PR Approval: Timers and Event Generation

maximal

Domain Model

- Same as the previous purchase request approval example.

Workflow Rules

- Track due dates on PR Approval tasks.
  - For types PM and FM, due date is 2 days after the created date.
  - For CFO, due date is 4 days after created date.
- If approval task is not completed on the due date, generate an event to alert the delay.
  - In this example, we will just generate the event which gets queued In the database.
  - How this event is routed to the appropriate queues for handling how further actions are taken (such as sending an email notification) will be discussed in later illustrations.
- When the purchase request approval process is complete, i.e., all appropriate approval tasks are completed, generate an event to notify the requester.

Modeling Concepts Illustrated

- Generating events based on timers.
- Generating events based on variable value changes. In this case, we will be generating events when a change occurs to ”IsApproved” attribute of a purchase request.

Workflow Logic

```
CONSTRAINT PR C1 "PM Approval is needed for every request with amount over $500."
  pr.PurchaseAmt>500 =>
  ENSURE approval WITH approval.Type=PM, approval.Request = pr;

CONSTRAINT PR C2 "If PM approves the request, create an approval for FM"
  pmapr.Status = APPROVED =>
  ENSURE approval WITH approval.Type=FM, approval.Request = pr;

CONSTRAINT PR C3 "If FM approves the request and the amount is over 10,000, then create an
approval for CFO"
  (fmapr.Status=APPROVED AND pr.PurchaseAmt >10000) =>
  ENSURE approval WITH approval.Type=CFO, approval.Request = pr;

CONSTRAINT PR C4 "Request is approved if there are no pending or rejected approvals."
  SSIZE(noapr)=0 => pr.IsApproved;

CONSTRAINT PR C5 "Request is rejected if there are any rejected approvals."
  apr.Status=REJECTED => NOT pr.IsApproved;

CONSTRAINT Approvals C1 "Due dates for approvals to be completed."
   (apr.Type IN [PM, FM] => apr.DueDate = DATEAFTER(apr.CreatedDate, 2, DAYS)) AND
   (apr.Type = CFO => apr.DueDate = DATEAFTER(apr.CreatedDate, 4, DAYS));

CONSTRAINT Approvals C2 TIMED "If approval is not completed by due date, generate an event
to send an alert."
   AT apr.DueDate:
   apr.Status=PENDING => EVENT("ApprovalDelayed", apr);

CONSTRAINT Approvals C3 "When a decision is made on approving a PR, generate an event to
send a notification to requestor"
   ONCHANGE(pr.IsApproved) => EVENT("PRApprovalDone", pr);
```

Calculation of due dates.

On the due date, if a PR Approval's status is still pending, generate an event.

- *Event Type*: "ApprovalDelayed"
- *Event Target*: PR Approval activity.

This triggers an event whenever there is a value change on pr.IsApproved.

In this example, initially IsApproved starts off with a null or unknown value. If it goes to APPROVED, an event is generated. Then if it changes to REJECTED, another event is generated.

Test Cases

maximal

Test 1

1. Create a purchase request with amount $20,000.
2. This creates a PRApproval for PM.
3. Update the status of this PRApproval to APPROVED.
4. This creates PRApproval for FM.
5. Update the status of this PRApproval to APPROVED.
6. This creates PRApproval for CFO.
7. Update the status of this PRApproval to APPROVED.
8. This changes purchase request's IsApproved to true.
9. Execute query events API call. You should see an event with type "PRApprovalDone".
10. Not change the CFO approval to REJECTED.
11. Now you should see another event of the same type.

Test 2

1. Create a purchase request with amount $20,000.
2. This creates a PRApproval for PM.
3. Update the status of this PRApproval to APPROVED.
4. This creates PRApproval for FM.
5. Update the status of this PRApproval to APPROVED.
6. This creates PRApproval for CFO.
7. Change the Created Date for CFO Approval task to 10 days prior to today. This will make its Due Date to be sometime in the past. This should generate an event "ApprovalDelayed".
8. Run the query events to check if the event is generated.

You can now modify the created date back and forth to multiple different values. Every time the due date goes back to sometime in past, an event gets generated.

Maximal actively monitors these timers and takes actions appropriately.

Modifying Event Generation Logic

Now let us modify the constraint below and see what happens.

```
CONSTRAINT Approvals C3 "When a decision is made on approving a PR, generate
an event to send a notification to requestor"
   ONCHANGE(pr.IsApproved) => EVENT("PRApprovalDone", pr);
```

TO

```
CONSTRAINT Approvals C3 "When a PR is approved or rejected, generate an
appropriate event"
   (pr.IsApproved => EVENT("PRApproved", pr) AND
   (NOT pr.IsApproved => EVENT("PRRejected", pr)
```

This generates events of different type based on whether a PR is approved or rejected. Whenever a pr.IsApproved becomes TRUE from FALSE or UNKNOWN, an event is generated. Same with when the value turns to FALSE from TRUE or UNKNOWN.

Summary

- Maximal supports time-dependent constraints.
  - These are only active after the time specified.
  - The specified time can be a variable.
  - These constraints can become active or inactive based on the specified time.
- These time-dependent constraints have wide-applicability: blocking workflows for a duration, monitoring due dates and generating alerts, and so on.

- Maximal supports event generation based on timers, on any changes to variable values, or any arbitrary conditions.
  - These events can be routed to relevant queues for further processing.
    - Notifications, alerts, integrations, and other purposes.
  - We will study several use cases of this powerful event framework in later examples.