IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP5160

This paper appears in *Managing Modern Organizations Through Information Technology*, Proceedings of the 2005 Information Resources Management Association International Conference, edited by Mehdi Khosrow-Pour.

Partial Completion of Activity in Business Process Specification

Joe Y.-C. Lin and Maria E. Orlowska

School of Infor. Technology & Elect. Engin., The University of Queensland, St. Lucia, Brisbase, Australia, {jlin, maria@itee.uq.edu.au}

ABSTRACT

We examine the current workflow modelling capability from a new angle and demonstrate a weakness of current workflow specification languages in relation to execution of activities. This shortcoming is mainly due to serious limitations of the corresponding computational/execution model behind the business process modelling language constructs. The main purpose of this paper is the introduction of new specification/ modelling constructs allowing for more precise representation of complex activity states during its execution. This new concept enables visibility of a new activity state – partial completion of activity, which in turn allows for a more flexible and precise enforcement/monitoring of automated business processes.

This paper also shows feasibility of such extension to the current well-established workflows technology.

INTRODUCTION

The business requirements for enterprise-wide information system are generally very complex and engage a large number of activities that represent certain business policies, business rules and procedures. Therefore precise modelling of business process requirements is one of the challenges in information system design and specification. Ability to capture the most important parts of business procedure and filter out the irrelevant complexity of the real world is expected from experienced analysts. A correct process model is essential for any software development and its accuracy affects the overall system's quality.

The goals of business process modelling include the support of several major objectives [1]: human understanding of relevant process aspects, process improvement while activities are analysed in detail, process management and development by observing individual activity dependencies, and facilitation of detailed analysis for process execution. Thus process modelling is a powerful tool that provides a means of depicting complex business functions in a structured form that can serve as a blueprint for communication between all parties involved in a system's design.

Flow of activities modelling is one of the most common business process modelling techniques and it has proved to be useful in many applications' domains, mostly where the backbone of the system functionality is a predefined process with a high volume of expected instance executions. The main objectives of workflow process modelling are to provide high-level specification of processes that manage the execution of the tasks involved in a business activity, the scheduling of resources and the control of the associated data flow required executing the tasks.

Over time, many workflow modelling approaches have been presented, such as Petri-Nets [2], [3], and other more restricted languages [4][5][6][7]. Most of these published literatures emphasised the control flow perspective - the structure of the process that monitors and schedules workflow activities. The modelling of the control flow is the primary consideration and forms the basic skeleton for other modelling components. Although there are more and more successes in workflow research and development, there are still technical problems, such as inflexible and rigid process specification and execution mechanisms, insufficient possibilities to handle exceptions, dynamic modification of processes, process status monitoring, automatic enforcement of consistency, and concurrency control.

In this paper we concentrate our discussion on the problem of flexibility and extensibility of process specification and execution mechanism. We exam the current workflow modelling technique from another angle and demonstrate another dimension of the weakness of current workflow specification languages in relation to execution of activities. This weakness is supported by important observations on the corresponding computational models behind the language constructs leading to identification of the actual source of identified shortcomings. The main contribution of this paper is the introduction of new specification/ modelling constructs allowing for more precise representation of complex activity states. This new construct will enable a new concept of process execution – partial completion of activity, which allows a more flexible specification of activity definition for business constraints.

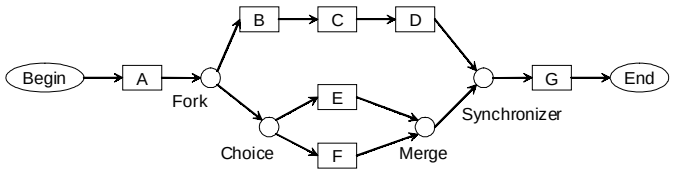
PROCESS CONTROL FLOW MODELLING AND ITS LIMITATION

A workflow process is an automated business task which consists of a collection of activities that support business or organizational objectives. The activities are often tied together by a set of precedence dependency relationships. The schema that specifies activities that constitute the workflow process and dependencies between these activities is called a workflow model.

To provide a background for our further discussion we illustrate some of the basic modelling components using a graphical workflow definition language proposed by Sadiq and Orlowska [8]. This language conforms closely to the Workflow Management Coalition (WfMC) standards [9] and is sufficiently complete to support most typical business applications. Figure 1 shows the graphical representation of the basic constructs in the language.

The graphical language of this workflow model uses two basic types of objects: node, and control flow. Nodes are classified into two subclasses: activity and condition. An *activity*, graphically a rectangle, is used to represent the work to be completed as a part of the process. A *condition*, represented by a circle, is used to construct the logic for and-split (fork), and-join (synchronizer), xor-split (choice) and xor-join (merge). A *control flow*, represented by a directed edge, always links two nodes. By

Figure1. Basic Workflow Constructs



connecting the nodes with control flows, we build a directed acyclic graph (DAG) called a workflow graph.

The above basic workflow constructs define the structural aspect of workflow that captures the flow of execution from one activity to another. The ordering of the above constructs is not arbitrary. Incorrect combination of the ordering structures may cause structural conflicts such as deadlock and non-reachability [6]. Other aspects of process such as temporal constraints, and participant allocation for activities can not be easily represented on the graph and are often considered separately. Data flow is another important aspect of the process that is tightly coupled with the control flow, and it is sometimes represented on the same graph.

Although the control flow model provides a solid expressive power for specifying the process execution between activities, it does not the have ability to express the execution behaviour below activity level once activity granularity is defined. We will demonstrate this limitation through a business example.

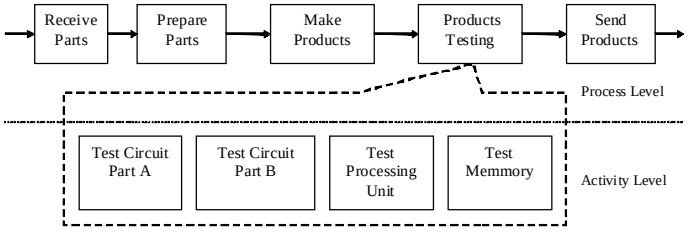
Consider a scenario in an electronics manufacturing company. A part of the manufacturing process of a specific product consist five activities, namely: a) Receive Parts, b) Prepare Parts, c) Make Products, d) Products Testing, e) Send products. The products testing activity is performed by running a test program to evaluate the build quality of four major product components. As these four evaluations are executed by single software that taking at the same place at the same time, it is modelled as a single activity. Figure 2 represents the workflow model of this manufacturing process in two levels.

The process assumes the production quality can be determined by the number of successful tests on the product. In situations when the process is delayed due to the delay of part arrival, the company would like to shorten the product testing time by completing fewer tests. However, due to concern of product quality, a policy is specified so that a minimum of X out of 4 tests must be successful in order to maintain a reasonable quality. If the decision of X is determined by human expert at run time, in order to capture every possible test case, the granularity of the activity must be reduced and all combinations of tests must be depicted.

The above scenario describes a special activity which has different execution behaviour from ordinary activity but the current modelling techniques can not express it easily. The activity requires a new feature to allow a flexible execution control we called “partial completion” of activity. Partial completion of activity means the activity is forced to be completed by the performer while the objective of that activity has not been fully achieved. Although this type of control is not required for every activity, it should be an option for activity that needs to be monitored in detail during execution.

Generally, there are two reasons for partially completing an activity: a) reduce the processing time b) release the resources in advance for other activities. While the decision to make an activity partially complete-able or not is a design issue, the activity may impact on overall process behaviour and as such it need more formal consideration. In the next section, we will focus on the internal activity structures and present a novel approach to model this type of activity.

Figure 2. A Manufacturing Process



MODELLING FINE-GRAINED ACTIVITY BEHAVIOUR

Activity Granularity

We begin with the observation that the granularity of activity is an important factor while designing the workflow model. As the granularity of activity is interrelated with the complexity of the process, it has a direct impact on the performance of process execution. In a workflow system, the granularity and structure of an atomic unit vary, depending on the individual activity requirement. While several approaches [10] [11] [12] model activity from different perspectives for different purposes, WfMC proposed a standard meta-model for activity [13]. Some of the basic activity properties are: Activity ID, Name, Type, Pre/ Post condition, Performer, Application, Duration limit and Deadline.

The key property that determines the complexity of an activity is the activity type. An activity can be implemented in one of the following type: Generic, Sub-process, Block and Loop. While the sub-process and block structure implies the activity is composed of multiple activities, and Loop implies the activity can be executed repeatedly, we will refer to the activity of generic type as the smallest atomic structure within a process. An atomic activity is performed by a single performer or a single role (which may consist of a set of performers). Generally, the activity boundary is determined by its Pre/Post condition. The pre-condition of activity implies the set of data input for the activity that will be used for performing the tasks in the activity. The post-condition implies the goal of the activity, which contains the set of data output that is expected to be generated by the activity.

Activity Execution

During run-time, an execution of an activity is called an activity instance. Most of current workflow management products associate one generic finite state machine as the computational model for every activity instance. Such finite state machine (FSM), as presented by the WfMC, is suggested to have the states; Scheduled, Active, Completed, Suspended, and Aborted [9]. Each FSM consists of a set of these visible states and a set of transitions between these states [14]. States in the machine represent the internal conditions defining the status of an activity instance at a particular point in time [13]. The workflow management system can observe only the states in the FSM. Generally, the transitions represent participant generated events, which controls transitions from one internal state of an activity instance to another. Figure 3 illustrated the states and transition for an activity instance.

Current technology also necessitates that the final state of an activity (generally “completed”) can only be triggered when all the post-conditions of the activity have been met, then the process flow triggers the next activity according the stored process definition. We argue that there is a need to relax this constraint in order to support more flexible execution. While this type of relaxation is necessary, not all types of activity should be allowed. In the next section we identify a specific type of activity that can have more detailed completion behaviour.

Partial Complete-Able Activity

Previously in section 3.1 we mentioned the activity goal is defined by its post-condition in the activity meta-model. The post-condition is only evaluated when the performer tries to complete the activity by

Figure 3. Example FSM for an Activity Instance

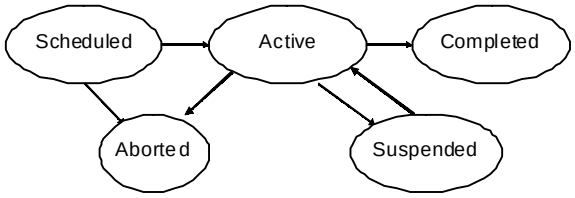
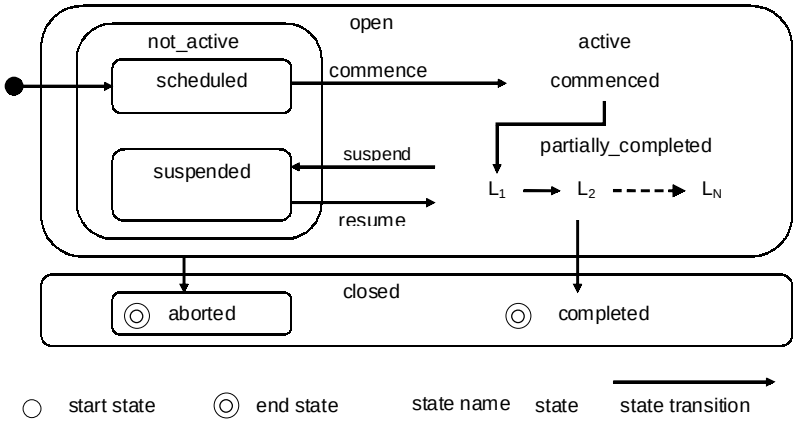


Figure 4. Customizable Finite State Machine for Partial Complete-Able Activity Instance



sending an action to the workflow management system (WfMS), e.g. click the "complete" button. The WfMS will evaluate the condition (for example: all tests have been completed), if the result is successful, the process will proceed to the next activity; if not; the WfMS rejects the completion and sends a notification back to the performer. Therefore the activity completion is based on an "all-or-nothing" behaviour.

However, the example we demonstrated in section 2 shows the ability to complete some activities' is determined by the performer rather than the system. We will refer this type of activity as "*partial complete-able activity*". A partial complete-able activity must have the following characteristics: a) the objective of activity can be decomposed, b) the activity is not process critical, c) monitoring of the activity progress is required. First of all, if an activity can be partially completed it means that the activity has an objective that is composed of multiple atomic objectives. The atomic objectives are semantically meaningful within the context of the process, and each one of them is independent. For example, running a product test on a specific part and return a complete result is meaningful, but running a test without getting a result is meaningless. Secondly, the partial complete-able activity should not be responsible for producing any data that affects the flow of process, otherwise it is critical. And thirdly, since the activity needs to be partial complete-able at any time, the completion of its sub-objectives must be captured and monitored.

At this point, one may ask "If the activity has several objectives, why not model it as a sub-process with multiple activities?" The reasons for not modelling this type of activity into multiple activities include: a) The activity shares the same resource and application for achieving the objectives, b) The objectives must be performed by a single performer at the same time, c) Splitting of the activity potentially increases the overhead on user interaction with the system, e.g. clicking the unnecessary "commence" button followed by clicking the "completed" button multiple times, d) To avoid the system overheads (audit and recovery data, system scheduling overheads, participant assignment and navigation).

Enabling partial complete-able activity provides a new feature to manage activity execution, more importantly; it should allow us to specify "levels" of completion on activity progress. Traditionally, the progress of an activity is only determined by "active" and "completed" states. Therefore the intermediate completion status can not be observed. Now if close monitoring of the progress of completion is required, a number of milestone levels can be modelled in the activity meta-model. Nevertheless, partial completion of activity must have a limit; at the lowest level of partial completion, a constraint should be imposed to ensure the minimum requirement is met. Since the generic finite state machine for generic activity may not support partial complete-able activity, we propose the following modelling enhancement to modelling specification language:

An activity can have a customisable finite state machine as an optional property, for the monitoring of precise activity completion behaviour and allow partial completion [15].

Enhanced Activity Finite State Machine

The modelling enhancement we suggest to support partial complete-able activity can be described using a finite state machine diagram. Figure 4 shows the life-cycle of a partial complete-able activity instance. Like the state model from WfMC [9], this model consists of nested states and it is its simple but useful generalisation.

A partial complete-able activity instance is either in the state *open*, or it is in the state *closed* if it has been *completed* or *aborted*. Upon scheduling by the workflow engine, an activity instance is in the state *open.not_active.scheduled*, which means it is waiting to be taken by the assigned participant(s). Once the activity is selected by a participant from his/her worklist, the activity instance changes into *open.active.commenced*. The active activity instance may be *suspended* or *resumed* any number of times. When the participant performs the minimum requirement of the activity, the activity instance would changes into the first *open.active.partially_completed* state. Once the activity instance reaches this state, the participant may have the option to complete the activity or continue to a higher level of partially completed state. Let the first partial completed state is denoted by L_1, the second state is denoted by L_2, and so on... By progressively completing more of the activity objectives, the activity would eventually move from the lowest partial completed state to the highest (L_N).

Clearly, the model presented subsumes the commonly accepted WfCM model with only one (completed) activity state.

Partial Completion Constraints

The proposed customizable FSM allows the specification of a number of milestone levels in individual task's execution. Each of these levels, are bounded with a requirement or so called *partial completion constraint* which are determined by one of the activity goal aspects, for example, data generated, time spent, etc. The specification of the partial completion constraints can be specified by formal expressions but that requires introduction of formalisation of the concept. The example in section 2 only shows one type of partial completion constraint, which is based on the cardinality of the activity objectives. However, by selecting several types of constraints (out of N) for each partial complete-able level, there is a need of formal reasoning about interdependencies between constraints from this class. The formal results lead towards verification of the specification and subsumption dependencies.

Since, the main purpose of this paper is the introduction of this additional workflows flexibility and discussion on its feasibility and

usefulness, the formal considerations are outside of the presentation scope.

Once this basic concept of partial completion is well positioned in the workflows capability there is a range of related constraints that may offer further additional functionality; for instance - partial completion constraints based on the temporal aspects of the activity.

CONCLUSION

We have presented a business scenario showing the activity execution behaviour is limited by the granularity of activity. While reducing the granularity of activity may not be the most effective way to capture the execution behaviour, due to the complexity of execution paths, overhead of users and system, we proposed a customizable finite state machine model for activity behaviour. The customizable finite state model allows the specification of intermediate states of activities, which captures the activity execution more precisely and enables the activity to be partially complete-able at any time but within the limits expressed by the constraint. The presented notion of the partial completion constraints is simple but its implementation offers powerful extension to features of the current technology. Its feasibility by providing an extension to the activity meta data model has been demonstrated.

The specification language component to capture this type of business requirements needs to be incorporated to the specification language. It is merely the matter of adopting intuitive graphical representation.

REFERENCES

[1] Curtis B, Kellner M and Over J. Process Modelling. Communications of ACM 35(9):75-90, 1992.

[2] Ellis C and Nutt G. "Modeling and Enactment of Workflow Systems," in Application and Theory of Petri Nets, Proceedings 14th International Conference Chicago, Illinois, USA, vol. 691, Lecture Notes in Computer Science, M. Ajmone {Marsan, Ed.: Springer-Verlag, pp.1 – 16, 1993.

[3] Aalst. WMP van der. The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, 8(1):21-66, 1998

[4] Carlsen S. Conceptual Modeling and Composition of Flexible Workflow Models. PhD Thesis. Department of Computer Science and Information Science, Norwegian University of Science and Technology, Norway, 1997.

[5] Casati F, Ceri S, Pernici B and Pozzi G. Conceptual Modeling of Workflows. In M.P. Papazoglou, editor, Proceedings of the 14th International Object-Oriented and Entity-Relationship Modeling Conference, volume 1021 of Lecture Notes in Computer Science, pages 341-354. Springer-Verlag, 1995

[6] Kuo D, Lawley M, Liu C and Orlowska ME. A General Model for Nested Transactional Workflow. In *Proceedings of the International Workshop on Advanced Transaction Models and Architecture (ATMA'96)*, Bombay India, pp.18-35, 1996.

[7] Sadiq W and Orlowska ME. On Capturing Process Requirements of Workflow Based Information Systems. In Proceedings of the 3rd International Conference on Business Information Systems (BIS '99), Poznan, Poland, April 14-16, 1999.

[8] Sadiq W and Orlowska ME. On Correctness Issues in Conceptual Modelling of Workflows. In Proceedings of the 5th European Conference on Information Systems (ECIS '97), Cork, Ireland, June 19-21, 1997

[9] Workflow Management Coalition "Workflow Management Coalition – The Wokflow Reference Model", Document No. WFMC-TC-1003, 1995.

[10] Eder J and Liebhart W. The Workflow Activity Model WAMO. In Proceedings of 3rd Intl Conference on Cooperative Information Systems, Vienna, Austria, September 1995.

[11] Liu L and Pu C. A transactional activity model for organizing open-ended cooperative activities. Technical Report TR96-11, Department of Computer Science, University of Alberta. 1996.

[12] Kiepuszewski B, Hofstede A.H.M. ter and Bussler C. (1999) On structured workflow modelling. Lecture Notes in Computer Science 1789. Springer, 1999.

[13] Workflow Management Coalition. Workflow Management Coalition - Terminology, Document No. WFMC-TC-1011, Issue 3.0, 1999.

[14] Workflow Management Coalition. Workflow Management Application Programming Interface (Interface 2 & 3) Specification. Document Number WFMC-TC-1009. Version 2.0. 1998.

[15] Carter B, Lin J and Orlowska ME. Customizing Internal Activity Behaviour for Flexible Process Enforcement. In Proceedings of the Fifteenth Australasian Database Conference (ADC '04) Dunedin, New Zealand, 2004.