Journal of Southeast University (English Edition) Vol. 23, No. 3, pp. 474 – 478 Sept. 2007 ISSN 1003—7985

Dynamic hierarchy description and constraint rules of flexibility workflow

Yang Fei Yin Baolin

(School of Computer Science and Technology, Beihang University, Beijing 100083, China)

Abstract: A dynamic hierarchical description method for workflow is presented. The method provides a dynamic hierarchical way to define a workflow with non-determinate or dynamic factors. With this method, the main process defined at build-time can be reified and extended by the principle of the sub-organizations at either the build-time or the run-time. To ensure the consistency and integrity of the description, a series of constraint rules are also discussed to realize seamless integration between a decomposed process and its original one. This approach supports the description of unpredictable uncertainties, the dynamic hierarchy of business process, and the dynamic modification of enterprise organizations, and all of these improve the flexibility and extendability of workflow management systems dramatically.

Key words: flexible workflow; dynamic hierarchy description; task decomposition; split process

Present workflow management systems have to face new challenges such as enlargement of system scalability, expansion of the application fields, and re-organization of enterprises. The traditionally static schema cannot adapt to the changes and always enforces a process deviating from its definition. In practice, most process flows are flexible and have random factors. Some of these factors are determinate and others are non-determinate or dynamic, which are unpredictable at build-time[1]. The non-deterministic and dynamic factors cannot be described using traditional modeling techniques. A new modeling technique is required to support the dynamic description of the non-determinate factors to make workflow management systems more adaptive, flexible, and extensible[2].

Flexibility of workflow allows some parts of the process definition to be changed at run-time[3]. Saqid et al. [3] presented a notion of BUILD activity to solve the flexible description of a process with multi-branches. The approach in Ref. [4] introduced a new node named block box to encapsulate non-determinate factors. In Ref. [5], a workflow model was provided based on coordination theory and feedback mechanism. However, all of these researches have the same premise that the non-determinate factors in processes are predictable. Furthermore, adding new nodes can make a system more complex, and cannot be dealt with by the above mentioned methods.

Mangan et al. [6] proposed a flexible workflow model based on a sub-process. Jorgensen[7] developed an interaction-based framework for flexible workflow models. Bassil et al. [8] extended the application programming interface of the workflow reference model to support flexible operations. These researches support dynamic extension of processes, but lack consideration for the influence of organization changes on business processes and the integrity of process extension.

In order to solve the above problems, we present a dynamic hierarchical description methodology of the workflow. In every layer of process definition, a unified description method is used without introducing any special modeling elements. The methodology supports the description of unpredictable non-determinate factors and the dynamic changes of organization structures. The process designer may define only the main process at build-time, and allow the activity performer to decide whether to execute its task according to the definition or extend it in more details by dividing it into smaller units which will be done by his subordinates. To ensure the consistency and integrity of the extended process, some constraint rules will apply.

1 Flexibility Workflow Supporting Dynamic Hierarchy Description

Flexible workflow should be able to handle dynamic changes of enterprise organizations and realize business process reengineering promptly. There are two meanings of the term “flexibility”. One is flexible definition, and the other is flexible extension. In flexible definition, process definition tools can describe all complex business processes in the simplest way but

Received 2007-05-18.

Biographies: Yang Fei (1978—), female, graduate; Yin Baolin (corresponding author), male, professor, yin@ nlsde. buaa. edu. cn.

leave flexibility to the user to supplement the process definition and subordinate definition at run-time based on his demand. In flexible extension, users can extend some unpredictable non-determinate factors at build-time during process execution and set the function range of his subordinates.

1.1 Workflow definition

Definition 1 A business process is an ordered set composed of a series of activities. A business process contains several activities and control rules between them. Process is defined as

$$\text{Process} = (\text{Activities, Rules})$$

where Activities is the set of activities in a process; Rules is the set of the relation between activities.

Definition 2 Activity is the basic unit of process, formally expressed as

$$\text{Activity} = (\text{ID, Name, Performer, Data, Tool, Mapping, Condition})$$

where ID is the unique identifier of an activity; Performer appoints the activity actor; Data is the set of workflow data related to the activity, including input data and output data; Tool indicates the data processing tool; Mapping is the set of data mapping, describing the relation between workflow data and input/output data of the process tool; Condition is the control condition of state transition.

Definition 3 The performer is the activity actor, it is defined as

$$\text{Performer} = (P_{\text{ID}}, P_{\text{Name}}, P_{\text{IP}}, \text{SuperiorList, JuniorList, ColleagueList, RoleList})$$

where P_{ID} is the unique identifier of a performer; P_{Name} is performer name; P_{IP} is the node processor that the performer assigned; SuperiorList is the superior list of the performer; JuniorList is the junior list of the performer; ColleagueList is the colleague list of the performer; RoleList is the role list of the performer' s acts.

Definition 4 Organization stands for the relation of the hierarchy and the subjection among performers inside an enterprise, it is defined as

$$\text{Organization} = (O_{\text{ID}}, O_{\text{Name}}, \text{Principal, Performers, ParentID})$$

where O_{ID} is the unique identifier of an organization; O_{Name} is the organization' s name; the Principal stands for the head of an organization; Performers are the members of an organization who are the primary activity actors and subordinates of the Principal; ParentID is the organization ID that O_{ID} belongs to, the former is the parent-organization and the latter is the sub-organization. A parent-organization may have many sub-organizations, but a sub-organization can only belong to one parent.

Definition 5 Role is a 3-tuple:

$$\text{Role} = (\text{RoleName, Capabilities, Privilege})$$

where RoleName is the name of a role; Capabilities is the set of capabilities, the role has such as order handling, remittance statistics; Privilege is the set of access privileges of the role to workflow data.

Definition 6 Workflow data is a variable list in an activity defined as

$$\text{Data} = (D_{\text{Name}}, D_{\text{Type}}, D_{\text{Value}})$$

where D_{Name}, D_{Type}, D_{Value} represent data name, data type, and data value, respectively.

Definition 7 The node processor is comprised of a definition node processor and a running one, defined as

$$\text{Node} = (N_{\text{Name}}, N_{\text{IP}})$$

where N_{Name} is node name; N_{IP} is node IP. A definition node processor that is provided for a designer can be assigned to one or more designers. Similarly, a running node processor provided for a performer can be assigned to one or more performers. A node processor can be either a definition one or a running one, or both simultaneously. On the condition of not being confused, we can also call the running node processor as the node processor.

Designer depicts main process at build-time. Upon completion, the process description will be divided into smaller units to form the least description set each activity execution needs, which will be distributed to the corresponding node processor.

1.2 Dynamic hierarchy description of flexibility workflow

Dynamic hierarchy description places emphasis on maintaining the logic structure in a superior layer, but leaves some uncertainties in the inferior layer. At build-time, the designer defines the main process and gives each activity a complete description. At run-time, the performer (principal of an organization) further reifies or extends the description. The process description will not be confirmed until the task is finished.

At run-time, the performer can take the task as a primitive one or a composite one:

1) A primitive task is the basic entity of workflow execution with a single function. Under the drive of a workflow engine, the performer can directly call a process tool to handle the task according to its description.

2) A composite task is composed of multiple subtasks. If the performer of the task is the principal of an organization, he has the privilege to decompose the

task within the organization and generate multiple process units to be handled by his subordinates, and its function after decomposition is unchanged.

Let M(Activity) be the performer of Activity, O be an organization and O^P be the principal of O. Task decomposition is defined as Decompose(Activity) = (Activity$_1$, …, Activity$_v$) if and only if M(Activity) = O^P, where Activity is decomposed activity, the process Activity belongs to is the main process, the process the sequence Activity$_1$, …, Activity$_v$ are composed of is the split process, and v is the number of activities in the split process. The main process and the split process are a couple of correlated conceptions.

The precondition of task decomposition is that the performer must have subordinates under his command. He can decompose the task and let his subordinates handle it. A performer may decompose a task on his node processor directly, or when one of its instances has arrived. After that, a new split process independent on the main process will be produced, which takes the decomposed activity as the starting node and the ending node, and the new produced activities as middle nodes.

The example shown in Fig. 1 is used to demonstrate dynamic hierarchy description. Process A described by the process designer in the first layer including six activities: A_1, A_2^*, A_3, A_4, A_5^* and A_6, where the performers of A_2^* and A_5^* with the symbol $*$ is the principal of an organization, respectively. After process A is defined and distributed, each node in it has its own activity description. The performer of A_2^* has the privilege to decompose the task and A_2^* is decomposed into process B, described as Decompose(A_2^*) = (A_2^*, A_{21}, A_{22}, A_{23}, A_{24}^*, A_2^*). The performer of A_{24}^* in process B is the principal, A_{24}^* can be further decomposed as Decompose (A_{24}^*) = (A_{24}^*, A_{241}, A_{242}, A_{243}, A_{244}, A_{24}^*). A_5^* is decomposed as Decompose (A_5^*) = (A_5^*, A_{51}, A_{52}^*, A_{53}, A_5^*) similarly. The performer of A_{52}^* is the principal, but he takes the task as a primitive one and handles it according to the activity description by himself. The task execution sequence in Fig. 1 is A_1, A_2^*, A_{21}, A_{22}, A_{23}, A_{24}^*, A_{241}, A_{242} (A_{243}), A_{244}, A_{24}^*, A_2^*, A_3, (A_4), A_5^*, A_{51}, A_{52}^*, A_{53}, A_5^*, A_6.

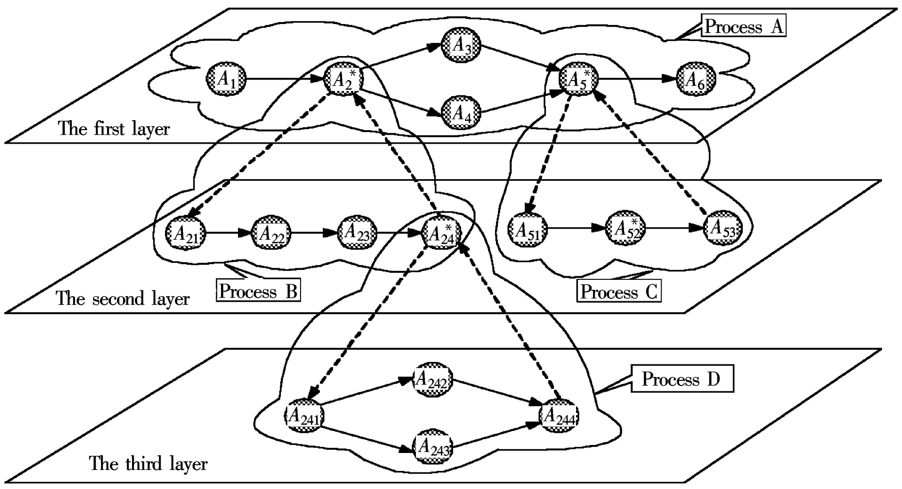

Fig. 1 Process structure supported dynamic hierarchy description

It is not necessary for the designer to distinguish the primitive task and the composite one or to care about how the activity performer treats the task at runtime. Each task is taken as a primitive one by designers. It is only the activity performer who can make the final decision. All specific features can be encapsulated into its main process. In this way, the concrete task description and task disposition of sub-layers can be encapsulated as well as predictable and unpredictable non-determinate factors, which not only facilitate cooperation among different organizations but also make it easy to dynamically adapt process models.

2 Constraints on Dynamic Hierarchy Description

New process activity nodes are inserted during task decomposition. In order to ensure the consistency and integrity of the decomposed process and avoid errors such as data loss and illegal access, we should enforce some constraints upon task decomposition. Assume that the main process has an activity sequence of A_1, …, A_n ($n \geqslant 1$), where A_k ($1 \leqslant k \leqslant n$) is the decomposed activity. Let S be split process sequence A_{k1}, …, A_{kv} ($v \geqslant 1$) of A_k.

2.1 Process node constraints

Process node constraints are the limits to the start activity and the end activity, and the activity performers in the split process, where the start and end activities must be the decomposed activities A_k, i. e., $A_{k1} = A_k, A_{kv} = A_k$.

The aim of task decomposition is to divide a task into smaller units and assign them to subordinates of the performer, thus the activity performer of split process S must be subordinates of A_k' s performer, defined as follows:

If $m = M(A_k), m \in O$, for each $m' = M(A_{kj}), j = 1, \ldots, v$, then we have $m' \in O$.

2.2 Activity data constraints

Activity data constraints are the limits to data access. Since data is the core of the business process, access to it must conform to some constraints to ensure the consistency and integrity of task decomposition.

There are two subsets of data in an activity: input data and output data. The data constraints apply to both the data field set and the data type. Let the data field set of the decomposed activity A_k be denoted as $D(A_k)$, which includes the input data field set $D_{\mathrm{IN}}(A_k)$ and the output data field set $D_{\mathrm{OUT}}(A_k)$; and let the data type be denoted as $T(d), d \in D(A_k)$, which include the input data type and the output data type.

1) Input data constraints

During decomposition, the original input data of the decomposed activity A_k should be unchanged, i. e., the input data of A_{k1} after decomposition should be the same as those of A_k. However, there may be some temporary data added in the split process. This means the input data of A_k is a subnet of the set of the input data of the union of all activities in the split process. Let $N(d)$ be the name of input data field d, the input data constraints can be described as follows:

For any $d \in D_{\mathrm{IN}}(A_{k1})$ there exists $d' \in D_{\mathrm{IN}}(A_k)$ such that $N(d') = N(d)$, and vice versa; at the same time we have $D_{\mathrm{IN}}(A_k) \subseteq \cup D_{\mathrm{IN}}(A_{kj}), j = 1, 2, \ldots, v$.

If $d \in D_{\mathrm{IN}}(A_k), d' \in D_{\mathrm{IN}}(A_{k1})$, and $N(d') = N(d)$, we have $T(d') = T(d)$.

2) Output data constraints

During decomposition, the original output data of the decomposed activity should be unchanged though we may add some temporary output data to save intermediate results. Namely, the output data of A_{kv} after decomposition should be the same as those of A_k. The set of output data of A_k is a subset of the union of the output data of all activities in the split process, described as follows:

For any $d \in D_{\mathrm{OUT}}(A_{kv})$ there exists $d' \in D_{\mathrm{OUT}}(A_k)$ such that $N(d') = N(d)$, and vice versa; at the same time we have $D_{\mathrm{OUT}}(A_k) \subseteq \cup D_{\mathrm{OUT}}(A_{kj}), j = 1, 2, \ldots, v$.

If $d \in D_{\mathrm{OUT}}(A_k), d' \in D_{\mathrm{OUT}}(A_{kv})$, and $N(d') = N(d)$, we have $T(d') = T(d)$.

2.3 Access privilege constraints

Access privilege constraints are operation limits to activity data. There are three kinds of access privilege to data field d of activity A_k, denoted as $P(d)$: ① No read and no write; ② Read only; ③ Read and write. The access level, denoted by L, from the lowest to the highest is $L(1) < L(2) < L(3)$.

The access privilege to data field d of A_{k1} after decomposition is the same as that of A_k. If the access privilege level of data field d is $L(P(d))$ in an activity, then each access privilege level of d in activities of the split process is no higher than $L(P(d))$. In a word, the access privilege constraints can be described as follows:

For each $d \in D_{\mathrm{IN}}(A_k), d' \in D_{\mathrm{IN}}(A_{k1})$, if $N(d') = N(d)$, then we have $P(d') = P(d)$.

For each $d \in D_{\mathrm{IN}}(A_k), d' \in D_{\mathrm{IN}}(A_{kj}), j = 1, 2, \ldots, v$, if $N(d') = N(d)$, then we have $L(P(d')) \leqslant L(P(d))$.

3 Conclusion

In this paper, a new description method of flexibility workflow is discussed. The method provides a dynamic hierarchical way to define a workflow with non-determinate or dynamic factors during the definition stage. With this method, the main process of a workflow defined at build-time can be reified and extended afterwards by the principal of the sub-organizations, at either the build-time or the run-time. This approach supports the description of unpredictable uncertainties, the dynamic hierarchy of business processes, and the dynamic modification of enterprise organizations, and all of these improve the flexibility and extendibality of workflow management systems dramatically. Furthermore, a series of constraint rules on process node, activity data, and access privilege are discussed, which ensure the consistency and integrity of dynamic hierarchy description.

References

[1] van der Aalst W M P, Jablonski S. Dealing with workflow change: identification of issues and solutions [J]. *International Journal of Computer Systems Science and Engineering*, 2000, **15**(5): 267 − 276.

[2] Zhou Jiantao, Shi Meilin, Ye Xinming. State of arts and trends on flexible workflow technology [J]. *Computer Integrated Manufacturing Systems*, 2005, **11**(11): 1501 − 1510. (in Chinese)

[3] Sadiq Shazia, Sadiq Wasim, Orlowska Maria. Pockets of flexibility in workflow specification [C]//*Proc of the* 20*th International Conference on Conceptual Modeling*. Yokohama, 2001: 513 − 526.

[4] Sun Ruizhi, Shi Meilin. A process meta-model supporting dynamic change of workflow[J]. *Journal of Software*, 2003, **14**(1): 62 − 67. (in Chinese)

[5] Fan Yushun, Wu Cheng. Research on a workflow modeling method to improve system flexibility [J]. *Journal of Software*, 2002, **13**(4): 833 − 839. (in Chinese)

[6] Mangan Peter, Sadiq Shazia. On building workflow models for flexible processes [C]//*Proc of the* 13*th Australasian Database Conference*. Melbourne: University of Melbourne, 2002: 103 − 109.

[7] Jorgensen H D. Interaction as a framework for flexible workflow modeling [C]//*Proc of the International ACM SIGGROUP Conference on Supporting Group Work*. New York: ACM Press, 2001: 32 − 41.

[8] Bassil S, Rolli D, Keller R K, et al. Extending the workflow reference model to accommodate dynamism [R]. Montreal: Software Engineering Laboratory of University of Montreal, 2003.

柔性工作流动态层次描述及其约束规则

杨　飞　　尹宝林

(北京航空航天大学计算机学院,北京 100083)

摘要:为支持流程中不确定性因素和动态因素的描述,提出了柔性工作流的动态层次描述方法,使得流程设计人员在流程定义阶段定义的主流程可以在任意时刻由子组织的负责人进行细化和扩充. 为保证动态层次描述的正确性和完整性,讨论了动态层次描述的一系列约束规则,确保分解后的流程与原有流程实现无缝衔接. 该描述方法支持对不可预知的非确定性因素的描述,支持业务流程的动态层次描述以及组织机构的变化,极大地增加了工作流管理系统的柔性和可扩展性.

关键词:柔性工作流;动态层次描述;任务分解;分解流程

中图分类号:TP391