi-learning path* engaged in their cross-domain knowledge coordination between  $T_i$  and  $T_j$ . Here, a knowledge coordinating process between  $T_i$  and  $T_j$  is essentially a cross-domain workflow execution process that spans  $KR_{i-j}$ 's context and  $TR_{i-j}$ 's context in a compatible way, in which knowledge producing process and knowledge consuming process as defined by Definition4 are knowledge-intensive workflow fragments for fulfilling certain attributes.

In this situation, workflow patterns as defined in [46] could be improved for demonstrating  $KC_{i-j}$ 's applications. These improved workflow patterns would be named knowledge coordination patterns in this paper for highlighting their knowledge-intensive features. Table1 specifies some typical knowledge coordination patterns between two task executions. Please note that the knowledge coordination patterns as listed in Table1 could be extended to specify knowledge coordination among three or more task domains. For example,  $AND\text{-}Split(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x}), t_{k-x}(a_{k-x}))$  specifies a knowledge coordination situation among three task executions of  $t_{i-x}$ ,  $t_{j-x}$ , and  $t_{k-x}(a_{k-x})$  that are respectively derived from three task domains of  $T_i$ ,  $T_j$  and  $T_k$ , in which  $(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) \in KC_{i-j}$  and  $(t_{i-x}(a_{i-x}), t_{k-x}(a_{k-x})) \in KC_{i-k}$  are held.

As indicated by the author [46], the scope of workflow patterns is limited to static control flow with no explicit parameter. Compared with the workflow pattern definitions as presented in [46], the knowledge coordination patterns as listed in Table1 are explicitly promoted by process parameters of domain-specific attributes. They focus on dynamic knowledge coordination with certain control policy. The control policy will be reified into temporal disciplines that will be investigated in the following sections.

## 2.Temporal-Dependency Analysis among Task Executions

In practice, knowledge coordination is not only a task-driven process, but also a time-consuming process. Generally, temporal knowledge is demanded in a wide range of disciplines, especially for scheduling application [22][24][44]. In [22], some typical temporal relations are presented for specifying temporal dependencies among intervals. As indicated in [23], the temporal logics found in [22] are merely exploited in representing qualitative temporal information. However, in a wide range of application areas, a quantitative temporal analysis is required, especially for



specifying a task execution [24]. Here, some typical time parameters related to quantitative temporal specification are defined as follows. Please note that the time parameters presented in this paper are relative time rather than absolute time.

TABLE1  
TYPICAL KNOWLEDGE COORDINATION PATTERNS BETWEEN TWO TASK EXECUTIONS

Pattern Style	Specification
Sequence	For $t_{i-x}(a_{i-x})$ and $t_{j-x}(a_{j-x})$ , $(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) \in KC_{i-j}$ , if $t_{j-x}(\sim a_{i-x})$ occurs only after $t_{i-x}(a_{i-x})$ 's execution is completed, the knowledge coordination pattern between $t_{i-x}(a_{i-x})$ and $t_{j-x}(a_{j-x})$ would be formalized into Sequence( $t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})$ ).
Parallel	For $t_{i-x}(a_{i-x})$ and $t_{j-x}(a_{j-x})$ , $(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) \in KC_{i-j}$ , if $t_{j-x}(\sim a_{i-x})$ occurs during $t_{i-x}(a_{i-x})$ 's execution, the knowledge coordination pattern between $t_{i-x}(a_{i-x})$ and $t_{j-x}(a_{j-x})$ would be formalized into Parallel( $t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})$ ).
AND-Split	For $t_{i-x}(a_{i-x})$ , $t_{j-x}(a_{j-x})$ and $t_{j-x}(a_{j-x})$ , $t_{i-x}(a_{i-x}) \in T_i$ , $t_{j-x}(a_{j-x}) \in T_j$ , $t_{j-x}(a_{j-x}) \in T_j$ , $t_{j-x}(a_{j-x}) \neq t_{j-x}(\sim a_{i-x})$ , and $a_{j-x} \neq a_{i-x}$ . If $(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) \in KC_{i-j}$ and $(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) \in KC_{i-j}$ are held, that is, both $t_{j-x}(a_{j-x})$ and $t_{j-x}(\sim a_{i-x})$ are enabled or advanced by $t_{i-x}(a_{i-x})$ 's execution, the knowledge coordination pattern among $t_{i-x}(a_{i-x})$ , $t_{j-x}(a_{j-x})$ and $t_{j-x}(\sim a_{i-x})$ would be formalized into AND-Split( $t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x}), t_{j-x}(\sim a_{i-x})$ ).
XOR-Split	For $t_{i-x}(a_{i-x})$ , $t_{j-x}(a_{j-x})$ and $t_{j-x}(a_{j-x})$ , $t_{i-x}(a_{i-x}) \in T_i$ , $t_{j-x}(a_{j-x}) \in T_j$ , $t_{j-x}(a_{j-x}) \in T_j$ , $t_{j-x}(a_{j-x}) \neq t_{j-x}(\sim a_{i-x})$ , and $a_{j-x} \neq a_{i-x}$ . If $(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) \in KC_{i-j}$ or $(t_{i-x}(a_{i-x}), t_{j-x}(\sim a_{i-x})) \in KC_{i-j}$ is held, that is, $t_{i-x}(a_{i-x})$ is followed by either $t_{j-x}(a_{j-x})$ or $t_{j-x}(\sim a_{i-x})$ , the knowledge coordination pattern among $t_{i-x}(a_{i-x})$ , $t_{j-x}(a_{j-x})$ and $t_{j-x}(\sim a_{i-x})$ would be formalized by XOR-Split( $t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x}), t_{j-x}(\sim a_{i-x})$ ).
AND-Join	For $t_{i-x}(a_{i-x})$ , $t_{i-x}(a_{i-x})$ and $t_{j-x}(a_{j-x})$ , $t_{i-x}(a_{i-x}) \in T_i$ , $t_{i-x}(a_{i-x}) \in T_i$ , $t_{j-x}(a_{j-x}) \in T_j$ , $t_{i-x}(a_{i-x}) \neq t_{i-x}(\sim a_{i-x})$ , and $a_{i-x} \neq a_{i-x}$ . If $(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) \in KC_{i-j}$ and $(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) \in KC_{i-j}$ are held, that is, $t_{j-x}(a_{j-x})$ is enabled after the completion of both $t_{i-x}(a_{i-x})$ and $t_{i-x}(\sim a_{i-x})$ , the knowledge coordination pattern $t_{i-x}(a_{i-x})$ , $t_{i-x}(\sim a_{i-x})$ and $t_{j-x}(a_{j-x})$ would be formalized by AND-Join( $t_{i-x}(a_{i-x}), t_{i-x}(\sim a_{i-x}), t_{j-x}(a_{j-x})$ ).
XOR-Join	For $t_{i-x}(a_{i-x})$ , $t_{i-x}(a_{i-x})$ and $t_{j-x}(a_{j-x})$ , $t_{i-x}(a_{i-x}) \in T_i$ , $t_{i-x}(a_{i-x}) \in T_i$ , $t_{j-x}(a_{j-x}) \in T_j$ , $t_{i-x}(a_{i-x}) \neq t_{i-x}(\sim a_{i-x})$ , and $a_{i-x} \neq a_{i-x}$ . If $(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) \in KC_{i-j}$ or $(t_{i-x}(a_{i-x}), t_{j-x}(\sim a_{i-x})) \in KC_{i-j}$ are held, that is, $t_{j-x}(a_{j-x})$ is enabled after either $t_{i-x}(a_{i-x})$ or $t_{i-x}(\sim a_{i-x})$ is executed, the knowledge coordination pattern $t_{i-x}(a_{i-x})$ , $t_{i-x}(\sim a_{i-x})$ and $t_{j-x}(a_{j-x})$ would be formalized by XOR-Join( $t_{i-x}(a_{i-x}), t_{i-x}(\sim a_{i-x}), t_{j-x}(a_{j-x})$ ).

**Definition5.**  $T_i$ 's expected executable duration could be scheduled by a time period  $[T_{Start}(T_i), T_{End}(T_i)]$ , where  $T_{Start}(T_i)$  stands for  $T_i$ 's expected start time,  $T_{End}(T_i)$  stands for  $T_i$ 's expected end time, and  $T_{Start}(T_i) \leq T_{End}(T_i)$  is held.

**Definition6.**  $t_{i-x}(a_{i-x})$ 's expected executable duration could be scheduled by a time period  $[T_{Start}(t_{i-x}(a_{i-x})), T_{End}(t_{i-x}(a_{i-x}))]$ , where  $T_{Start}(t_{i-x}(a_{i-x}))$  stands for  $t_{i-x}(a_{i-x})$ 's expected start time,  $T_{End}(t_{i-x}(a_{i-x}))$  stands for  $t_{i-x}(a_{i-x})$ 's expected end time, and  $T_{Start}(t_{i-x}(a_{i-x})) \leq T_{End}(t_{i-x}(a_{i-x}))$  is held.

Obviously,  $[T_{Start}(t_{i-x}(a_{i-x})), T_{End}(t_{i-x}(a_{i-x}))] \subseteq [T_{Start}(T_i), T_{End}(T_i)]$  is held.

**Definition7.** If there is a knowledge coordination  $KC_{i-j}$  between  $T_i$  and  $T_j$ ,  $t_{j-x}(\sim a_{i-x})$ 's executable duration could be scheduled by a time period  $[T_{Start}(t_{j-x}(\sim a_{i-x})), T_{End}(t_{j-x}(\sim a_{i-x}))]$ , where,  $T_{Start}(t_{j-x}(\sim a_{i-x})) \leq T_{End}(t_{j-x}(\sim a_{i-x}))$  is held.

In application logic, knowledge producing process (i.e., problem solving activities) is always promoted by certain knowledge consuming process (knowledge referring activities). Therefore, a knowledge producing process covers its knowledge consuming process in executive interval, i.e.,  $[T_{Start}(t_{j-x}(\sim a_{i-x})), T_{End}(t_{j-x}(\sim a_{i-x}))] \subseteq [T_{Start}(t_{j-x}(a_{j-x})), T_{End}(t_{j-x}(a_{j-x}))]$ . However, in practice, it is difficult to exactly specify  $[T_{Start}(t_{j-x}(\sim a_{i-x})), T_{End}(t_{j-x}(\sim a_{i-x}))]$  during  $t_{j-x}$ 's execution in interval specification, especially in knowledge-based scheduling application. Therefore, in this paper, we suppose that  $t_{j-x}(\sim a_{i-x})$  equals to  $t_{j-x}(a_{j-x})$ , especially in executive time cost (i.e.,  $[T_{Start}(t_{j-x}(a_{j-x})), T_{End}(t_{j-x}(a_{j-x}))] = [T_{Start}(t_{j-x}(\sim a_{i-x})), T_{End}(t_{j-x}(\sim a_{i-x}))]$ ). It indicates that  $t_{j-x}(a_{j-x})$  and  $t_{j-x}(\sim a_{i-x})$  are executed in a concurrent way.

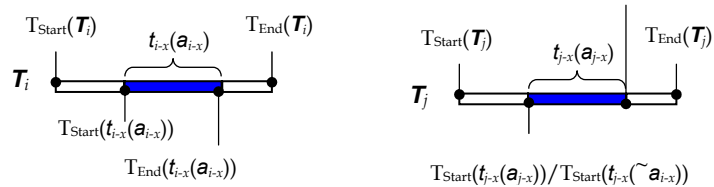


Fig.2 Typical time parameters for specifying  $T_i$  and  $T_j$  with cross-domain knowledge coordination initiated by  $a_{i-x}$ .

Fig.2 illustrates the typical time parameters as defined in Definition5, Definition6, and Definition7.

With these hypotheses, *Temporal-dependency Relations TR* presented in Definition1 could be further specified by Definition 8.

**Definition8.** If there is a knowledge coordination  $KC_{i-j}$  between  $T_i$  and  $T_j$ , temporal-dependency relation  $TR_{i-j}$  between  $T_i$  and  $T_j$  could formalized into  $T_{SE}(t_{i-x}(a_{i-x}))R^T T_{SE}(t_{j-x}(\sim a_{i-x}))$ , in which  $T_{SE}(t_{i-x}(a_{i-x})) \in \{T_{Start}(t_{i-x}(a_{i-x})), T_{End}(t_{i-x}(a_{i-x}))\}$ ,  $T_{SE}(t_{j-x}(\sim a_{i-x})) \in \{T_{Start}(t_{j-x}(\sim a_{i-x})), T_{End}(t_{j-x}(\sim a_{i-x}))\}$ , and  $(t_{i-x}(a_{i-x}), t_{j-x}(a_{j-x})) \in KC_{i-j}$ ; besides,  $R^T$  stands for a logic relation between  $T_{SE}(t_{i-x}(a_{i-x}))$  and  $T_{SE}(t_{j-x}(\sim a_{i-x}))$ , i.e.,  $R^T \in \{<, =, >\}$ .

With these definitions presented in this section, the temporal relations as defined in [22] could be extended with explicit process parameters for modeling knowledge coordination from temporal-dependency aspect. Table2 illustrates some typical temporal-dependency relations, which enables qualitative knowledge coordination into quantitative cross-domain task executions.

### 3.A FollowMe Temporal Reasoning Rule Promoted by Temporal-dependency Relations TR

For a task execution, if it has no temporal-dependency relation with other task executions, it would be scheduled in a self-governing way based on certain expectation and past experiences [24]. For two task executions that have certain knowledge coordination in their collaboration, their scheduling application should take into consideration of the knowledge coordination patterns as listed in Table1 and the temporal-dependency relations as listed in Table2. In view of this demand, a temporal reasoning rule, named *FollowMe*, will be investigated in this section for scheduling cross-domain knowledge coordination based on their temporal-dependency relation.

For two task executions of  $t_{i-x}(a_{i-x})$  and  $t_{j-x}(a_{j-x})$ , suppose that there is a knowledge coordination process for their collaboration. Here, their knowledge coordination is initiated by  $a_{i-x}$ . For  $T_i$ , as  $[T_{Start}(t_{i-x}(a_{i-x})), T_{End}(t_{i-x}(a_{i-x}))] \subseteq [T_{Start}(T_i), T_{End}(T_i)]$  is held in  $T_i$ 's internal temporal distribution, time period  $[T_{Start}(T_i), T_{End}(T_i)]$  covers time period  $[T_{Start}(t_{i-x}(a_{i-x})), T_{End}(t_{i-x}(a_{i-x}))]$ . In this situation,  $T_{Start}(T_i)$  and  $T_{End}(T_i)$  should be always determined firstly in its scheduling application. Thereafter,  $T_{Start}(t_{i-x}(a_{i-x}))$  and  $T_{End}(t_{i-x}(a_{i-x}))$  are determined according to  $T_i$ 's internal temporal distribution.

Therefore,  $T_i$ 's scheduling process associated with  $t_{i-x}(a_{i-x})$ 's execution is promoted along a global-to-local scheduling path, i.e.,  $[T_{Start}(T_i), T_{End}(T_i)] \mapsto [T_{Start}(t_{i-x}(a_{i-x})), T_{End}(t_{i-x}(a_{i-x}))]$ . For  $T_j$ , although time period  $[T_{Start}(T_j), T_{End}(T_j)]$  covers time period  $[T_{Start}(t_{j-x}(\sim a_{i-x})), T_{End}(t_{j-x}(\sim a_{i-x}))]$  in  $T_j$ 's internal temporal distribution,  $T_j$ 's has a different temporal scheduling path associated with

TABLE2  
TYPICAL TEMPORAL-DEPENDENCY RELATIONS BETWEEN TWO TASK EXECUTIONS

Style	Specification
Before	If $T_{End}(t_{i-x}(a_{i-x})) < T_{Start}(t_{j-x}(\sim a_{i-x}))$ , the temporal-dependency relation between $t_{i-x}$ and $t_{j-x}$ would be defined as a Before style. It would be formalized into $Before(t_{i-x}(a_{i-x}), t_{j-x}(\sim a_{i-x}); T_{i-j})$ in this paper, where $T_{i-j}$ stands for a time period from $T_{End}(t_{i-x}(a_{i-x}))$ to $T_{Start}(t_{j-x}(\sim a_{i-x}))$ , i.e., $T_{i-j} = T_{Start}(t_{j-x}(\sim a_{i-x})) - T_{End}(t_{i-x}(a_{i-x}))$ .
Meet	If $T_{End}(t_{i-x}(a_{i-x})) = T_{Start}(t_{j-x}(\sim a_{i-x}))$ , the temporal-dependency relation between $t_{i-x}$ and $t_{j-x}$ would be defined as a Meet style. It would be formalized into $Meet(t_{i-x}(a_{i-x}), t_{j-x}(\sim a_{i-x}))$ in this paper.
Overlap	If $(T_{Start}(t_{j-x}(\sim a_{i-x})) < T_{End}(t_{i-x}(a_{i-x}))) \wedge (T_{Start}(t_{i-x}(a_{i-x})) < T_{Start}(t_{j-x}(\sim a_{i-x})))$ , the temporal-dependency relation between $t_{i-x}$ and $t_{j-x}$ would be defined as an Overlap style. It would be formalized into $Overlap((t_{i-x}(a_{i-x}), t_{j-x}(\sim a_{i-x})); T_{i-j})$ in this paper, where $T_{i-j}$ stands for a time period from $T_{Start}(t_{j-x}(\sim a_{i-x}))$ to $T_{End}(t_{i-x}(a_{i-x}))$ , i.e., $T_{i-j} = T_{End}(t_{i-x}(a_{i-x})) - T_{Start}(t_{j-x}(\sim a_{i-x}))$ .
Start	If $T_{Start}(t_{i-x}(a_{i-x})) = T_{Start}(t_{j-x}(\sim a_{i-x}))$ , the temporal-dependency relation between $t_{i-x}$ and $t_{j-x}$ would be defined as a Start style. It would be formalized into $Start(t_{i-x}(a_{i-x}), t_{j-x}(\sim a_{i-x}))$ in this paper.
During	If $(T_{Start}(t_{i-x}(a_{i-x})) < T_{Start}(t_{j-x}(\sim a_{i-x}))) \wedge (T_{End}(t_{i-x}(a_{i-x})) > T_{End}(t_{j-x}(\sim a_{i-x})))$ , the temporal-dependency relation between $t_{i-x}$ and $t_{j-x}$ would be defined as a During style. It would be formalized into $During(t_{i-x}(a_{i-x}), t_{j-x}(\sim a_{i-x}))$ in this paper.
Finish	If $T_{End}(t_{i-x}(a_{i-x})) = T_{End}(t_{j-x}(\sim a_{i-x}))$ , the temporal-dependency relation between $t_{i-x}$ and $t_{j-x}$ would be defined as a Finish style. It would be formalized into $Finish(t_{i-x}(a_{i-x}), t_{j-x}(\sim a_{i-x}))$ in this paper.
Equal	If $(T_{Start}(t_{i-x}(a_{i-x})) = T_{Start}(t_{j-x}(\sim a_{i-x}))) \wedge (T_{End}(t_{i-x}(a_{i-x})) = T_{End}(t_{j-x}(\sim a_{i-x})))$ , the temporal-dependency relation between $t_{i-x}$ and $t_{j-x}$ would be defined as a Equal style. It would be formalized into $Equal(t_{i-x}(a_{i-x}), t_{j-x}(\sim a_{i-x}))$ in this paper.

$t_{j-x}(\sim a_{i-x})$ 's execution. More specifically, as  $T_{Start}(t_{j-x}(\sim a_{i-x}))$  and  $T_{End}(t_{j-x}(\sim a_{i-x}))$  is always triggered by  $T_{Start}(t_{i-x}(a_{i-x}))$  and  $T_{End}(t_{i-x}(a_{i-x}))$ , navigated by this temporal-dependency relation,  $[T_{Start}(t_{i-x}(a_{i-x})), T_{End}(t_{i-x}(a_{i-x}))] \mapsto [T_{Start}(t_{j-x}(\sim a_{i-x})), T_{End}(t_{j-x}(\sim a_{i-x}))]/[T_{Start}(t_{j-x}(a_{j-x})), T_{End}(t_{j-x}(a_{j-x}))]$  is always put into effect in scheduling  $t_{j-x}(a_{j-x})$ 's execution. In this situation, for  $T_j$ 's scheduling application, it is initiated by  $[T_{Start}(t_{j-x}(\sim a_{i-x})), T_{End}(t_{j-x}(\sim a_{i-x}))]/[T_{Start}(t_{j-x}(a_{j-x})), T_{End}(t_{j-x}(a_{j-x}))]$ . Only after  $T_{Start}(t_{j-x}(a_{j-x}))$  and  $T_{End}(t_{j-x}(a_{j-x}))$  are determined,  $[T_{Start}(T_j), T_{End}(T_j)]$  could be deduced according to  $T_j$ 's internal temporal distribution. Therefore,  $T_j$ 's scheduling process is promoted along a local-to-global scheduling logic, i.e.,  $[T_{Start}(t_{j-x}(\sim a_{i-x})), T_{End}(t_{j-x}(\sim a_{i-x}))] \mapsto [T_{Start}(T_j), T_{End}(T_j)]$ . It is different from  $T_i$ 's scheduling process in application, as it is navigated by certain temporal-dependency constraints derived from  $T_i$ .

According to these analyses, a temporal reasoning rule is defined as follows, for evaluating cross-domain knowledge coordination between two task domains.

**Definition9.** For two task executions of  $t_{i-x}(a_{i-x})$  and  $t_{j-x}(a_{j-x})$ , their knowledge coordination initiated by  $a_{i-x}$  could be scheduled along the following path:  $[T_{Start}(T_i), T_{End}(T_i)] \mapsto [T_{Start}(t_{i-x}(a_{i-x})), T_{End}(t_{i-x}(a_{i-x}))] \mapsto [T_{Start}(t_{j-x}(\sim a_{i-x})), T_{End}(t_{j-x}(\sim a_{i-x}))]/[T_{Start}(t_{j-x}(a_{j-x})), T_{End}(t_{j-x}(a_{j-x}))] \mapsto [T_{Start}(T_j), T_{End}(T_j)]$ .

In this paper, the cross-domain scheduling logic as defined by Definition9 would be named as *FollowMe* temporal reasoning rule. Furthermore, the deduced unique temporal state of  $T_j$  would be treated as a *runtime temporal state* associated with a process parameter  $a_{i-x}$ . It would be indicated by  $T_j(\sim a_{i-x})$  in this paper for highlighting cross-domain knowledge coordination between  $T_i$  and  $T_j$ . Once  $T_j$ 's runtime temporal state  $T_j(\sim a_{i-x})$  is determined, the global time cost of  $T_i$  and  $T_j$ 's executions could be calculated out in an incorporated executive environment. It is helpful for evaluating  $T_i$  and  $T_j$ 's collaboration in time cost, which is a key demands in scheduling application.

**Definition10.** Let  $T_{Start}(T_j(\sim a_{i-x}))$  and  $T_{End}(T_j(\sim a_{i-x}))$  respectively stand for  $T_j(\sim a_{i-x})$ 's start time and its end time. Let  $T_{Start} = \min\{T_{Start}(T_i), T_{Start}(T_j(\sim a_{i-x}))\}$ ,  $T_{End} = \max\{T_{End}(T_i), T_{End}(T_j(\sim a_{i-x}))\}$ , the global time cost of  $T_i$  and  $T_j$ 's executions, i.e.,  $T_{TC(i-j)}(a_{i-x})$ , could be calculated in an incorporated executive environment, i.e.,  $T_{TC(i-j)}(a_{i-x}) = T_{End} - T_{Start}$ .

Here, there is just one process parameter between  $T_i$  and  $T_j$  in their cross-domain collaboration. If there are  $n$  process parameters between  $T_i$  and  $T_j$  in their collaboration,  $T_j$  would have  $n$  runtime temporal states. Each runtime temporal state is associated with a unique process parameter. For  $T_j$ 's  $n$  runtime temporal states, taking advantage *FollowMe* temporal reasoning rule and Definition10, we could also deduce out the global time cost of  $T_i$  and  $T_j$ 's executions. The more complex application situations would be investigated in the following section.

#### 4. *FollowMe* Temporal Reasoning Rule's Application Analyses

In this section, for better understating our discussion, we will use some typical examples to demonstrate *FollowMe* temporal reasoning rule's applications. These examples could be generalized for similar applications, which benefit complex situations in further applications.

##### Case1: Temporal reasoning between two task domains initiated by one process parameter

Typically, we will firstly consider two task domains of  $T_i$  and  $T_j$ . Suppose that  $T_i$ 's expected executable duration is 10 time units, and  $T_j$ 's duration is 9 time units. Taking no temporal-dependency relation into consideration,  $T_i$  and  $T_j$  are independently illustrated by Fig.3 and Fig.4. During  $T_i$ 's execution as illustrated in Fig.3, there is a  $t_{i-x}(a_{i-x})$ .  $t_{i-x}(a_{i-x})$ 's expected start time is at 3 time point and its expected end time is at 8 time point, i.e.,  $T_{Start}(t_{i-x}(a_{i-x})) = 3$ , and  $T_{End}(t_{i-x}(a_{i-x})) = 8$ . During  $T_j$ 's execution as illustrated in Fig.4, there is a  $t_{j-x}(a_{j-x})$ .  $t_{j-x}(a_{j-x})$ 's expected start time is at 3 time point and its expected end time is at 7 time point, i.e.,  $T_{Start}(t_{j-x}(a_{j-x})) = 3$ , and  $T_{End}(t_{j-x}(a_{j-x})) = 7$ . Here, Fig.3 and Fig.4 indicate two isolated executive environments respectively indi-

cated by two different time axes of  $t'$  and  $t''$ . Here, we use  $T_{\text{Isolated}}(\mathbf{T}_i)$  and  $T_{\text{Isolated}}(\mathbf{T}_j)$  to respectively indicate the executable durations of  $\mathbf{T}_i$  and  $\mathbf{T}_j$  in their isolated executive environments, i.e.,  $T_{\text{Isolated}}(\mathbf{T}_i) = 10$  time units and  $T_{\text{Isolated}}(\mathbf{T}_j) = 9$  time units.

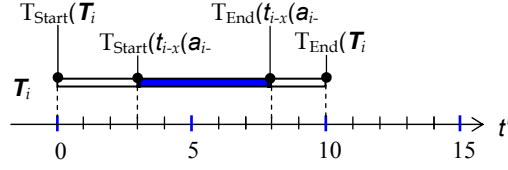


Fig.3  $\mathbf{T}_i$ 's temporal parameters and their distributions associated with  $a_{i-x}$  specified in its isolated executive environment (i.e., time axis  $t'$ )

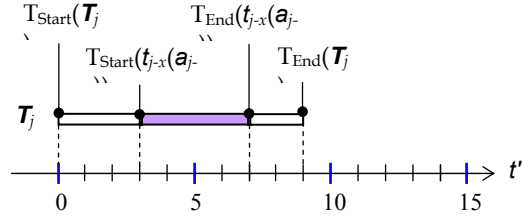


Fig.4  $\mathbf{T}_j$ 's temporal parameters and their distributions associated with  $a_{j-x}$  specified in its isolated executive environment (i.e.,  $t''$ )

In practice, taking into no temporal-dependency relation into consideration, a task domain is often scheduling in a self-governing way as demonstrated by Fig.3 and Fig.4. Now, a Before style temporal-dependency relation is brought into  $\mathbf{T}_i$  and  $\mathbf{T}_j$ 's collaboration for their cross-domain knowledge coordination initiated by  $a_{i-x}$ . More specifically,  $\text{Before}(t_{i-x}(a_{i-x}), t_{j-x}(\sim a_{i-x}); T_{i-j})$  is held between  $t_{i-x}(a_{i-x})$  and  $t_{j-x}(a_{j-x})$ , in which  $T_{i-j}=1$  time unit is held. With these temporal constraints,  $\mathbf{T}_i$  and  $\mathbf{T}_j$ 's temporal parameters should be re-specified in an incorporated executive environment for orchestrating cross-domain knowledge coordination between  $t_{i-x}(a_{i-x})$  and  $t_{j-x}(\sim a_{i-x})$ . For brevity and without the loss of generality, time axis  $t'$  is recruited as a united time axis  $t$  for indicating their incorporated executive environment.

Facilitating our discussion, some duration parameters associated with  $\mathbf{T}_j$  are specified as below.

- (1) Let  $T_{j-d1}$  indicate the duration from  $T_{\text{Start}}(\mathbf{T}_j)$  to  $T_{\text{Start}}(t_{j-x}(a_{j-x}))$ , i.e.,  $T_{j-d1} = 3$  time units as illustrated in Fig.4;
- (2) Let  $T_{j-d2}$  indicate the duration from  $T_{\text{Start}}(t_{j-x}(a_{j-x}))$  to  $T_{\text{End}}(t_{j-x}(a_{j-x}))$ , i.e.,  $T_{j-d2} = 4$  time units as illustrated in Fig.4;
- (3) Let  $T_{j-d3}$  indicate the duration from  $T_{\text{End}}(t_{j-x}(a_{j-x}))$  to  $T_{\text{End}}(\mathbf{T}_j)$ , i.e.,  $T_{j-d3} = 2$  time units as illustrated in Fig.4.

Taking advantage of these parameters and *FollowMe* temporal reasoning rule,  $\mathbf{T}_j$ 's time parameters are re-specified, as below, in an incorporated executive environment indicated by a united time axis  $t$ .

- (1)  $T_{\text{Start}}(t_{j-x}(\sim a_{i-x})) = T_{\text{End}}(t_{i-x}(a_{i-x})) + T_{i-j} = 8 + 1 = 9$ .
- (2)  $T_{\text{End}}(t_{j-x}(\sim a_{i-x})) = T_{\text{Start}}(t_{j-x}(\sim a_{i-x})) + T_{j-d2} = 9 + 4 = 13$ .
- (3)  $T_{\text{Start}}(\mathbf{T}_j(\sim a_{i-x})) = T_{\text{Start}}(t_{j-x}(\sim a_{i-x})) - T_{j-d1} = 9 - 3 = 6$ .
- (4)  $T_{\text{End}}(\mathbf{T}_j(\sim a_{i-x})) = T_{\text{Start}}(\mathbf{T}_j(\sim a_{i-x})) + T_{j-d1} + T_{j-d2} + T_{j-d3} = 6 + 3 + 4 + 2 = 15$ .
- (5)  $T_{\text{TC}(i-j)}(a_{i-x}) = T_{\text{End}}(\mathbf{T}_j(\sim a_{i-x})) - T_{\text{Start}}(\mathbf{T}_i) = 15 - 0 = 15$ .

Fig.5 illustrates an incorporated executive environment, in which  $\mathbf{T}_j$ 's runtime temporal state is associated with a group of re-specified temporal parameters. Please note that the re-specified

temporal parameters associated with  $T_j$ 's runtime temporal state have the same distributions as they are specified in  $T_j$ 's insulated executive environment (See Fig.4).

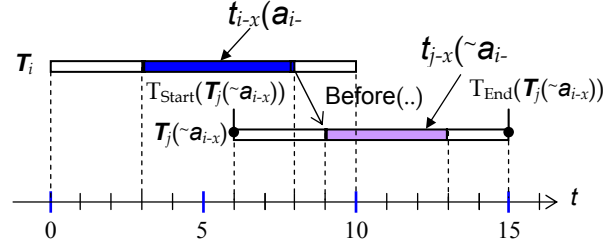


Fig.5  $T_j$ 's re-specified temporal parameters and their distributions in  $T_i$  and  $T_j$ 's incorporated executive environment (i.e.,  $t$ ).

In Fig.5, the start time of  $T_j$ 's runtime temporal state is at the 6 time point as indicated by time axis  $t$  (i.e.,  $T_{\text{Start}}(T_j(a_{i-x})) = 6$ ). In practice, it is an ideal start time to meet  $\text{Before}(t_{i-x}(a_{i-x}), t_{j-x}(a_{i-x}); T_{i-j})$  temporal-dependency relation. Otherwise, some additional time cost would be occupied or temporal-dependency relation  $\text{Before}(t_{i-x}(a_{i-x}), t_{j-x}(a_{i-x}); T_{i-j})$  would be spoiled. More specifically, suppose  $T_j$  starts its execution at the 5 time point, i.e.,  $T_j(a_{i-x}) = 5$ . According to the Before style temporal-dependency relation,  $t_{j-x}(a_{i-x})$  just could start its execution at the 9 time point i.e.,  $T_{\text{Start}}(t_{j-x}(a_{i-x})) = 9$ . With this hypothesis, we could deduce that the executable duration from  $T_{\text{Start}}(T_j(a_{i-x}))$  to  $T_{\text{Start}}(t_{j-x}(a_{i-x}))$  is 4 time units, i.e.,  $T_{j-d1} = 4$ . Obviously, it extends  $T_{j-d1}$ 's original value (i.e.,  $T_{j-d1} = 3$ ) scheduled in advance. On the other hand, suppose  $T_j$  starts its execution at the 7 time point, i.e.,  $T_j(a_{i-x}) = 7$ . As the internal temporal distribution is fixed as we scheduled in advance (See Fig.4), the duration from  $T_{\text{Start}}(T_j(a_{i-x}))$  to  $T_{\text{Start}}(t_{j-x}(a_{i-x}))$  is a fixed value. In this situation, we could deduce that the duration from  $T_{\text{End}}(t_{i-x}(a_{i-x}))$  to  $T_{\text{Start}}(t_{j-x}(a_{i-x}))$  is 2 time units. As this duration is associated with the parameters of  $T_{i-j}$  as specified in  $\text{Before}(t_{i-x}(a_{i-x}), t_{j-x}(a_{i-x}); T_{i-j})$ , i.e.,  $T_{i-j} = 2$  time units. Obviously, it spoils  $T_{i-j}$ 's original value (i.e.,  $T_{i-j} = 1$ ). Therefore, FollowMe temporal reasoning rule provide an efficient approach to accurately determine  $T_j$ 's active interval for satisfy cross-domain collaboration with  $T_i$ .

#### Case2: Temporal reasoning between two task domains initiated by $n$ process parameters

Suppose that there is a multi-learning path between  $T_i$  and  $T_j$ , that is, there are  $n$  ( $n > 1$ ) process parameters engaged in  $T_i$  and  $T_j$ 's cross-domain collaboration. For brevity and without the loss of generality, let  $a_{i-1}, a_{i-2}, \dots, a_{i-n}$  respectively stands for the  $n$  process parameters, i.e.,  $A_{i-j} = \{a_{i-1}, a_{i-2}, \dots, a_{i-n}\}$ . Besides, suppose there is an unique temporal-dependency relation for  $t_{i-x}(a_{i-x})$  and  $t_{j-x}(a_{i-x})$ 's knowledge coordination process initiated by  $a_{i-x}$  ( $a_{i-x} \in \{a_{i-1}, a_{i-2}, \dots, a_{i-n}\}$ ). With these hypotheses, as we discussed in Case1, a runtime temporal state  $T_j(a_{i-x})$  could be deduced associated with each temporal-dependency relation. Therefore, initiated by the  $n$  process parameters, there would be  $n$  runtime temporal states associated with  $T_j$ 's execution, i.e.,  $T_j(a_{i-1}), T_j(a_{i-2}), \dots, T_j(a_{i-n})$ . For  $T_j(a_{i-x}) \in \{T_j(a_{i-1}), T_j(a_{i-2}), \dots, T_j(a_{i-n})\}$ , it has a start time value (i.e.,  $T_{\text{Start}}(T_j(a_{i-x}))$ ) and a end time value (i.e.,  $T_{\text{End}}(T_j(a_{i-x}))$ ). Associated with  $T_j$ 's  $n$  runtime temporal states, there are  $n$  start time values and  $n$  end time values. In this paper, the minimal value of the  $n$  start time values, i.e.,  $\min\{T_{\text{Start}}(T_j(a_{i-1})), T_{\text{Start}}(T_j(a_{i-2})), \dots, T_{\text{Start}}(T_j(a_{i-n}))\}$ , indicates  $T_j$ 's earliest start time to satisfy  $T_i$  and  $T_j$ 's cross-domain collaboration. In this paper, it is indicated by  $T_{\text{Start-Earliest}}(T_j(KC_{i-j}))$ , i.e.,  $T_{\text{Start-Earliest}}(T_j(KC_{i-j})) = \min\{T_{\text{Start}}(T_j(a_{i-1})), T_{\text{Start}}(T_j(a_{i-2})), \dots, T_{\text{Start}}(T_j(a_{i-n}))\}$ . Similarly, the maximal value of the  $n$  end time values, i.e.,  $\max\{T_{\text{End}}(T_j(a_{i-1})), T_{\text{End}}(T_j(a_{i-2})), \dots, T_{\text{End}}(T_j(a_{i-n}))\}$ , indicates  $T_j$ 's latest end time to satisfy the collaboration between  $T_i$  and  $T_j$ . In this paper, it is indicated by  $T_{\text{End-Latest}}(T_j(KC_{i-j}))$ , i.e.,  $T_{\text{End-Latest}}(T_j(KC_{i-j})) = \max\{T_{\text{End}}(T_j(a_{i-1})),$

$T_{\text{End}}(T_j(\sim a_{i-2})), \dots, T_{\text{End}}(T_j(\sim a_{i-n}))\}$ .

In this paper, we use  $T_{\text{Collaboration}}(T_j(KC_{i-j}))$  to indicate the duration from  $T_{\text{Start-Earliest}}(T_j(KC_{i-j}))$  to  $T_{\text{End-Latest}}(T_j(KC_{i-j}))$ . It specifies  $T_j$ 's active interval to collaborate with  $T_i$  initiated by  $n$  process parameters  $a_{i-1}, a_{i-2}, \dots, a_{i-n}$  in a trade-off way. Generally,  $T_{\text{Collaboration}}(T_j(KC_{i-j}))$  is longer than  $T_j$ 's expected executable durations in an isolated executive environments as illustrated in Fig.4, i.e.,  $T_{\text{Collaboration}}(T_j(\sim a_{i-x})) \geq T_{\text{Isolated}}(T_j)$ .

Once  $T_{\text{Start-Earliest}}(T_j(KC_{i-j}))$  and  $T_{\text{End-Latest}}(T_j(KC_{i-j}))$  is deduced out, taking advantage of Definition10, global time cost of  $T_i$  and  $T_j$ 's executions (i.e.,  $T_{\text{TC}(i-j)}(KC_{i-j})$ ), could be calculated. Here, we typically suppose that there are just two process parameters of  $a_{i-1}$  and  $a_{i-2}$  engaged in  $T_i$  and  $T_j$ 's cross-domain collaboration. Moreover, to simplify our discussion, for  $a_{i-1}$ ,  $T_i$  and  $T_j$  have the same specifications as presented in *Case1*'s analysis. More specifically, for  $a_{i-1}$ , as illustrated by Fig.3 and Fig.4, let  $t_{i-x}(a_{i-x}) = t_{i-1}(a_{i-1})$  and  $t_{j-x}(a_{j-x}) = t_{j-1}(a_{j-1})$ . Besides, there is a Before style temporal-dependency relation for cross-domain knowledge coordination between  $t_{i-1}(a_{i-1})$  and  $t_{j-1}(a_{j-1})$ . Before( $t_{i-1}(a_{i-1}), t_{j-1}(\sim a_{i-1}); T_{i-j}$ ) has the same specification as illustrated in Fig.5, i.e.,  $T_{i-j} = 1$  time unit. For  $a_{i-2}$ ,  $T_i$  and  $T_j$ 's specifications are illustrated by Fig.6 and Fig.7, in which  $T_{\text{Start}}(t_{i-2}(a_{i-2}))=1$ ,  $T_{\text{End}}(t_{i-2}(a_{i-2}))=3$ ,  $T_{\text{Start}}(t_{j-2}(a_{j-2}))=1$ , and  $T_{\text{End}}(t_{j-2}(a_{j-2}))= 5$ . Initiated by  $a_{i-2}$ , there is a Meet style temporal-dependency relation for cross-domain knowledge coordination between  $t_{i-2}(a_{i-2})$  and  $t_{j-2}(a_{j-2})$ , that is, a Meet( $t_{i-2}(a_{i-2}), t_{j-2}(\sim a_{i-2})$ ) style temporal-dependency relation is held between  $t_{i-2}(a_{i-2})$  and  $t_{j-2}(a_{j-2})$ 's collaboration. Fig.8 illustrates the multi-knowledge coordination processes initiated by  $a_{i-1}$  and  $a_{i-2}$  for  $T_i$  and  $T_j$ 's cross-domain collaboration.

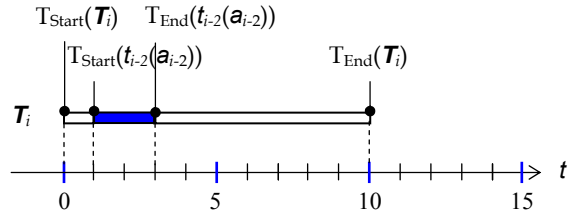


Fig.6  $T_i$ 's temporal parameters and their distributions associated with  $a_{i-2}$  specified in its isolated executive environment (i.e.,  $t'$ )

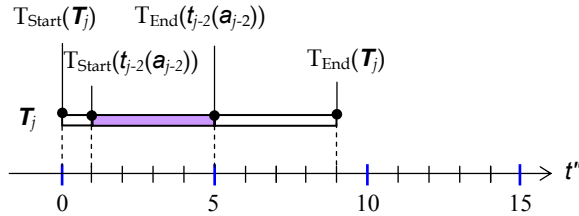


Fig.7  $T_j$ 's temporal parameters and their distributions associated with  $a_{j-2}$  specified in its isolated executive environment (i.e.,  $t''$ )

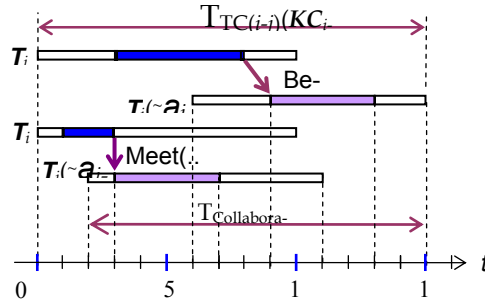


Fig.8 Temporal distributions in  $T_i$  and  $T_j$ 's cross-domain collaboration initiated by two process parameters of  $a_{i-1}$  and  $a_{i-2}$  in an incorporated executive environment (i.e.,  $t$ ).

In Fig.8, for  $T_j(\sim a_{i-1})$ , its expected start time and end time have been deduced as demonstrated in *Case1*'s analysis, i.e.,  $T_{\text{Start}}(T_j(\sim a_{i-1})) = 6$  time units and  $T_{\text{End}}(T_j(\sim a_{i-1})) = 15$  time units as illustrated by Fig.5; for  $T_j(\sim a_{i-2})$ , its runtime state could be also deduced according to *FollowMe* temporal reasoning rule as we demonstrated in *Case1*'s analysis. As a result,  $T_j(\sim a_{i-2})$ 's expected start time is at the 2 time point and its expected end time is at the 11 time point, i.e.,  $T_{\text{Start}}(T_j(\sim a_{i-2})) = 2$  time units and  $T_{\text{End}}(T_j(\sim a_{i-2})) = 11$  time units in an incorporated executive environment. Therefore, we could deduced that  $T_{\text{Start-Earliest}}(T_j(KC_{i-j})) = \min\{T_{\text{Start}}(T_j(\sim a_{i-1})), T_{\text{Start}}(T_j(\sim a_{i-2}))\} = 2$  and  $T_{\text{End-Latest}}(T_j(KC_{i-j})) = \max\{T_{\text{End}}(T_j(\sim a_{i-1})), T_{\text{End}}(T_j(\sim a_{i-2}))\} = 15$  are held. Furthermore, we also could deduced that  $T_{\text{Collaboration}}(T_j(KC_{i-j}))$  is 13 time units, i.e.,  $T_{\text{Collaboration}}(T_j(KC_{i-j})) = T_{\text{End-Latest}}(T_j(KC_{i-j})) - T_{\text{Start-Earliest}}(T_j(KC_{i-j})) = 15 - 2 = 13$  time units. As  $T_{\text{Isolated}}(T_j) = 9$  time units as specified in Fig.4,  $T_{\text{Collaboration}}(T_j(KC_{i-j})) > T_{\text{Isolated}}(T_j)$  is obviously held. At last, the global time cost of  $T_i$  and  $T_j$ 's executions initiated by  $a_{i-1}$  and  $a_{i-2}$  (i.e.,  $T_{\text{TC}(i-j)}(KC_{i-j})$ ) could be calculated according to Definition10, i.e.,  $T_{\text{TC}(i-j)}(KC_{i-j}) = T_{\text{End-Latest}}(T_j(KC_{i-j})) - T_{\text{Start}}(T_i) = 15$  time units.

Here, please note that two typical situations should be taken into consideration: 1)  $t_{j-x}(a_{j-x})$  is promoted by a number of task executions contained in  $T_i$ 's execution domain, and 2)  $t_{i-x}(a_{i-x})$  serves a number of task executions contained in  $T_j$ 's execution domain. The first situation is promoted with an AND-Join() style of knowledge coordination, and the second situation is promoted with an AND-Split() style of knowledge coordination between  $T_i$  and  $T_j$ . In our example as demonstrated by Fig.8, the first situation equals to an AND-Join( $t_{i-1}(a_{i-1})$ ,  $t_{i-2}(a_{i-2})$ ;  $t_{j-1}(a_{j-1})$ ) style of knowledge coordination with a hypothesis that  $t_{j-1}(a_{j-1}) = t_{j-2}(a_{j-2})$  is held; the second situation equals to an AND-Split( $t_{i-1}(a_{i-1})$ ;  $t_{j-1}(a_{j-1})$ ,  $t_{j-2}(a_{j-2})$ ) style of knowledge coordination with a hypothesis that  $t_{i-1}(a_{i-1}) = t_{i-2}(a_{i-2})$  is held. For these two application situations,  $T_j$ 's runtime temporal states could also be deduced according to *FollowMe* temporal reasoning rule. Here, limited by the length of the paper, we don't demonstrate these application situations in detail. Interested readers could calculate  $T_i$ 's temporal parameters by themselves for verifying our method.

### Case3: Temporal reasoning among three task domains

Here, we will consider a more complex situation of cross-domain collaboration among three task domain  $T_i$ ,  $T_j$  and  $T_k$  that collaborate in a sequential way. Fig.9 illustrates four typical knowledge transferring path from  $T_i$  to  $T_k$  through  $T_j$ , i.e., 1:1 knowledge transferring style, 1: $n$  knowledge transferring style,  $n$ :1 knowledge transferring style, and  $n_1$ : $n_2$  knowledge transferring style. Here,  $n$ ,  $n_1$  and  $n_2$  respectively indicate that there are  $n$ ,  $n_1$  and  $n_2$  process parameters engaged in cross-domain knowledge coordination between two task domains.

#### 1) Temporal reasoning of 1:1 knowledge transferring style

For a single learning path between  $T_i$  and  $T_j$ , there is just one process parameter  $a_{i-x}$  engaged in  $T_i$  and  $T_j$ 's knowledge coordination, which is associated with a unique temporal-dependency relation. Corresponding to  $T_i$  and  $T_j$ 's temporal-dependency relation,  $T_j$ 's runtime temporal state could be deduced for orchestrating  $T_i$  and  $T_j$ 's collaboration in an incorporated executive environment. It has been demonstrated in *Case1*'s analysis. Similarly, as there is also one process parameter  $a_{j-y}$  engaged in  $T_j$  and  $T_k$ 's cross-domain knowledge coordination,  $T_k$ 's runtime temporal state could also be deduced according to  $T_j$ 's runtime temporal state in  $T_i$  and  $T_j$ 's incorporated executive environment. Once the temporal distributions among  $T_i$ ,  $T_j$  and  $T_k$  are deduced out, the global time cost of  $T_i$ ,  $T_j$  and  $T_k$ 's executions could be calculated to evaluate their cross-domain collaboration in time cost. More specifically, let  $T_{\text{Start}} = \min\{T_{\text{Start}}(T_i), T_{\text{Start}}(T_j(\sim a_{i-x})), T_{\text{Start}}(T_k(\sim a_{j-y}))\}$ ,  $T_{\text{End}} = \max\{T_{\text{End}}(T_i), T_{\text{End}}(T_j(\sim a_{i-x})), T_{\text{End}}(T_k(\sim a_{j-y}))\}$ , the global time cost of  $T_i$ ,  $T_j$  and  $T_k$ 's executions, i.e.,  $T_{\text{TC}(i-j-k)}(a_{i-x}/a_{j-y})$ , could be calculated according to Definition10, i.e.,  $T_{\text{TC}(i-j-k)}(a_{i-x}/a_{j-y}) = T_{\text{End}} - T_{\text{Start}}$ , in  $T_i$ ,  $T_j$  and  $T_k$ 's incorporated executive environment.

## 2) Temporal reasoning of 1:n knowledge transferring style

Here, there is just one process parameter  $a_{i-x}$  engaged in  $T_i$  and  $T_j$ 's knowledge coordination. According to the temporal-dependency relation imposed on  $t_{i-x}(a_{i-x})$  and  $t_{j-x}(\sim a_{i-x})$ ,  $T_j$ 's runtime temporal state could be deduced, as we demonstrated in **Case1**'s analysis, in  $T_i$  and  $T_j$ 's incorporated executive environment. Besides, as there are  $n$  process parameters  $a_{j-1}, a_{j-2}, \dots, a_{j-n}$  engaged in  $T_j$  and  $T_k$ 's knowledge coordination, i.e.,  $A_{j-k} = \{a_{j-1}, a_{j-2}, \dots, a_{j-n}\}$ ,  $T_k$ 's  $n$  runtime temporal states could be deduced according to  $T_j$ 's runtime temporal state, as we demonstrated in **Case2**'s analysis. For  $T_k$ 's  $n$  runtime temporal states, we could determined  $T_k$ 's active interval to collaborate with  $T_j$  for their cross-domain knowledge coordination's initiated by the  $n$  process parameters  $a_{j-1}, a_{j-2}, \dots, a_{j-n}$ . Moreover, let  $T_{Start} = \min\{T_{Start}(T_i), T_{Start}(T_j(\sim a_{i-x})), T_{Start-Earliest}(T_k(KC_{j-k}))\}$ ,  $T_{End} = \max\{T_{End}(T_i), T_{End}(T_j(\sim a_{i-x})), T_{End-Latest}(T_k(KC_{j-k}))\}$ , the global time cost of  $T_i$ ,  $T_j$  and  $T_k$ 's executions, i.e.,  $T_{TC(i-j-k)}(a_{i-x}/A_{j-k})$ , could be calculated according to Definition10, i.e.,  $T_{TC(i-j-k)}(a_{i-x}/A_{j-k}) = T_{End} - T_{Start}$ .

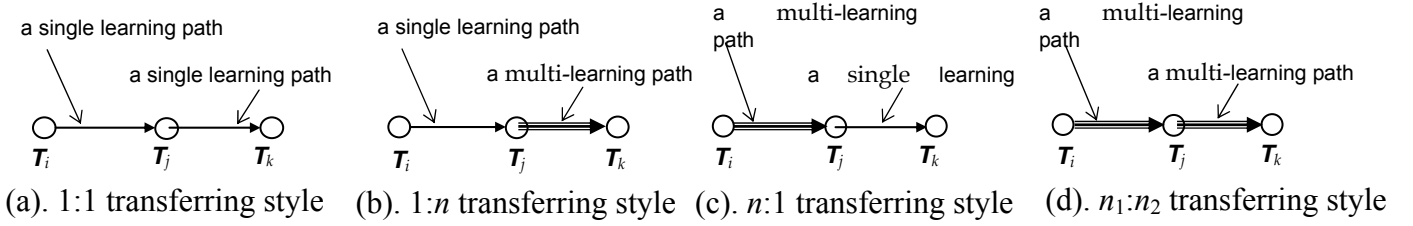


Fig.9 Knowledge transferring styles from  $T_i$  to  $T_k$  through  $T_j$ .

## 3) Temporal reasoning of n:1 knowledge transferring style

Here, there are  $n$  process parameters  $a_{i-1}, a_{i-2}, \dots, a_{i-n}$  engaged in  $T_i$  and  $T_j$ 's knowledge coordination, i.e.,  $A_{i-j} = \{a_{i-1}, a_{i-2}, \dots, a_{i-n}\}$ . For each process parameter, there is a temporal-dependency relation between  $t_{i-x}(a_{i-x})$  and  $t_{j-x}(\sim a_{i-x})$ 's executions, where  $a_{i-x} \in A_{i-j}$ . Associated with these  $n$  process parameters,  $T_j$ 's  $n$  runtime temporal states could be deduced out, as we demonstrated in **Case2**'s analysis, in  $T_i$  and  $T_j$ 's incorporated executive environment. Besides,  $T_j$ 's  $T_{Start-Earliest}(T_j(KC_{i-j}))$  and  $T_{End-Latest}(T_j(KC_{i-j}))$  could also be deduced out. For  $T_j$ 's  $n$  runtime temporal states, which one should be selected for deducing  $T_k$ 's runtime temporal state? In this paper, two selecting strategies are proposed as follows.

I) Let  $a_{j-y}$  be a process parameter engaged in  $T_j$  and  $T_k$ 's knowledge coordination. As we prescribed in Definition7's interpretations,  $t_{j-1}(\sim a_{i-1}) = t_{j-1}(a_{j-1})$ ,  $t_{j-2}(\sim a_{i-2}) = t_{j-2}(a_{j-2})$ ,  $\dots$ , and  $t_{j-n}(\sim a_{i-n}) = t_{j-n}(a_{j-n})$  are held in executive logic and active interval. With these hypotheses,  $t_{j-y}(a_{j-y}) \in \{t_{j-1}(\sim a_{i-1}), t_{j-2}(\sim a_{i-2}), \dots, t_{j-n}(\sim a_{i-n})\}$  is used to indicate that  $t_{j-y}(a_{j-y})$ 's execution is uniquely promoted by one execution of  $t_{j-1}(\sim a_{i-1})$ ,  $t_{j-2}(\sim a_{i-2})$ ,  $\dots$ , and  $t_{j-n}(\sim a_{i-n})$ . In this paper, it would be treated as a **direct knowledge transferring process** from  $T_i$  to  $T_k$  through  $T_j$ . Here, let  $t_{j-x}(\sim a_{i-x})$  be the unique knowledge consuming process for promoting  $t_{j-y}(a_{j-y})$ 's execution, where  $a_{i-x} \in \{a_{i-1}, a_{i-2}, \dots, a_{i-n}\}$ . Accordingly,  $T_j(\sim a_{i-x})$  should be selected as a referred temporal state for calculating  $T_k$ 's runtime temporal state in  $T_i$ ,  $T_j$  and  $T_k$ 's incorporated executive environment. Fig.10 illustrates a **direct knowledge transferring process** from  $T_i$ ,  $T_j$  and  $T_k$  that is associated with Before( $t_{i-1}(a_{i-1})$ ,  $t_{j-1}(\sim a_{i-1})$ ;  $T_{i-j}=1$ ) and Meet( $t_{j-1}(a_{j-1})$ ,  $t_{k-1}(\sim a_{j-1})$ ) style of temporal-dependency relations.



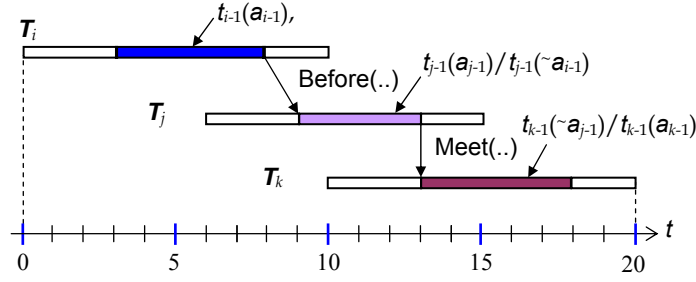


Fig.10 A direct knowledge transferring process from  $T_i$  to  $T_k$  through  $T_j$

II) Let  $a_{j-y}$  be a process parameter engaged in  $T_j$  and  $T_k$ 's knowledge coordination. If  $t_{j-y}(a_{j-y}) \notin \{t_{j-1}(\sim a_{i-1}), t_{j-2}(\sim a_{i-2}), \dots, t_{j-n}(\sim a_{i-n})\}$ , we believe that there is an **indirect knowledge transferring process** from  $T_i$  to  $T_k$  through  $T_j$ . Here, two more fine-granular situations are taken into consideration:

(1) Here,  $t_{j-y}(a_{j-y})$ 's execution interval has been specified in  $T_j$ 's isolated executive environment. If  $t_{j-y}(a_{j-y})$ 's runtime state (i.e.,  $t_{j-y}(a_{j-y})$ 's runtime execution interval  $[T_{\text{Start}}(t_{j-y}(a_{j-y})), T_{\text{End}}(t_{j-y}(a_{j-y}))]$ ) could also be determined in  $T_i$  and  $T_j$ 's incorporated executive environment,  $T_j$ 's runtime temporal state that is associated with  $[T_{\text{Start}}(t_{j-y}(a_{j-y})), T_{\text{End}}(t_{j-y}(a_{j-y}))]$  would be selected as a referred temporal state for calculating  $T_k$ 's runtime temporal state.

(2) Here,  $t_{j-y}(a_{j-y})$ 's execution interval has been specified in  $T_j$ 's isolated executive environment. If  $t_{j-y}(a_{j-y})$ 's runtime state (i.e.,  $t_{j-y}(a_{j-y})$ 's runtime execution interval  $[T_{\text{Start}}(t_{j-y}(a_{j-y})), T_{\text{End}}(t_{j-y}(a_{j-y}))]$ ) could not be determined at runtime in  $T_i$  and  $T_j$ 's incorporated executive environment, we would take another selecting rule. In this situation, as  $[T_{\text{Start}}(t_{j-y}(a_{j-y})), T_{\text{End}}(t_{j-y}(a_{j-y}))]$  is covered by time period  $[T_{\text{Start-Earliest}}(T_j(KC_{i-j})), T_{\text{End-Latest}}(T_j(KC_{i-j}))]$ ,  $T_j$ 's runtime temporal state whose end time equals to  $T_{\text{End-Latest}}(T_j(KC_{i-j}))$  would be selected as a referred temporal state for calculating  $T_k$ 's runtime temporal state in  $T_i$ ,  $T_j$  and  $T_k$ 's incorporated executive environment. It guarantees that the required knowledge assets for promoting  $t_{k-y}(a_{k-y})$ 's execution could be achieved before  $t_{k-y}(a_{k-y})$ 's execution.

For example, suppose that  $T_i$  and  $T_j$ 's temporal-dependency relation is illustrated by Fig.8, in which  $T_j$  has two runtime temporal states respectively associated with two process parameters of  $a_{i-1}$  and  $a_{i-2}$ . Now, a third task execution  $T_k$  is taken into consideration as illustrated by Fig.11. Let  $a_{j-y}$  be a process parameter engaged in  $T_j$  and  $T_k$ 's Meet( $t_{j-y}(a_{j-y}), t_{k-y}(\sim a_{j-y})$ ) style of knowledge coordination.

(1) In the situation that  $t_{j-y}(a_{j-y}) \in \{t_{j-1}(\sim a_{i-1}), t_{j-2}(\sim a_{i-2})\}$ , if  $t_{j-y}(a_{j-y}) = t_{j-1}(\sim a_{i-1})$ ,  $T_j(\sim a_{i-1})$  would be selected as a referred temporal state for calculating  $T_k$ 's runtime temporal state; if  $t_{j-y}(a_{j-y}) = t_{j-2}(\sim a_{i-2})$ ,  $T_j(\sim a_{i-2})$  would be selected as a referred temporal state for calculating  $T_k$ 's runtime temporal state.

(2) If  $t_{j-y}(a_{j-y}) \notin \{t_{j-1}(\sim a_{i-1}), t_{j-2}(\sim a_{i-2})\}$ , but we could determined the runtime state of  $[T_{\text{Start}}(t_{j-y}(a_{j-y})), T_{\text{End}}(t_{j-y}(a_{j-y}))]$  in  $T_i$  and  $T_j$ 's incorporated executive environment,  $T_j$ 's runtime temporal state that is associated with  $[T_{\text{Start}}(t_{j-y}(a_{j-y})), T_{\text{End}}(t_{j-y}(a_{j-y}))]$  would be selected as a referred temporal state for calculating  $T_k$ 's runtime temporal state. More specifically, as illustrated in Fig.11, if we could determine that the runtime state of  $[T_{\text{Start}}(t_{j-y}(a_{j-y})), T_{\text{End}}(t_{j-y}(a_{j-y}))]$  is covered by  $T_j(\sim a_{i-2})$ 's executive interval,  $T_j(\sim a_{i-2})$  would be selected as a referred temporal state for calculating  $T_k$ 's runtime temporal state. It guarantee that  $t_{k-y}(a_{k-y})$  could timely achieve its referred knowledge from  $t_{j-y}(a_{j-y})$ 's execution.

(3) If  $t_{j-y}(a_{j-y}) \notin \{t_{j-1}(\sim a_{i-1}), t_{j-2}(\sim a_{i-2})\}$  and  $[T_{\text{Start}}(t_{j-y}(a_{j-y})), T_{\text{End}}(t_{j-y}(a_{j-y}))]$  could not be determined at runtime,  $T_j(\sim a_{i-1})$  would be selected as a referred temporal state for calculating  $T_k$ 's runtime temporal state, as its end time equals to  $T_{\text{End-Latest}}(T_j(KC_{i-j}))$ . It guarantees that the required

knowledge assets for promoting  $t_{k-y}(a_{k-y})$ 's execution could be achieved before  $t_{k-y}(a_{k-y})$ 's execution.

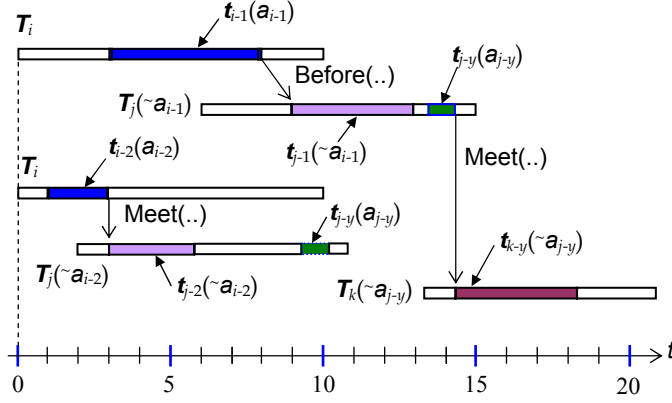


Fig.11 An *indirect knowledge transferring process* from  $T_i$  to  $T_k$  through  $T_j$ .

Once  $T_k$ 's runtime temporal states are determined, we could calculate  $T_{TC(i-j-k)}(A_{i-j}/A_{j-k})$ 's value according to Definition10 for evaluating their cross-domain collaboration in time cost.

#### 4) Temporal reasoning of $n_1:n_2$ knowledge transferring style

Suppose that there are  $n_1$  process parameters  $a_{i-1}, a_{i-2}, \dots, a_{i-n1}$ , i.e.,  $A_{i-j} = \{a_{i-1}, a_{i-2}, \dots, a_{i-n1}\}$  for  $T_i$  and  $T_j$ 's knowledge coordination. As we demonstrated in **Case2**'s analysis,  $T_j$ 's  $n_1$  runtime temporal states could be determined in  $T_i$  and  $T_j$ 's incorporated executive environment. Suppose that there are  $n_2$  process parameters  $a_{j-1}, a_{j-2}, \dots, a_{j-n2}$ , i.e.,  $A_{j-k} = \{a_{j-1}, a_{j-2}, \dots, a_{j-n2}\}$  for  $T_j$  and  $T_k$ 's knowledge coordination. Here, we focus on how to deduce  $T_k$ 's runtime temporal states in  $T_i, T_j$  and  $T_k$ 's incorporated executive environment.

In our method, the  $n_2$  process parameters engaged in  $T_j$  and  $T_k$ 's knowledge coordination are firstly classified into two groups (i.e., *Group-1*, *Group-2*). *Group-1* consists of the process parameters, each of which is engaged in a **direct knowledge transferring process** from  $T_i$  to  $T_k$  through  $T_j$  as we discussed in  $n:1$  knowledge transferring style. For brevity and without the loss of generality, let *Group-1* consist of process parameters  $a_{j-1}, a_{j-2}, \dots, a_{j-m}$ , i.e.,  $Group-1 = \{a_{j-1}, a_{j-2}, \dots, a_{j-m}\}$ , where  $m \leq n_2$ . *Group-2* consists of the process parameters, each of which is engaged in an **indirect knowledge transferring process** from  $T_i$  to  $T_k$  through  $T_j$  as we discussed in  $n:1$  knowledge transferring style, i.e.,  $Group-2 = \{a_{j-(m+1)}, a_{j-(m+2)}, \dots, a_{j-n2}\}$ . Here,  $Group-1 \cap Group-2 = \emptyset$  and  $Group-1 \cup Group-2 = A_{j-k}$ .

For the  $m$  process parameters contained in *Group-1*,  $T_k$ 's  $m$  runtime temporal states could be determined according to the first selected strategy presented in  $n:1$  knowledge transferring style's analysis. Furthermore, associated with the  $m$  runtime temporal states, a group of time period  $[T_{Start}(T_k(a_{j-1})), T_{End}(T_k(a_{j-1}))], [T_{Start}(T_k(a_{j-2})), T_{End}(T_k(a_{j-2}))], \dots, [T_{Start}(T_k(a_{j-m})), T_{End}(T_k(a_{j-m}))]$  could be determined. Similarly, for the  $n_2-m$  process parameters contained in *Group-2*,  $T_k$ 's  $n_2-m$  runtime temporal states could also be determined as we discussed in the second selected strategy presented in  $n:1$  knowledge transferring style's analysis. Furthermore, associated with the  $n_2-m$  runtime temporal states, a group of time period  $[T_{Start}(T_k(a_{j-(m+1)})), T_{End}(T_k(a_{j-(m+1)}))], [T_{Start}(T_k(a_{j-(m+2)})), T_{End}(T_k(a_{j-(m+2)}))], \dots, [T_{Start}(T_k(a_{j-n2})), T_{End}(T_k(a_{j-n2}))]$  could be determined. Let  $T_{Start-Earliest}(T_k(KC_{j-k})) = \min\{T_{Start}(T_k(a_{j-1})), T_{Start}(T_k(a_{j-2})), \dots, T_{Start}(T_k(a_{j-m})), T_{Start}(T_k(a_{j-(m+1)})), \dots, T_{Start}(T_k(a_{j-n2}))\}$ , let  $T_{End-Latest}(T_j(KC_{i-j})) = \max\{T_{End}(T_k(a_{j-1})), T_{End}(T_k(a_{j-2})), \dots, T_{End}(T_k(a_{j-m})), T_{End}(T_k(a_{j-(m+1)})), \dots, T_{End}(T_k(a_{j-n2}))\}$ ,  $T_{Collaboration}(T_k(KC_{j-k}))$ 's value could be calculated out, i.e.,  $T_{Collaboration}(T_k(KC_{j-k})) = T_{End-Latest}(T_j(KC_{i-j})) - T_{Start-Earliest}(T_k(KC_{j-k}))$ . Moreover, once

$T_k$ 's runtime temporal states are determined, we could calculate  $T_{TC(i-j-k)}(A_{i-j}/A_{j-k})$ 's value according to Definition10 for evaluating their cross-domain collaboration in time cost.

In practice, time checking is an important issue in an administrable system [22][23][24]. **Case1** and **Case2**'s analyses cover nearly all the possible temporal-dependency situations engaged in collaboration between two task domains. **Case3** covers nearly all the possible temporal-dependency situations engaged in collaboration among three task domains. They provide basic temporal-dependency situations for more complex temporal analysis in cross-domain collaboration.

The temporal knowledge modeling and the temporal reasoning logic presented in this section could be illustrated by Fig.12, which is essentially an application of linear reasoning process corresponding to *Markov* model or *Markov* chain [51]. More specifically, for two task domains engaged in certain collaboration, their cross-domain temporal modeling is often initiated by a pioneer task domain's knowledge modeling. Only after a pioneer task domain's temporal specification is determined, the succeeding task's temporal modeling could be unfolded in a serial way.

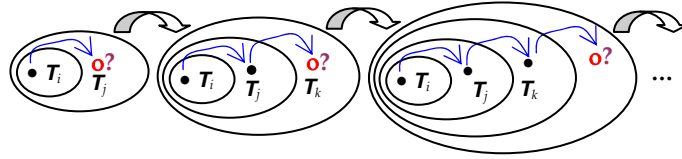


Fig.12. Application logic for temporal modeling and temporal reasoning between two task domains for their cross-domain collaboration.

Here, the time complexity of the linear reasoning process illustrated by Fig.12 would be analyzed to evaluate the feasibility of our proposal. Let Set-TE be a set of task domains engaged in global collaborative problem solving, and  $N_{i-j}$  be a set of process parameter  $a_{i-j}$  engaged in  $T_i$  and  $T_j$ 's collaboration. Besides, let  $|\text{Set-TE}|=m$ , and  $|N_{i-j}|=n_{i-j}$ . For calculating global time cost, we need to calculate local time cost between two collaborative task domains, from  $T_1$  to  $T_m$ , for  $m-1$  times in a hierarchical way. For task domain  $T_i$  and  $T_j$ , the time cost is  $O(n_{i-j})$ . So, the time complexity for calculating the global time cost is  $O(n_{1-2}+n_{2-3}+\dots+n_{(m-1)-m})$ .

## Related Works and Comparison Analysis

In past decades, problem solving environment(PSE) [14][27], knowledge modeling framework [9][15][50], method discovery [9][11][34], and task-specific or domain-specific scheduling applications [4][9][24][38][50] have been investigated in collaborative problem solving domain. Compared to these existing research works, the main contributions of this paper are summarized from the following aspects.

(1) About knowledge modeling framework development, CommonKADS is enacted based on systematic separation between task, method, and domain [9][18]. By introducing an “application” component into its definition, a knowledge modeling framework, named TMDA, are introduced in [9][50]. As mentioned in [9][50], TMDA is constructed based on a Task-Method-Domain-Application application logic. Compared to CommonKADS and TMDA knowledge modeling framework, **POTMe** framework consists of four typical components of problem, ontology, task, and method that are coordinated by a Problem-Ontology-Task-Method application logic. It provides a well-structured knowledge association context for problem cognition and solving. Furthermore, in TMDA framework presented in [9][50], a task is often pre-specified without definite problem context. Their researches focus on how to find efficient methods for a task implementation. In our research, a task execution owns an explicit application context specified by cer-

tain knowledge-dependency and temporal-dependency definitions for cross-domain collaboration. Here, once a task is definitely specified in an application context, an existing task-specific method library (e.g., method library presented in [9]) would be helpful for promoting task execution through method discovery. Therefore, Task and Method objects are two knowledge components shared by *POTMe*'s application logic and TMDA's application logic.  $\{\text{Problem, Ontology, Task, Method}\} \cup \{\text{Task, Method, Domain, Application}\} = \{\text{Problem, Ontology, Task, Method, Domain, Application}\}$  covers nearly all knowledge components engaged in collaborative problem solving. They could satisfy more complex knowledge modeling requirement with a combined application logic of Problem-Ontology-Task-Method-Domain-Application.

(2) As mentioned in [9], "scheduling deals with the assignment of jobs and activities to resources and time ranges in accordance with relevant constraints and requirements." Some scheduling methods, such as time-related Petri nets (e.g., Time Petri net, Timed Petri net and Timing Petri net), have been presented for evaluating task enactments and executions [24]. In this paper, *POTMe*'s knowledge-based temporal modeling and its application is reified into a concrete temporal reasoning rule, i.e., *FollowMe* temporal reasoning rule, for enhancing the validity of task enactments from scheduling aspect. It provides a method for evaluating global scheduling application, through simulating cross-domain collaboration in an incorporated executive environment. Compared to Petri net-based scheduling methods, time complexity of our approach is degraded in a limited scope. The time complexity of the reasoning process for global joint time is  $O(n_{1.2} + n_{2.3} + \dots + n_{(m-1)-m})$ . For Petri nets, we should use  $m$  places to represent  $m$  meta-cognitions respectively. The capacity of each place  $m_i$  is  $n_{i-j}$ . Hence, there are  $n_{1.2} \times n_{2.3} \times \dots \times n_{(m-1)-m}$  reasoning steps and the time complexity is  $O(n_{1.2} \times n_{2.3} \times \dots \times n_{(m-1)-m})$ . Obviously, the time complexity of our approach is smaller than the time complexity of time-related Petri nets, which is helpful for receding state explosion problem as mentioned in [51].

Here, the temporal reasoning process is enacted among distributed process fragments, uncertainty issues or some exceptions at runtime may degrade the generic nature of our methods in practice. The methods presented in [21][37][41][42][43] are helpful for dynamically dealing with uncertainty issues or exceptions at runtime. Another limitation of our approach is that our work just supports unilateral learning-compliant knowledge dependency and temporal dependency. If there are some bidirectional knowledge dependency and temporal dependency, the approaches presented in this paper should be perfected or modified to deal with the complex application requirements. In our future research, we would centralize on exploring a more complex situation with bidirectional knowledge and temporal-dependency relation based on the approaches presented in this paper.

## Conclusions and Future Work

In this paper, we investigated a novel knowledge modeling framework, named *POTMe*, for enabling collaborative problem solving. *POTMe*'s knowledge-based temporal modeling and its application are reified by a concrete temporal reasoning rule, i.e., *FollowMe* temporal reasoning rule, for enhancing the validity of task enactments from scheduling aspect. It provides a method for evaluating global scheduling applications, through simulating cross-domain collaboration in an embedded executive environment. The scenarios are validated by an industrial case study. Although our methods have been put into practice in a collaborative product development, more real-life and benchmark applications are under way as our future research.

## References

- [1] Hai Zhuge and Xiaoqing Shi, "Communication cost of cognitive co-operation for distributed team development," *Journal of Systems and Software*, vol.57, no.3, pp.227-233, 2001.
- [2] C.Weber, "Knowledge transfer and the limits to profitability: an empirical study of problem-solving practices in semiconductor manufacturing and process development," *IEEE Trans. Semiconductor manufacturing*, vol.15, no.14, pp.420-426, 2002.
- [3] P.N.Robillard, "Opportunistic problem solving in software engineering," *IEEE Software*, vol.22, no.6, pp. 60-67, 2005.
- [4] Kawata, S. , Fuju, H. ; Sugiura, H. , Saitoh, Y. , Hayase, Y. , Teramoto, T. , Kikuchi, T, "A distributed problem solving environment (PSE) for scientific computing," *e-Science and Grid Computing*, First International Conference on ,2005.
- [5] T.Cohen, B.Blatter, C.Almeida, E.H.Shortliffe, and V.L.Patel, "A cognitive blueprint of collaboration in context: distributed cognition in the psychiatric emergency department," *Artificial Intelligence in Medicine*, vol.37, no.2, pp.73-83, 2006.
- [6] Pek Hui Foo , Gee Wah Ng , Khin Hua Ng , Rong Yang , "Application of intent inference for surveillance and conformance monitoring to aid human cognition," *Information Fusion*, 2007 10th International Conference on Digital Object Identifier,pp1-8,2007.
- [7] Jonassen, David, "Using Cognitive Tools to Represent Problems," *Journal of Research on Technology in Education (International Society for Technology in Education)* Vol. 35 Issue 3, 2003.
- [8] Stevens, R., Soller, A., Giordani, A., Gerosa, L., Cooper, M., Cox, C.: "Developing a framework for integrating prior problem solving and knowledge sharing histories of a group to predict future group performance," *Collaborative Computing: Networking, Applications and Worksharing*, 2005 International Conference on.
- [9] D.Rajpathak, E.Motta, Z.Zdrahal, and R.Roy, "A generic library of problem solving methods for scheduling applications," *IEEE Trans. Knowledge and Data Engineering*, vol.18, no.6, pp.815-828, 2006.
- [10] Lumala, A.F.N. , Quenum, J.G, "A Distributed Problem Solving Approach for Service-Oriented Computing Systems," *Services - I,World Conference on*, pp.530-538, 2009.
- [11]D.Fensel and E.Motta, "Structured development of problem solving methods," *IEEE Trans. Knowledge and Data Engineering*, vol.13, no.6, pp.913-932, 2001.
- [12]U.Deshpande, A.Gupta, and A.Basu, "Coordinated problem solving through resource sharing in a distributed environment," *IEEE Trans. Systems, Man, and Cyber. (Part B)*, vol.34, no.2, pp.1299-1304, 2004.
- [13]Edmund H. Durfee, "Distributed Problem Solving and Planning," *Multi-Agent Systems and Applications Lecture Notes in Computer Science Volume 2086*, pp.118-149, 2001.
- [14]A Willem, M Buelens, "Knowledge sharing in inter-unit cooperative episodes: The impact of organizational structure dimensions," *International Journal of Information Management, Volume 29, Issue 2* pp.151-160, 2009.

- [15]Jan M. van Bruggen, Henny P. A. Boshuizen, Paul A. Kirschner, "A Cognitive Framework for Cooperative Problem Solving with Argument Visualization," *Visualizing Argumentation Computer Supported Cooperative Work*, pp.25-47, 2003.
- [16]Majedi, M.R. , Osman, K.A. , Wilcox, A.J., "A Generic Task Partitioning Framework for Internet Based Control of Multiple Co-operatively Working Robotic Devices," *Pervasive Computing and Applications, 2007. ICPCA 2007. 2nd International Conference on*, pp.451-456.
- [17]J Blythe, Y Gil, "Extending the role-limiting approach: Supporting end users to acquire problem-solving knowledge," *ECAI Workshop on Applications of Ontologies and Problem-Solving Methods*, 2000.
- [18]Quynh-Nhu Numi Tran, Henderson-Sellers, B. , Debenham, J. , Gonzalez-Perez, C., "Conceptual modelling within the MAS-CommonKADS plus OPEN method engineering approach," *Information Technology and Applications, 2005. ICITA 2005. Third International Conference on (Volume:1 )*, pp.29-34
- [19]Simon M. Becker, Thomas Haase, Bernhard Westfechtel, "Model-based a-posteriori integration of engineering tools for incremental development processes," *Software & Systems ModelingMay, Volume 4, Issue 2* pp.123-140, 2005.
- [20]Kent, P. , Noss, R., "The mathematical components of engineering expertise: the relationship between doing and understanding mathematics," *Engineering Education 2002: Professional Engineering Scenarios (Ref. No. 2002/056), IEE (Volume:2 )*.
- [21]E.Vicario, "Static analysis and dynamic steering of time-dependent systems," *IEEE Trans. Software Eng.*, vol.27, no.8, pp.728-748, 2001.
- [22]Ma, Jixin , Knight, B. , Petridis, Miltos , Xiao Bai, "A Graphical Representation for Uncertain and Incomplete Temporal Knowledge," *Intelligent Systems (GCIS), Second WRI Global Congress on (Volume:1 )*, 2010.
- [23]A Krokhin, P Jeavons, P Jonsson, "Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra," *Journal of the ACM (JACM) JACM Homepage archiveVolume 50 Issue 5*, pp.591-640, 2003.
- [24]J.Q. Li, Y.S.Fan, M.C.Zhou, "Timing constraint workflow nets for workflow analysis," *IEEE Trans. Systems, Man, and Cyber. (Part A)*, vol.33, no.2, pp. 179-193, 2003.
- [25]R.Sun, E.Merrill, T.Peterson, "From implicit skills to explicit knowledge: a bottom-up model of skill learning," *Cognitive Science*, vol.25, no.2, pp.203-364, 2001.
- [26]Ikujiro Nonaka, Ryoko Toyama, Noboru Konno, "SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation," *Long Range Planning Volume 33, Issue 1*, pp.5-34, 2000.
- [27]Y.S.Ong, and A.J.Keane, "A domain knowledge based search advisor for design problem solving environments," *Engineering Applications of Artificial Intelligence*, vol.15, no.1, pp.105-116, 2002.
- [28]D.RICHARDS and S.SIMOFF, "Design ontology in context-a situated cognition approach to conceptual modeling," *Artificial Intelligence in Engineering*, vol.15, no.2, pp.121-136, 2001.
- [29]Al-Khatib, W. , Day, Y.F. , Ghafoor, A. , Bruce Berra, P "Semantic modeling and knowledge representation in multimedia databases," *Knowledge and Data Engineering, IEEE Transactions*

on (Volume:11 , Issue: 1 ), pp. 64-80, 2002.

- [30]Y.Lee and W.Zhao, “An ontology-based approach for domain requirements elicitation and analysis,” *In Proc. 1st Int. Multi-Symp. Computer and Computational Sciences*, Hangzhou, China, 20-24 June, 2006, pp.364-371.
- [31]V.Khatri, I.Vessey, S.RamAM, and V.Ramesh, “Cognitive fit between conceptual schemas and internal problem representations:the case of geospatio–temporal conceptual schema comprehension,” *IEEE Trans. Professional Communication*, vol.9, no.2, pp.109-127, 2006.
- [32]S.M.Lucas and T.J.Reynolds, “Learning finite-state transducers: evolution versus heuristic state merging,” *IEEE Trans. Evolutionary Computation*, vol.11, no.3, pp.308-325, 2007.
- [33]C.L. Storto, “The fit between problem solving style and perceived problem complexity as a major determinant of knowledge generation during product innovation: empirical evidence and implications for a theory of learning,” *Int. Conf. Management of Engineering and Technology*, Portland, 29 July-2 Aug., 2001, pp.291 --302.
- [34]Pathak J, Johnson, Thomas M, Chute, C.G., “Modular ontology techniques and their applications in the biomedical domain,” *Information Reuse and Integration, IEEE*, pp.351-356, 2008.
- [35]Frederico T. Fonseca, Max J. Egenhofer, Peggy Agouris and Gilberto Câmara, “Using Ontologies for Integrated Geographic Information Systems,” *GIS Volume 6, Issue 3*, pp. 231–257, June 2002
- [36] E Motta, S Buckingham Shum, J Domingue, “Ontology-driven document enrichment: principles, tools and applications,” *International Journal of Human-Computer Studies*. Volume 52, Issue 6, pp. 1071–1109, 2000.
- [37]J.S.Gero, “Computational models of innovative and creative design processes,” *Technological Forecasting and Social Change*, vol.9, no.2, pp.183-196, 2000.
- [38]Liane Gabora, “Cognitive mechanisms underlying the creative process, ” *Proceeding C&C '02 Proceedings of the 4th conference on Creativity & cognition*, ACM New York, NY, USA pp.126-133,2002.
- [39]Y.Reich and A.Kapeliuk, “A framework for organizing the space of decision problems with application to solving subjective, context-dependent problems,” *Decision Support Systems*, vol.41, no.1, pp.1-19, 2005.
- [40]David Corsar, Derek Sleeman, “KBS development through ontology mapping and ontology driven acquisition,” *Proceeding K-CAP '07 Proceedings of the 4th international conference on Knowledge capture* , ACM New York, NY, USA pp.23-30, 2007.
- [41]Kar-Ming Cheung, A. Ko, V. Dang, D. Heckman, “Risk analysis for nondeterministic mission planning and sequencing,” *California Inst. of Technol., Pasadena, CA, USAIn proceeding of: Aerospace Conference, IEEE*, 2005.
- [42]L Marengo, G Dosi, “Division of labor, organizational coordination and market mechanisms in collective problem-solving,” *Journal of Economic Behavior & Organization Volume 58, Issue 2*, pp.303-326, 2005.
- [43]Martinez, N. , Garcia, M.M. , Hurtado, J.E. , “Model for designing Intelligent Tutorials Systems using Conceptual Maps and knowledge-based Systems”, *Latin America Transactions, IEEE*

(*Revista IEEE America Latina*) (Volume:10 , Issue: 6 ), pp.2301-2308, 2012.

[44]Roddick, J.F., Spiliopoulou, M. “A survey of temporal knowledge discovery paradigms and methods,” *Knowledge and Data Engineering, IEEE Transactions on* (Volume: 14 , Issue: 4) ,2002.

[45]J.Dorr, S.Adam, M.Eisenbarth, M.Ehresmann, “Implementing requirements engineering processes: using cooperative self-assessment and improvement,” *IEEE Software*, vol.25, no.3, pp.71-77, 2008.

[46]W.M.P. van der Aalst, A.H.M.ter Hofstede, and A.P.Barros, “Workflow patterns,” *Distributed and Parallel Databases*, vol.14, no.1, pp.5–51, 2003.

[47]F.S.Hillier, and G.J.Lieberman, *Introduction to Operations Research* (8e), 2005, ISBN: 0073017795, Holden Day,Inc.Oakland, California, 2005.5.

[48]Ken Kennedy, Bradley Broome, Keith Cooper, Jack Dongarra, Rob Fowler, Dennis Gannon, Lennart Johnsson, John Mellor-Crummey, Linda Torczona, “Telescoping Languages: A Strategy for Automatic Generation of Scientific Problem-Solving Systems from Annotated Libraries,” *Journal of Parallel and Distributed Computing* Volume 61, Issue 12, pp.1803-1826, 2001.

[49]I.H.M. van Stokkum and H.E.Bal: “A problem solving environment for interactive modelling of multiway data”, *Concurrency and Computation: Practice and Experience*, vol.18, no.2, pp.263-269, 2006.

[50]E.Motta, *Reusable Components for Knowledge Modelling: Principles and Case Studies in Parametric Design*. Amsterdam: IOS Press, 1999.

[51]S Demri, F Laroussinie, P Schnoebelen, “*A parametric analysis of the state-explosion problem in model checking*” Volume 72, Issue 4, June 2006, pp. 547–575