**Workflow Framwork**

# NodeEnterActionHandler

* Not an end-state

1. New a step. *stepManager.newStep(jobId, stepName, actionTime)*
2. Publish node enter message.

* Is an end-state

1. Update job status. *jobManager.updateStatus(jobId,finalJobStatus)*
2. Publish job completed/aborted/failed event
3. If there is parent job, call back parent task.

# NodeLeaveActionHandler

* Not a start-state

1. Update step table *stepManager.updateStepStop(jobId, stepName, stepStatus.getFinalStatus(), actionTime)*
2. Set step status to jBPM context variable.
3. Publish step leaving event.

* Is a start-state

1. Update job status to “Started” *jobManager.updateStatus(jobId, Status.Started)*

# Step life cycle

1. NodeEnterActionHandler --> new a step and set step status to "Started, & set STEP\_STATUS\_ variable to "Good"
2. At NodeEnterActionHandler or SubProcessActionHandler create new step and update STEP\_STATUS\_ variable to “Good”. At JbpmFacadeImpl.cancelTask update update STEP\_STATUS\_ variable to "Failed" or "Aborted"
3. At NodeLeaveActionHandler, update step status according to STEP\_STATUS\_ variable.

if STEP\_STATUS\_=="Good", then step\_status="Completed"

else step\_status = STEP\_STATUS\_;

# Job life cycle

1. JobManagerImpl.createJob set job status to "Created" and set createTime
2. At NodeLeaveActionHandler for start node, set job status to "Started" and set JOB\_STATUS variable to "Good"
3. At NodeLeaveActionHandler update JOB\_STATUS variable to "Failed" or "Aborted", if a step "Failed" or "Aborted"
4. At NodeEnterActionHandler for end-state

* Update job status *jobManager.updateStatus(jobId,finalJobStatus)*
* Publish event *publishEvent(jobId, finalJobStatus.toString(), job.getName(), EventType.job.toString(), actionTime, jobProperties)*
* Call back parent task, if there is one

# Auto Task life cycle

# User Task life cycle

# CommandAssignmentHandler

1. Set actor-id = “tdm-agent”
2. Set env variables
3. Set taskType = “Command” and excutionHost
4. Start runCommandAsync on tdm agent side
5. On exception, add exception comment and set exception-task-id to variable

# RoleAssignmentHandler

1. Set task type to “User”
2. Set poolActors to role users

# completeTask

1. Add comment
2. End current task, and according to stop step to move to next step or to the end-state

# exitTask (TaskInstance ti, Status stepStatus)

exitTask is called by cancelTask, exitOnException and endProcessInstance

1. Cancel current task, and put step status
2. cancel all other open tasks in the whole job tree

* Get all process instance in the tree and call cancelProcessInstanceTasks(processInstance, cancelActorId) for each one
* In cancelProcessInstanceTasks, look for all open task and cancel each one.

# cancelTask

1. Add comment
2. Call exitTask (see below)

# exitOnException

exitOnException is called by

* JbpmManagerImpl.startProcess --> exitOnException
* WorkflowManagerImple.completeTask --> JbpmManagerImpl.exitTaskOnException --> exitOnException

It get exception task id from *BPM\_EXCEPTION\_TASK\_ID* variable and call exitTask.

# endProcessInstance

# Require new transaction.

StepManageImpl.newStep from NodeEnterActionHandler

jobManager.updateStatus from ProcessStartActionHandler & ProcessEndActionHandler to update job status to started or others

stepManager.updateStepStop from NodeLeaveActionHandler to update step status

JobManagerImpl.createJob

JbpmManagerImpl.startProcess

JbpmManagerImpl.exitOnException

TODOs

-------------------------

3. prepare doc

4. Create mapping for Step-->Task-->subJob

5. set lazy fetch in TdmTaskInstance ???

6. \*\*\*\*\*\*\*\*\*\* abort on one leg, stop all other legs' job \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To check

-------------------------

1. Job table

2. Step table

3. Task instance table

4. getJob summary

5. getJob

Support for stop phrase

------------------------\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8

1. In completeTask, check stop phase, if the step in stop phase, go to end. How to handle multi tasks in a step?

Currently, step step is only work for linear flow without sub flow !!!

User needs to be in a right role to start/complet/abort a task

----------------------- \*\*\*\*\*\*\*\*\*\*\*\*\*\*?

put a property in task/job VO to indicate the priviledge?

Test cases

-------------------------

1. Simple flow with exception on automatic task. (e.g can talk to agent) -- good

2. Simple flow with success automatic task. -- good

3. Simple flow with user task

searchPendingTask

start task

add comment

completeTask

abortTask

4. test getStep

5. test getTask

6. combine flow with auto and user task. Complete/abort case

7. Test stop phrase support for linear flow.

8. Parallel flow with auto and user task, complete/abort case

\* complete all task normally -- good

\* complete one leg, abort the other leg -- good

\* abort the first let, to check cascade abort on the other leg and the whole process. -- good

9. Double parallel flow with auto and user task, complete/abort case ------\*\*\*\*\*\*\*\*\*\*Still having problem

\* complete all task normally -- good

\* complete one leg, abort the other leg -- good

\* abort the first let, to check cascade abort on the other leg and the whole process. -- good

support for multiple transition at start

----------------------------------------

Done

sub-flow design

-------------------------------

1. create a parentJob & parentStepName in Job

3. create subJob in TdmTaskInstance

4. create main release flow

5. create all sub flow for step

6. start main flow

7. at sub-flow step, start sub flows one by one

8. creates <event type="process-end"> for every sub flow. end parent task

9. cancel case \*\*\*\*\*???

create(Job)

Create(job, jobs)

1. put status and create time somewhere else

\*\*\*\*\*\*\*\*\*\*\*\*\*Very important \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<task-node name="deploy-components" create-tasks="false" signal="last-wait">

<event type="node-enter">

<action class="com.eterra.asf.tdm.manager.workflow.handler.action.SubProcessActionHandler">

</action>

</event>

<task name="task"/> <!-- must have this, ortherwise the node won't flow to next node -->

<transition to="verify" name="to-verify"></transition>

</task-node>