

Towards Incompletely Specified Process Support in SwinDeW – A Peer-to-Peer Based Workflow System

Jun Yan¹, Yun Yang¹, and Gitesh K. Raikundalia^{2,1}

¹ Faculty of Information and Communication Technologies,
Swinburne University of Technology,

PO Box 218, Hawthorn, Melbourne, Australia 3122
{jyan, yyang}@it.swin.edu.au

² School of Computer Science and Mathematics,
Victoria University,

P.O. Box 14428, Melbourne City, Australia 8001
Gitesh.Raikundalia@vu.edu.au

Abstract. Due to increased complexity and flexibility of processes and lack of modelling information, workflow processes are not always defined completely before their execution. Support for incompletely specified processes which require on-the-fly articulation of processes has become a desirable feature of workflow management systems. Unfortunately, this aspect is rather weak in contemporary workflow research. This paper reports innovative research on incompletely specified process support carried out in the context of SwinDeW, a peer-to-peer based decentralised workflow system. In order to extend the SwinDeW architecture and system functions seamlessly for supporting incompletely specified processes, a hierarchical process modelling and execution approach is presented in this paper. This approach supports stepwise elaboration of incompletely-specified processes on-the-fly. Further elaboration of a process is innovatively modelled as essential steps towards the process goal, thus being scheduled to execute as ordinary tasks.

1 Introduction

Workflow Management Systems (WfMSs) provide automated support for business processes, through the use of software. Over the past decade, as a solution to business process management, workflow technology has attracted intense attention from both researchers and practitioners and experienced tremendous growth. The adoption of WfMSs has, consequently, brought direct and indirect benefits such as reduction in costs and flow times, increase of quality of service and productivity, and so on. Although workflow research and practice have reached a certain degree of maturity, some severe problems remain unsolved. More specifically, the inappropriate use of the client-server architecture which provides centralised workflow coordination, and the lack of ability to support incompletely specified processes (*incomplete processes* for short) have become two major obstacles to the wide deployment of workflow technology in the real world. As a consequence, two growing trends of workflow technology have been observed. One is to build workflow management systems in a genuinely distributed way to reflect workflow's inherently distributed feature more

naturally. The other is to develop enabling techniques and mechanisms for support of incomplete processes.

Many efforts have been devoted to addressing issues related to the first trend, from adding more distribution to client-server based workflow systems to adopting new computing technologies such as peer-to-peer (p2p) technology [2] for workflow support. In particular, the authors have conducted intensive research aiming at combining workflow and p2p technologies to offer decentralised workflow support. An innovative workflow approach, known as SwinDeW (*Swinburne Decentralised Workflow*) [12, 13], has been presented to support completely specified processes (*complete processes* for short) in a p2p environment. Regarding incomplete process support, although some preliminary work has addressed the problems initially from the process modelling perspective, system coordination support for incomplete processes has been unreasonably ignored. Moreover, to the best of the authors' knowledge, no research has been carried out in addressing these two aspects jointly.

The distinct research reported in this paper was carried out in the context of the SwinDeW project. The major focus of this paper is to address the issues of incomplete process support in a p2p-based decentralised workflow environment from a system coordination viewpoint. To enable this, the following specific requirements for SwinDeW are raised: (1) An incomplete process definition needs to be divided into task partitions and task partitions need to be distributed to and stored by relevant peers properly. An incomplete part of a process should be allowed to instantiate before execution. The workflow represented by an incomplete part of a process should be allowed to be allocated to peers; and (2) Run-time incomplete process support needs to be carried out in a decentralised environment so that on-the-fly process elaboration can be performed at the right time and the right place by the right participant. Accordingly, in this paper, an innovative hierarchical workflow modelling and execution approach is presented, which allows for incomplete process specification at build-time and on-the-fly process elaboration at run-time.

The rest of this paper is organised as follows. Section 2 introduces SwinDeW briefly, including its decentralised system design and corresponding mechanisms supporting complete processes. In Section 3, the approaches to support stepwise process modelling and execution in the SwinDeW decentralised environment are presented. Section 4 then uses a research project as a sample to illustrate the authors' key ideas. After that, major related work is introduced in Section 5. Finally, Section 6 concludes the paper with the authors' contributions and outlines the authors' future work.

2 Background

As the traditional client-server based centralised architecture has faced more and more challenges in supporting workflow systems properly, p2p-based workflow systems have been recently recognised as one of the most strategic future directions for workflow research [4]. Based on the rationale that p2p reflects decentralised workflow much more naturally than client-server, the authors have presented an innovative SwinDeW workflow approach. This approach aims at investigation of process support

technologies for decentralised workflow systems based on the p2p, rather than client-server, distributed system architecture.

As shown in Figure 1, the novel framework of SwinDeW implies the presence of neither a centralised data repository for information storage, nor a centralised workflow engine for coordination [12, 13]. The system is defined as four layers. The top layer is a set of workflow participant software (WfPS) which defines the application-oriented semantics to fulfil workflow functions. Core services of the workflow system are provided at the service layer. The data layer consists of data repositories (DRs) which store workflow-related information such as process definitions and instance information. The communication layer facilitates direct communication among workflow participants.

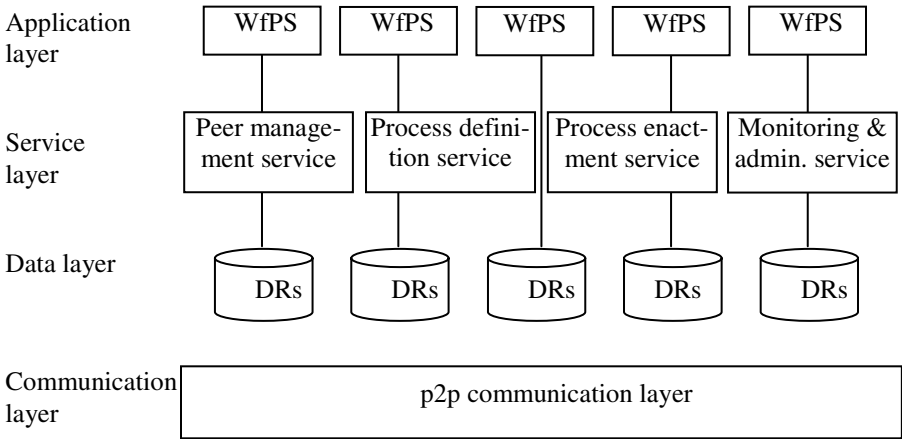


Fig. 1. Decentralised framework of SwinDeW

The basic working unit, known as a *peer*, consists of a WfPS and a set of data repositories. In most cases, each peer resides on a physical machine and is associated with a workflow participant. On behalf of the associated workflow participant, each peer is able to communicate with other peers directly to carry out functions. Given essential information and authority, a peer is a self-managing, autonomous entity whose ability to work both independently and collaboratively differentiates it from a client in the client-server architecture.

At build-time, the distinct data storage philosophy proposed in SwinDeW is named as “*know what you should know*” [12, 13], which is proposed in contrast to the conventional approaches in which each participant involved in a process knows either nothing or everything. Process definition data in SwinDeW are divided into a set of individual task partitions, each of which represents a single task in the context of the process. Thereafter, individual task partitions are distributed to relevant peers based on capability matching so that process definition data are stored in a distributed manner. By doing so, peers in the system have partial and essential knowledge, which enables them to collaborate in order to fulfil all the key functions of workflow execution.

Regarding run-time functions, SwinDeW mainly focuses on the issues of process instantiation and instance execution [12, 13]. The phase of process instantiation creates various task instances and determines performers of these task instances through peer coordination. All task instances are finally created at dispersed locations by peers actually performing them. Therefore, a process instance is represented by a network of peers performing various tasks in a certain order. Correspondingly, the execution of a process instance does not rely on a centralised engine to perform coordination. Peers communicate with one another to exchange control information and application data during process enactment. As each peer has knowledge about the task it is responsible for and its predecessor and successor peer(s), it can act independently to carry out different types of functions properly such as determining from whom they should receive notifications and data, when to start working, to whom they should pass work, and so on. In this way, the work is passed from one participant to another directly as predefined.

The above framework and mechanisms are designed to support complete processes. For demonstration and proof-of-concept purposes, a JXTA-based prototype has been implemented and the results so far are promising. The next logical step is to extend these mechanisms seamlessly for incomplete process support with unique features due to the p2p infrastructure. Some initial work of the authors is reported in [11, 12].

3 Hierarchical Process Modelling and Execution

To extend SwinDeW for supporting incomplete processes with on-the-fly task decomposition, a multi-tiered process modelling and execution approach is first presented in this section. Then, task decomposition is discussed. Finally, mechanisms for task execution are given.

3.1 Hierarchical Modelling Approach

Modelling workflow means representing work activities in relation to each other. The classical approach in this view is best represented in “Hierarchical Task Analysis” (HTA) [7], where a workflow is described as a hierarchical structure of tasks and sub-tasks. In the real world, modelling of non-trivial processes is normally not a one-off occurrence and may experience several rounds. Initially, a process is defined as a network of tasks. Some of these tasks, known as *atomic tasks*, represent the smallest executable units of work. Normally, these tasks can be specified clearly and completely. Some other tasks, on the contrary, are simply modelled as black boxes with intakes and outtakes. Although each of these tasks, known as *composite tasks*, has a pre-determined contribution to the overall process, how a task is fulfilled remains uncertain for the time being. In most cases, a composite task represents a unit of work which is fulfilled by executing a set of sub-tasks. These sub-tasks, each of which can be either atomic or composite, are also partially ordered, forming a sub-process. Thus, a process is defined at multiple levels of abstraction. The process specification at a higher level of abstraction contains composite tasks which need to be decomposed in the process definition at a lower level of abstraction. Evidently, this approach adopts a

hierarchical modelling mechanism. Further process modelling work is to convert composite tasks into sub-processes. Hence, process modelling is carried out in a stepwise manner. Figure 2 depicts this top-down hierarchical modelling approach.

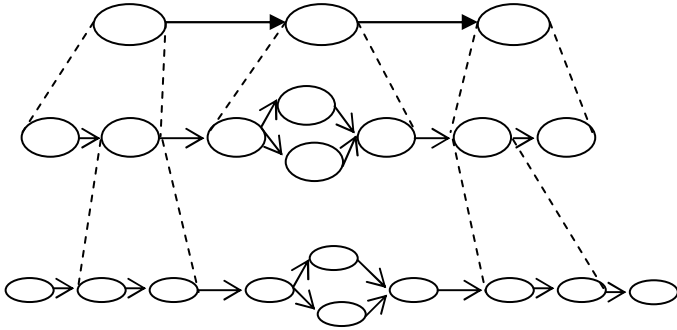


Fig. 2. The hierarchical modelling model

This approach rationalises the modelling work and reflects human's behaviours naturally, especially when the process is complex and the modelling information is insufficient. In addition, this approach also provides multiple-level process reusability because multiple rounds of modelling generate multiple, reusable, model fragment drafts. A process model at a higher level of abstraction can be reused to generate various process templates at a lower level of abstraction. This feature allows the workflow system to deal with flexible workflow processes. Process instances with variations can be derived from the same process model at a certain level of abstraction and inherit some common features of the model.

3.2 Task Decomposition

To extend SwinDeW, a special managerial task, known as a *decomposition task*, is designed. Each decomposition task is associated with a composite task and in charge of the decomposition of this task. Figure 3 shows a decomposition task (*Decom*) and its position in the process where *Pre* and *Succ* indicate the preceding and succeeding tasks. The associated composite task, $Compo(T_n)$, in Figure 3 is finally decomposed into a sub-process which consists of sub-tasks $t_{n,1}$, $t_{n,2}$, $t_{n,3}$ and $t_{n,4}$. The textual description of a decomposition task is as follows:

- *Description*: A decomposition task decomposes the associated composite task at run-time into a sub-process consisting of a set of partially ordered sub-tasks, each of which can be either atomic or composite.
- *Responsibility*: A decomposition task is carried out by an authorised person with special skills to model processes, such as a process engineer or a project manager.
- *Inputs*: The inputs of a decomposition task are very flexible. Normally, a decomposition task may take the outputs of the preceding tasks of the associated composite task, as its inputs. In addition, a decomposition task may have other inputs such as available resources, historical experience, specifics of the process, and so on.

- *Output*: The single output of a decomposition task (as an input of the composite task) is a detailed description of a sub-process, which is equivalent to the associated composite task in terms of the contribution to the overall process.
- *Incoming control*: A decomposition task receives notifications from the preceding tasks of the associated composite task upon their completion.
- *Outgoing control*: A decomposition task notifies the associated composite task upon its completion.

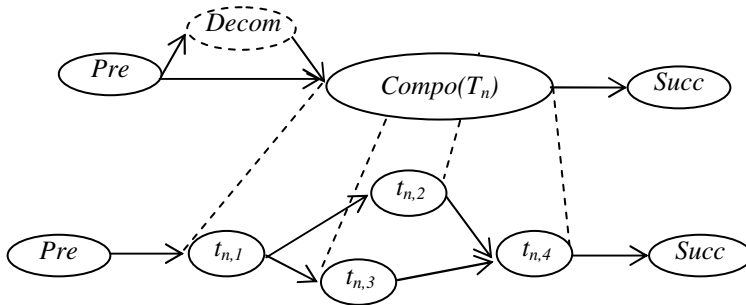


Fig. 3. A decomposition task and the associated composite task

Clearly, a decomposition task forms an *AND-JOIN* structure with the preceding tasks of the associated composite task. This is because the decomposition work should be completed before the execution of the composite task. Normally, a decomposition task should be carried out before the preceding tasks of the composite task are finished. This kind of decomposition is regarded as a “*pull*” model where a composite task is decomposed in advance. Hence, once the work is passed to the composite task, the sub-process resulting from the decomposition can be enacted immediately. However, in the worst case, the decomposition depends on the up-to-date status of process enactment and the peer associated with a decomposition task only starts when the preceding work has been done. This is the reason why a decomposition task receives the notifications from the preceding tasks of the composite task (see Figure 3). This kind of decomposition is regarded as a “*push*” model where the enactment of a decomposition task is triggered passively and may block the whole process.

With the support of the decomposition task, the definition of the composite task is temporarily stored together with the definition of the associated decomposition task, i.e., at those peers that are associated with authorised participants. As a result, the instantiation can be done with the same mechanism for atomic tasks. On behalf of an authorised person, a peer will create an instance of a composite task when required.

3.3 Task Execution

To enact a decomposition task in an organised manner, the associated peer may monitor the progress of process enactment through communication with other relevant peers and take various inputs into account. Additionally, analysis and modelling tools might be used to facilitate the decomposition performer. After the completion of a decomposition task, the associated peer notifies its single succeeding peer, i.e., the

peer associated with the composite task, and transmits an updated task description to advise the specifics of the composite task. The composite task is eventually enacted according to the updated task specification.

Once a composite task is decomposed into a set of sub-tasks, each sub-task is executed by a capable peer. Thus, the fulfilment of a composite task, i.e., the execution of sub-tasks, is achieved through coordination among relevant peers, including the peer creating the instance of this composite task and the peer(s) carrying out the sub-tasks. As a result, based on Figure 3, the peer network of the present instance is converted, as shown in Figure 4 where P_{pre} is a set of peers who execute the preceding tasks of T_n ; P_{succ} is a set of peers who execute the succeeding tasks of T_n ; P_{decom} is the peer who executes the decomposition task; and $P_n, P_{n,1}, P_{n,2}, P_{n,3}$ and $P_{n,4}$ are peers who execute $T_n, t_{n,1}, t_{n,2}, t_{n,3}$ and $t_{n,4}$, respectively.

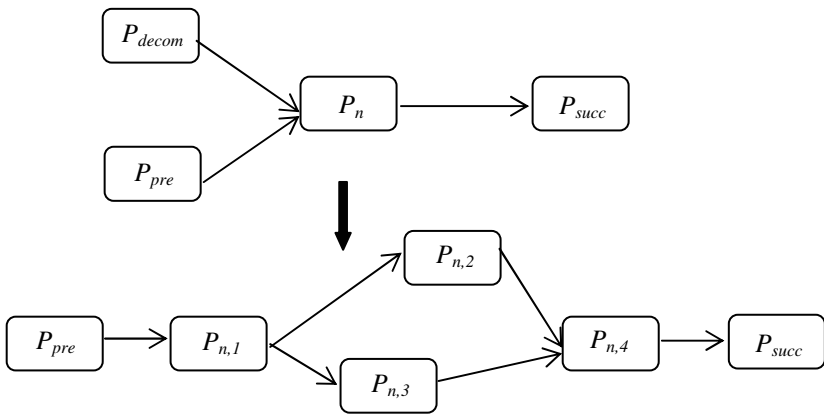


Fig. 4. Conversion of peer network for present instance

Decomposition can be performed at both the instance and process levels. Instance-level decomposition represents a provisional change and only takes effect in the present instance. Allowing instance-level decomposition reflects the fact that a flexible process may have multiple, variant instances. Each of the instances fulfils the composite tasks in a different way. On the contrary, process-level decomposition represents a permanent change to the workflow model and will be applied to all the instances created in the future. Permanent change is always associated with process evolution.

Instance-level decomposition is relatively simpler to handle, as the description of the sub-process is valid once only. In this case, peers $P_n, P_{n,1}, P_{n,2}, P_{n,3}$ and $P_{n,4}$ simply use the local temporary client-server architecture to satisfy coordination requirements, where P_n acts as the server and $P_{n,1}, P_{n,2}, P_{n,3}$ and $P_{n,4}$ act as the clients. All the instances of the sub-tasks are created at peer P_n , and presented to $P_{n,1}, P_{n,2}, P_{n,3}$ and $P_{n,4}$ via a work list. Again, the selection of each of $P_{n,1}, P_{n,2}, P_{n,3}$ and $P_{n,4}$ is dynamic and negotiation-based. However, peers performing sub-tasks do not have direct interactions. Scheduling of the execution of the sub-tasks is done on the server side because

the logical structure of the sub-process only resides on P_n . This approach eases the implementation and management of sub-tasks because the client-server architecture reflects logically the hierarchical relationship among the peers. At the same time, this local centralisation does not have many side-effects because centralised coordination only occurs in a small scope temporarily.

Process-level decomposition is more complicated. To reuse the sub-process definition to create instances later, process data should be refreshed at the process level. The definition of sub-processes should be distributed properly and the existing process repositories of relevant peers may need to be updated. In addition, to execute the present instance properly, the network of peers should be reconstructed properly to reflect the changes. Using the composite task in Figure 4 as an example, execution of the decomposition results in a process-level conversion from T_n to $t_{n,1}$, $t_{n,2}$, $t_{n,3}$ and $t_{n,4}$. The consequent operations of a process-level decomposition are described as follows:

- (1) The model of the sub-process, created by P_{decom} , is passed to P_n ;
- (2) P_n acts as the definition peer to partition and distribute the definition of the sub-process, with the mechanism discussed in [12, 13];
- (3) P_n instructs the relevant peers to create an instance of the sub-process, with the mechanism discussed in [12, 13]. The network of peers which consists of $P_{n,1}$, $P_{n,2}$, $P_{n,3}$ and $P_{n,4}$ is the result of this instantiation;
- (4) P_n advises P_{pre} to interact with $P_{n,i}$ directly instead of contacting P_n ;
- (5) Similarly, P_n advises P_{succ} that $P_{n,i}$, instead of P_n , will interact with P_{succ} directly;

The above steps convert the peer network of the present instance. The following steps will update the process repositories of relevant peers:

- (6) Peers in P_{pre} update their process repositories of the new sub-process model. Namely, the post-conditions of the corresponding tasks are changed. Besides, peers in P_{pre} also propagate this update to other capable peers which have the relevant process definition, and let them know about this change;
- (7) Similarly, peers in P_{succ} update their process repositories of the new sub-process model. Namely, the pre-conditions of the corresponding tasks are changed. Besides, peers in P_{succ} also propagate this update to other capable peers which have the relevant process definition, and let them know about this change;
- (8) P_n deletes the definition of T_n and propagates this deletion within the corresponding virtual community because task T_n no longer exists for future instances;
- (9) Similarly, P_{decom} deletes the definition of the decomposition task and propagates this deletion within the corresponding virtual community because the decomposition task no longer exists for future instances.

After discussing the support for incomplete composite tasks, incomplete atomic tasks can be handled in a similar way. The assumption is that an incomplete atomic task can be treated as an incomplete composite task in SwinDeW. This is justifiable because an incomplete atomic task can always be grouped with other relevant tasks to form an incomplete, controlled, composite task naturally. These tasks are component tasks of the formed composite task. Another reason to use incomplete composite task support for incomplete atomic tasks is that incomplete atomic tasks are normally interrelated and need to be addressed as a whole.

In summary, meeting the requirements of supporting incomplete processes for SwinDew, both instance- and process-level decomposition can be carried out adequately, with the SwinDeW extension smoothly prototyped for proof of concept. The right person (authorised performer of the decomposition task) performs the decomposition work at the right time (either push or pull model) and at the right place (the peer associated with the performer), without bringing the whole system down. Due to space limits, other details such as inconsistency handling for currently running process instances and prototyping are not addressed here, and are available in [12].

4 Case Study

To illustrate the key ideas proposed in this paper, a CICEC (Centre for Internet Computing and E-Commerce) research project is used as an example. The experience obtained from this case shows that the approach proposed in this paper can support various scenarios of similar nature well. The project attempts to provide a leading-edge forum for the establishment, development, coordination, visualisation, testing and evaluation of Internet-enabled e-business ventures that will lead to successful, new e-business models and supporting techniques. Some related initiatives this research targets are development of (1) a suitable e-business modelling environment utilising the current and substantive knowledge and data of many e-business descriptions and models; and (2) a suitable wide-area workflow framework as infrastructure support for e-business processes.

Coincidentally, this project is conducted in the same way as a workflow process is executed in SwinDeW, although the management of this research project is not formally supported by a workflow management system. First, the project aims are achieved through the accomplishment of individual research tasks which have inherent relationships and should be performed in a certain order. Thus, the overall research project is easily modelled as a process. Second, this project involves several research teams which focus on various research tasks where communication and coordination among these teams are normally carried out directly between them.

Initially, only the major tasks of this project are specified. Tasks for e-business modelling research and workflow research are carried out by two research teams in parallel first, and then the task for integration needs to be carried out by another research team, to integrate the achievements. Obviously, at the early stage of the project, these three research tasks can only be modelled as composite tasks. Although the goals and expected outcomes of each task are expressed, how to specifically achieve the goals through some steps remains uncertain. The particular research schedule for each task can be gained only after some initial work such as a literature review has been done. In other words, the decomposition of composite tasks into sub-processes should be performed on-the-fly. For this reason, decomposition work is modelled explicitly as essential steps before the execution of the research tasks. The descriptions of three research tasks are firstly distributed to the leaders of three research teams, respectively, together with three extra decomposition tasks. Using the task of workflow research as an example, after some initial work (treated as the preceding tasks of the workflow research task) has been done, the leader of the workflow research team is able to specify how the research should be conducted. The correspond-

ing decomposition task is executed, resulting in a sub-process that consists of four sub-tasks which should be executed in sequence, namely system design, build-time functions, run-time functions, and prototyping. Then, these four tasks are assigned to the investigators in the workflow research team for execution. There is no evident difference between instance-level decomposition and process-level decomposition in this example, because this research project is a case-based process.

Practical experience has proved that stepwise process modelling and on-the-fly task decomposition naturally reflect the requirements of applications in the real world. In addition, the successful management of the research project also indicates that the approach proposed in this paper works well in supporting incomplete processes with uncertain composite tasks.

5 Related Work

Incomplete process support for workflow is an important area [1, 6]. Some work, although very limited, has been done in this area. WASA workflow [10], which aims at supporting flexible and distributed scientific workflows, presents a hierarchical workflow execution approach based on a set of states and accompanying state transitions for workflow instances. A complex activity may have a nested structure and activity models that are created using a set of activity modelling operations. Some other research (e.g., [3, 5]) focuses on human-centred solutions and argue that interactive enactment should be pursued more vigorously as a framework for flexible workflow modelling, allowing incomplete workflow models to emerge. Kumar and Zhao's research [8] represents a new approach to process design. It emphasises the dynamic perspective and is based on enumerating various tasks to be performed. The approach maximises the alternatives routes and enables dynamic routing during process execution to enhance flexibility. Mangan and Sadiq [9] develop a framework for specifying the process model from a standard set of modelling constructs and given process constraints. The constraints specification allows a process schema to be tailored to individual instance requirements.

These approaches address the issues of incomplete process support from the modelling technique rather than the system coordination support point of view. Although various mechanisms for process modelling are presented, aspects of system support for on-the-fly process articulation are rarely addressed. Moreover, these approaches are all based on the conventional client-server architecture. Relevant research on decentralised workflow environments is hardly ever conducted. This is where the work reported in this paper is founded.

6 Conclusions and Future Work

Incompletely specified process support has evidently become important in today's workflow research. In particular, decomposition of composite tasks on-the-fly is a typical case for incompletely specified processes because of increased process complexity and lack of information. In this paper, this difficulty is addressed from a system coordination support viewpoint, in the context of SwinDeW, which is a peer-to-peer based decentralised workflow system. A hierarchical process modelling and execution

approach is presented, which models and executes a process incrementally in a stepwise manner. The major distinction of this approach is modelling decomposition work as essential steps towards process goals. In this way, the build-time work and run-time work are seamlessly integrated, and the mechanisms used to support completely specified processes can be easily extended to support on-the-fly task decomposition.

In the future, further research on incompletely specified process support will be conducted. Issues of consistency and validity at both the instance and process levels will be further addressed. Other incompletely specified process aspects, such as task elaboration, dynamic process navigation, will also be further investigated.

Acknowledgement

The research work reported in this paper is partly supported by Swinburne VC's Strategic Research Initiative Grant 2002-2004. It is also partly supported by the National Natural Science Foundation of China under grants No.60273026 and No.60273043.

References

1. Aalst, W.M.P., Jablonski, S.: Dealing with workflow change: identification of issues and solutions. *Int. Journal of Computer Systems Science & Engineering* 15 (2000) 267-276
2. Aberer, K., Hauswirth M.: Peer-to-peer information systems: concepts and models, state-of-the-art, and suture systems. *Proc. of the 8th European Software Engineering Conf. (ESEC) and 9th ACM SIGSOFT Symp. on the Foundations of Software Engineering (FSE-9)*, 326-327, Vienna, Austria, Sept. 2001
3. Faustmann, G.: Configuration for adaptation - a human-centred approach to flexible workflow enactment. *Computer Supported Cooperative Work* 9 (2000) 413-434
4. Fischer, L. (ed.): *Workflow Handbook 2002*, Lighthouse Point: Future Strategies, (2002)
5. Håvard, J. D.: Interaction as a framework for flexible workflow modelling. *Proc. of the 2001 Int. ACM SIGGROUP Conf. on Supporting Group Work* (2001) 32-41, Boulder, USA, Sept./Oct. 2001
6. Jablonski, S., Stein, K., Teschke, M.: Experiences in workflow management for scientific computing. *Proc. of the 8th Int. Workshop on Database and Expert Systems Applications (DEXA'97)*, Toulouse, France, Sept. 1997
7. Kirwan, B., Aisworth, L.K.: *A guide to task analysis*. Taylor and Francis, London, 1992
8. Kumar, A., Zhao, L.: Dynamic routing and operational controls in a workflow management system. *Management Science* 45 (1999) 253-272
9. Mangan, P., Sadiq, S.: On building workflow models for flexible processes. *Proc. of the 13th Australasian Database Conf.* (2002)
10. Weske, M.: State-based modelling of flexible workflow executions in distributed environments. *Journal of Integrated Design and Process Science* 3 (1999) 49-62
11. Yan, J., Yang, Y., Raikundalia, G.K.: Critical issues in extending p2p-based SwinDew system for incomplete process support. *Proc. of the 8th Int. Conf. on CSCWD* (2004) 312-317, Xiamen China, May 2004
12. Yan, J.: *A framework and coordination technologies for peer-to-peer based decentralised workflow systems*. PhD Thesis, Swinburne University of Technology, Australia, (2004)
13. Yan, J., Yang, Y., Raikundalia, G.K.: SwinDeW - a peer-to-peer based decentralised workflow management system. Accepted by *IEEE Transactions on Systems, Man, and Cybernetics, Part A* (2005)