ACL

Cancer Research UK

Advanced Computation Laboratory,

Cancer Research UK

44 Lincoln's Inn Fields

London WC2A 3PX

Tel: 020 7269 3627 Fax: 020 7269 3186

Tallis Training

Scheduling

Ayelet Oettinger

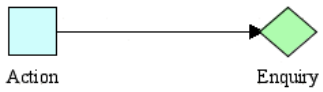
18 July 2005

1. Overview: Workflow vs. Dataflow

Task scheduling determines the flow of the process-description, and the order in which tasks are enacted. The ordering of the task can be controlled directly, by connecting two tasks with a scheduling constraint, or indirectly, by manipulating the state trigger or event trigger of a task.

1.1 Workflow

The scheduling constraint method of ordering tasks creates a deterministic workflow: A scheduling constraint is graphically represented as an arrow connecting two tasks; it species that a pair of tasks should be carried out in a particular order – the task at the head of the arrow cannot start until the task at the tail of the arrow (the antecedent task) has completed. In the figure below, the enquiry's antecedent task is the action. The action has to be completed before the enquiry can be enacted.



1.2 Dataflow

Using the state trigger of a task to control scheduling provides for a more flexible dataflow: Tasks do not necessarily follow one another, but rather monitor the state and the data changes in the process-description and become active in response to these changes. Thus, the graphical view of a dataflow network holds no information as to the ordering of the tasks. The scheduling is manifested in the expressions populating the scheduling properties of the tasks.

1.3 Non-linear Processes

Processes can also be event-driven. Tasks with event triggers monitor events happening outside the process-description and become active in response to certain events. Event triggers can also be used as a way of handing the control over to the end-user: event triggers allow end-users to trigger specific tasks at any point in time, regardless of the scheduling constraints or the state of the process-description. This allows for non-linear processes: end-users can choose between tasks, skip tasks and even re-run tasks.

2. Workflow: Scheduling Constraints and Preconditions

Scheduling constraints specify the order in which tasks are enacted. A scheduling constraint is graphically represented as an arrow connecting two tasks. It indicates that the task at the head of the arrow cannot start until the task at the tail of the arrow (the antecedent task) has completed.

A task can have several antecedent tasks. In order for a task to be considered for enactment, all its scheduling constraints must be met – that is, all antecedent tasks have to be either completed or discarded.

Once the scheduling constraints have been met, the engine then determines whether the task should be activated or whether it should be discarded. If *at least one* of the antecedent tasks was completed, then the task can be activated (though some task properties, such as a state trigger or precondition, might prevent it from running). However, if all the antecedent tasks were discarded, then the task is discarded.

2.1 Preconditions

A precondition is an expression that has to be met before the task can be executed. A task's precondition is checked at the point where a task is being considered for enactment, that is, when its scheduling constraints are met. If at that point the precondition is true, the task becomes active. Otherwise, it is discarded, and is not activated again even if its precondition subsequently becomes true.

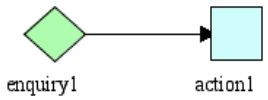
Example: The *Check pregnancy history* enquiry should only be executed if the patient is a woman, that is, if the expression `patient_gender = female` is true. If it is false then the *Check pregnancy history* enquiry is discarded.

Combining scheduling constraints and preconditions enhances the workflow flexibility.

2.2 Workflow Patterns

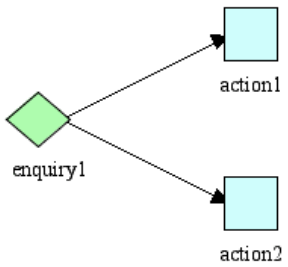
2.2.1 Sequence

A task in the process is activated after the completion of an antecedent task in the process.



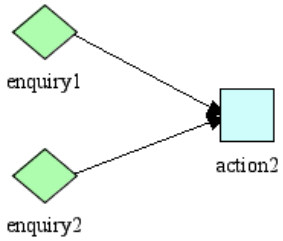
2.2.2 Parallel Split

A point where a single process thread splits into multiple process threads, thus allowing tasks to be executed simultaneously or in any order.



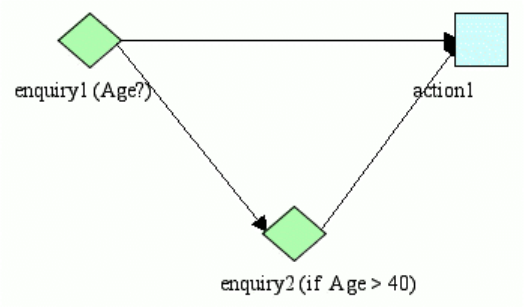
2.2.3 Synchronization

A point where multiple parallel process threads converge into one single thread, thus synchronizing multiple tasks.

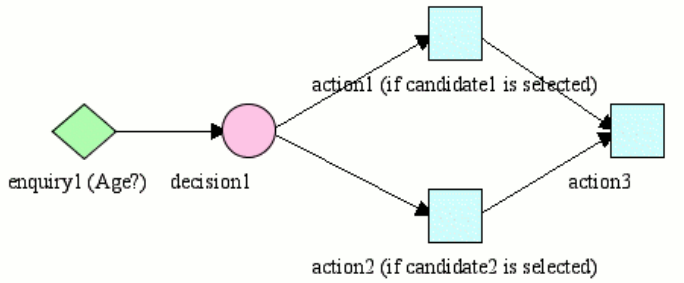


2.2.4 Synchronizing Merge

A point where multiple parallel process threads converge into one single thread. If more than one path is taken, synchronization of the active tasks takes place. If only one path is taken, the alternative branches are discarded (preconditions determine whether branches are discarded).



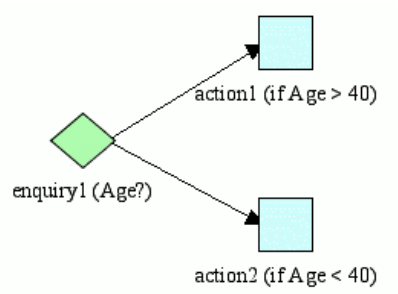
Note: Enquiry2 is activated if the patient is older than 40.



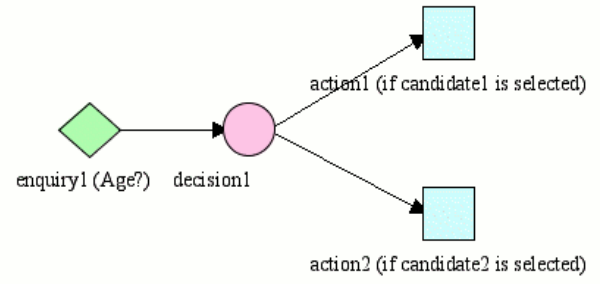
Note: Action1 is activated if candidate1 is selected; action2 is activated if candidate2 is selected.

2.2.5 Exclusive Choice

A point where, based on a decision or workflow data, one of several branches is chosen (preconditions control branch selection).



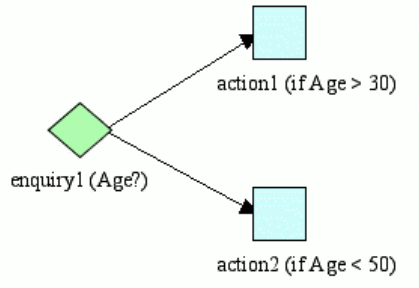
Note: Action1 is activated if the patient is over 40; action2 is activated if the patient is under 40.



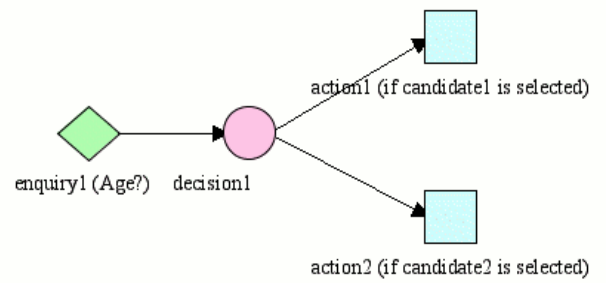
Note: Action1 is activated if candidate1 is selected; action2 is activated if the candidate2 is selected.

2.2.6 Multi Choice

A point where, based on a decision or workflow data, a number of branches are chosen (preconditions control branch selection).



Note: Action1 is activated if the patient is over 30; action2 is activated if the patient is under 50.

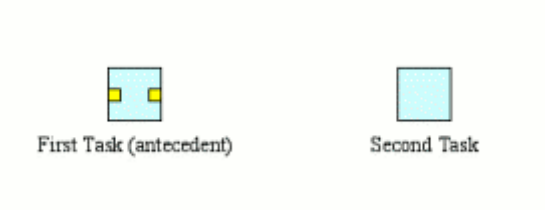


Note: Action1 is activated if candidate1 is selected; action2 is activated if the candidate2 is selected.

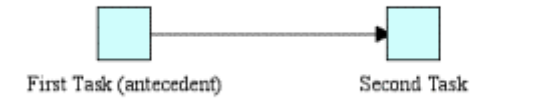
The workflow patterns above are based on patterns found at www.workflowpatterns.com.

2.3 Creating a Scheduling Constraint

Place the cursor above the first task. Two yellow squares appear on the task.

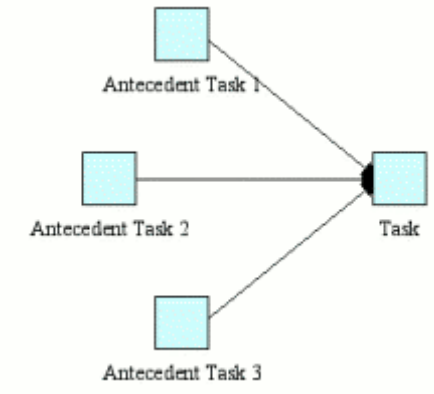


Click on one of the squares, and drag over the second task. Two yellow squares appear on the task – release the mouse key above one of them.



An arrow is formed between the two tasks. The second task executes only after the first task is completed. If the antecedent task is discarded, the second task is discarded as well.

Note: If a task has several antecedent tasks, it only executes after all of them have been executed *and* if at least one of them has completed.



3. Dataflow: State Triggers

State triggers can be used to construct a dataflow: a dynamic process in which tasks do not necessarily follow one another, but rather monitor the state and the data changes in the process-description and become active in response to these changes.

The graphical view of a dataflow network holds no information as to the ordering of the tasks. The scheduling is manifested in the expressions populating the state triggers.

3.1 State Triggers

A state trigger is an expression that has to be met before the task can be executed. The task remains dormant until the expression becomes true.

Like preconditions, state triggers can be combined with scheduling constraints to create complex flow patterns.

A task's state trigger is checked at the point where a task is being considered for enactment, that is, when its scheduling constraints are met. If at that point the state trigger is true, the task becomes active. Otherwise, it remains dormant and continues to monitor the expression. Once the expression becomes true, the task is activated.

Example: The *Control blood pressure* action should only be executed if and when the expression `blood_pressure = HIGH` becomes true.

Note: the main difference between a state trigger and a precondition is that if a precondition is found to be false the task gets discarded, whereas if a state trigger is found to be false the task remains dormant and waits for it to become true.

4. Non-linear processes: Event Triggers

An event trigger is a text string that enables tasks to respond to events taking place outside the scope of the process-description.

The event trigger is useful for creating processes that do not have a predefined workflow. Event triggers allow end-users to trigger specific tasks at any point in time, regardless of the scheduling constraints or the state of the process-description. This allows for non-linear processes: end-users can choose between tasks, skip tasks and even re-run tasks.

Example: A clinician in a patient consultation may choose to invoke one of a number of services by selecting the relevant event trigger for each (e.g. "perform_biopsy", or "perform_imaging", or "discharge_patient", etc.)

Note:

- A task with an event trigger does not execute unless the trigger is invoked, even if all scheduling conditions are met.

- When the trigger is invoked, the task executes, even if scheduling conditions are not met.

The task state returns to dormant after completion – tasks with event triggers can be re-activated.