Concept - Review Process (Workflow)

Goal

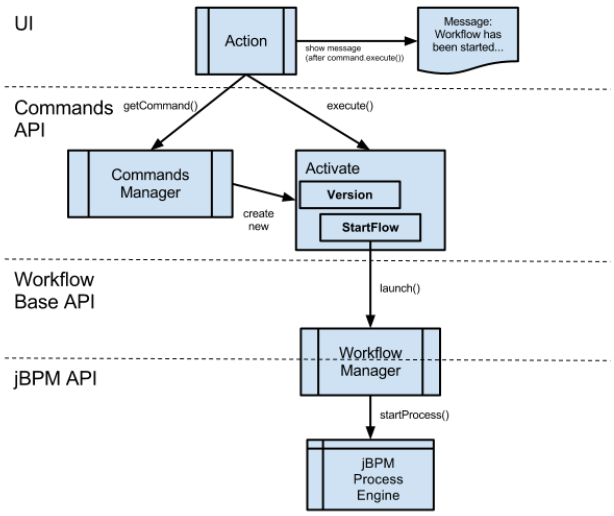
The goal of this story is to re-implement the 4-eye review process from Magnolia 4.5 using the jBPM process engine.

Proposal

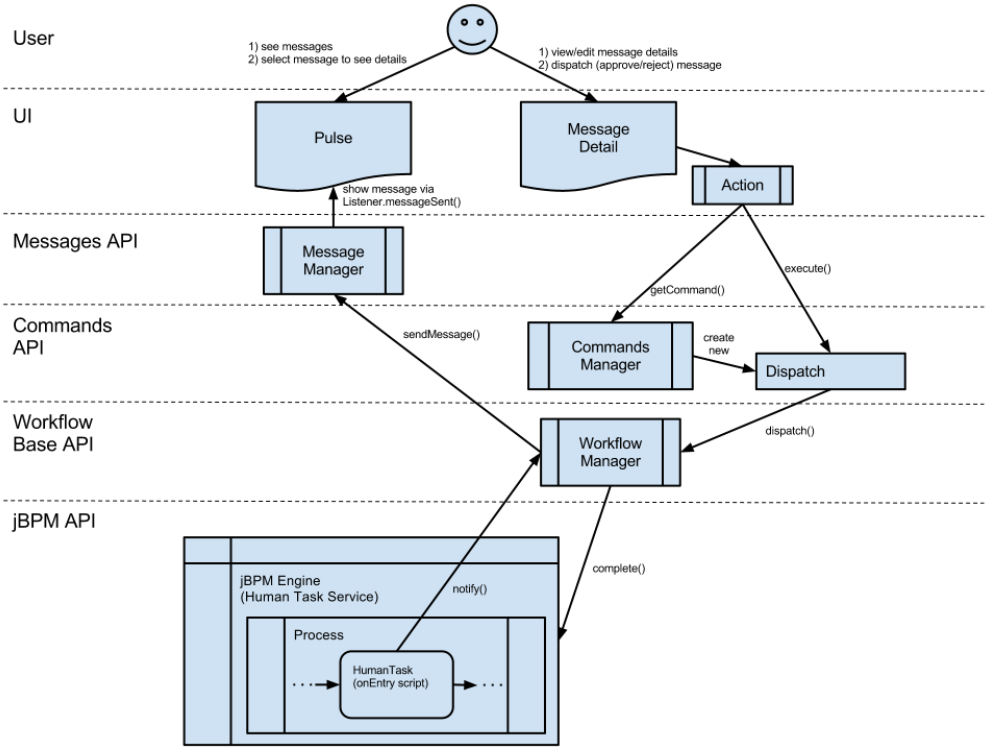
Architecture

The functionality will be split into two modules: **workflow-base** and **workflow-jbpm**. The workflow-base will contain the basic API definition and the UI implementation, whereas the workflow-jbpm module will contain the jBPM based back-end implementation.

Commands API will be used to launch new processes, as well as to provide human interaction with the running processes (approval/reject/...). This allows to untie the launching and especially the approval process from the M5 UI and allow external systems to enter the process.

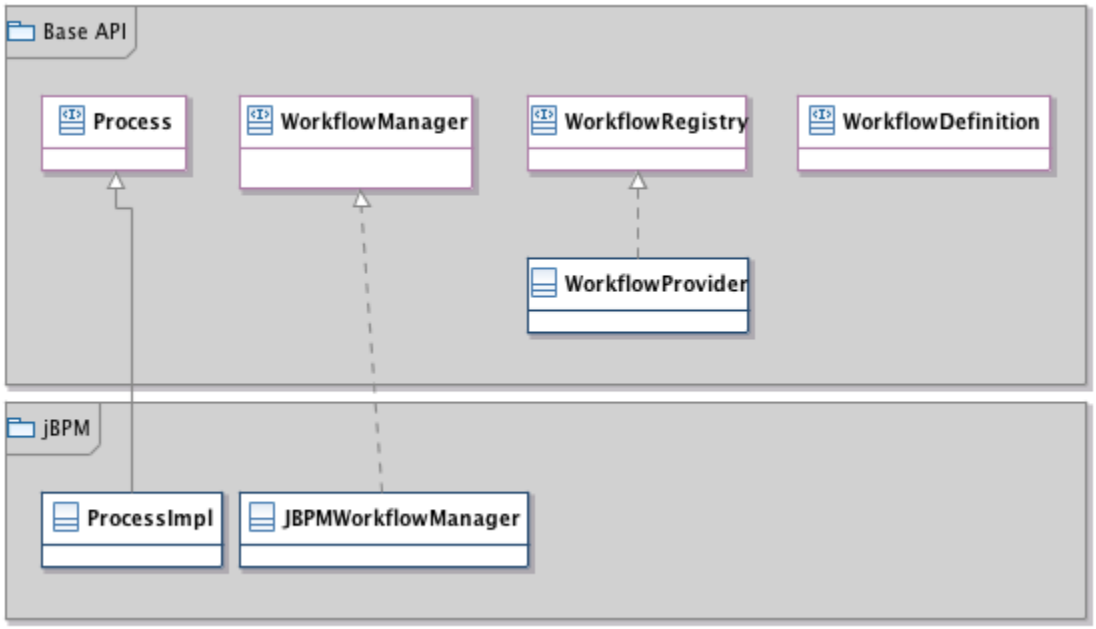


The Messages API will be used to inform users about processes waiting for approval.



Note to the above process: when the message is dispatched by user (approved/rejected/...), it has to be deleted from the list (to prevent multiple dispatching).

The following picture provides the class diagram of basic classes:



Base API

WorkflowManager

Core interface, defines methods to launch new process, resolve a running process by its ID, as well as the methods to be used from within the Process to communicate with Magnolia (e.g. send messages to the user/group, etc.).

Defined as a component, with a dummy implementation in the Base module.

WorkflowDefinition

This interface defines basic constants and methods of the bean that provides the definition of a workflow.

WorkflowRegistry

Provides getWorkflows() and getWorkflow(name) methods - to get all defined workflows and a concrete workflow definition by its name.

WorkflowProvider

Basic implementation of the WorkflowRegistry interface - allows to store the workflow definition as a String property in the configuration, or as a classpath resource identified by a file name.

Process

This interface defines basic constants and methods for the running process.

jBPM Module

ProcessImpl

An implementation of the Process interface, wraps the jBPM's ProcessInstance class.

JBPMWorkflowManager

`@Singleton`

An implementation of the WorkflowManager interface, provides its methods using the jBPM process engine.

Start-up

The Base module does no special initialization.

The jBPM module starts the KnowledgeService (process engine) and the (Local)HumanTaskService (and the persistence layer for them), and provides them to the JBPMFlowManager instance.

Configuration

Base Module

Besides the UI configuration, the Base module contains the workflow definitions. These are stored at `$MODULE_PATH/config/workflows`, e.g. activation and deactivation workflow definitions.

The definition node has following properties:

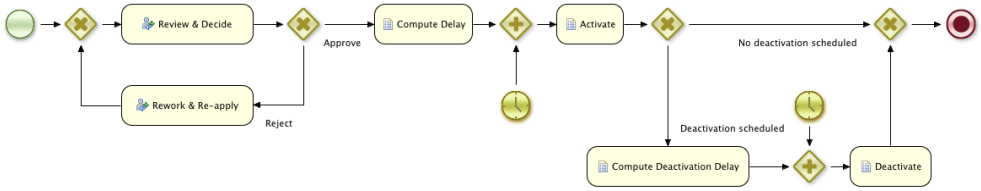
- `name` - simple name of the definition, e.g. `activation`
- `id` - the "system" ID of the definition, used to launch new process based on this definition; e.g. `info.magnolia.workflow.baseActivation`
- `type` - identifies the type of the definition, ATM always "`bpmn`"
- `definition` - the String property with the actual definition, may be empty / not present - in such event, the `fileName` property must exist;
- `fileName` - if exist, then defines the name of the resource in `CLASSPATH`, where the definition is stored;

jBPM Module

Contains the configuration of the persistence layer for the KnowledgeService and HumanTaskService.

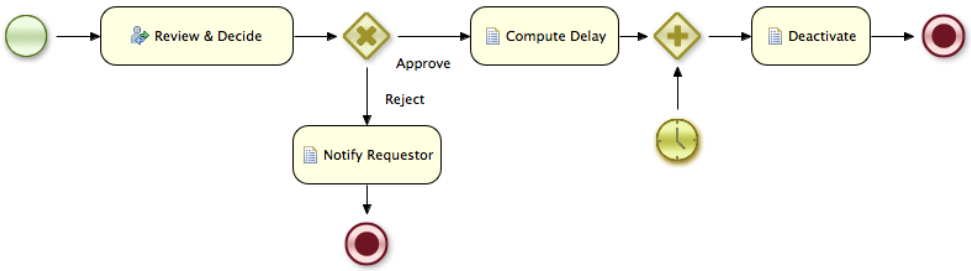
Workflow Definitions

Activation



TODO

Deactivation



TODO

Implementation Notes

The following is a scratchpad for ideas and notes from the implementation process - these should be processed in the next (Alpha4) sprint.

- What about using taskClient.start(taskID) when MessageDetail for the Task is open, to block others from performing actions on it? Of course, others will be still able to open MessageDetails on a task, but they should not be allowed to click the "Approve"/"Reject"/... buttons (and perhaps might see an information which user has claimed the task).
- DONE: Use TaskEventListener in the TaskService to handle incoming (and outcoming) Human Tasks, rather than to rely on the process designers to make things properly.
    - allow customers to register their own TaskEventListeners, to be able to cooperate with the existing customer BPMS;
- IMPORTANT (for Documentation): If the jBPM module fails to start with a *"User transactions are not supported by the current configuration, the jBPM engine cannot be started."* message, and the servlet container is Tomcat (and probably also Jetty), the `context.xml` file with the following content has to be added to the `META-INF` folder of the Magnolia webapp:

    context.xml

    ```
    <Context>
        <Transaction factory="bitronix.tm.BitronixUserTransactionObjectFactory" />
    </Context>
    ```

- ...