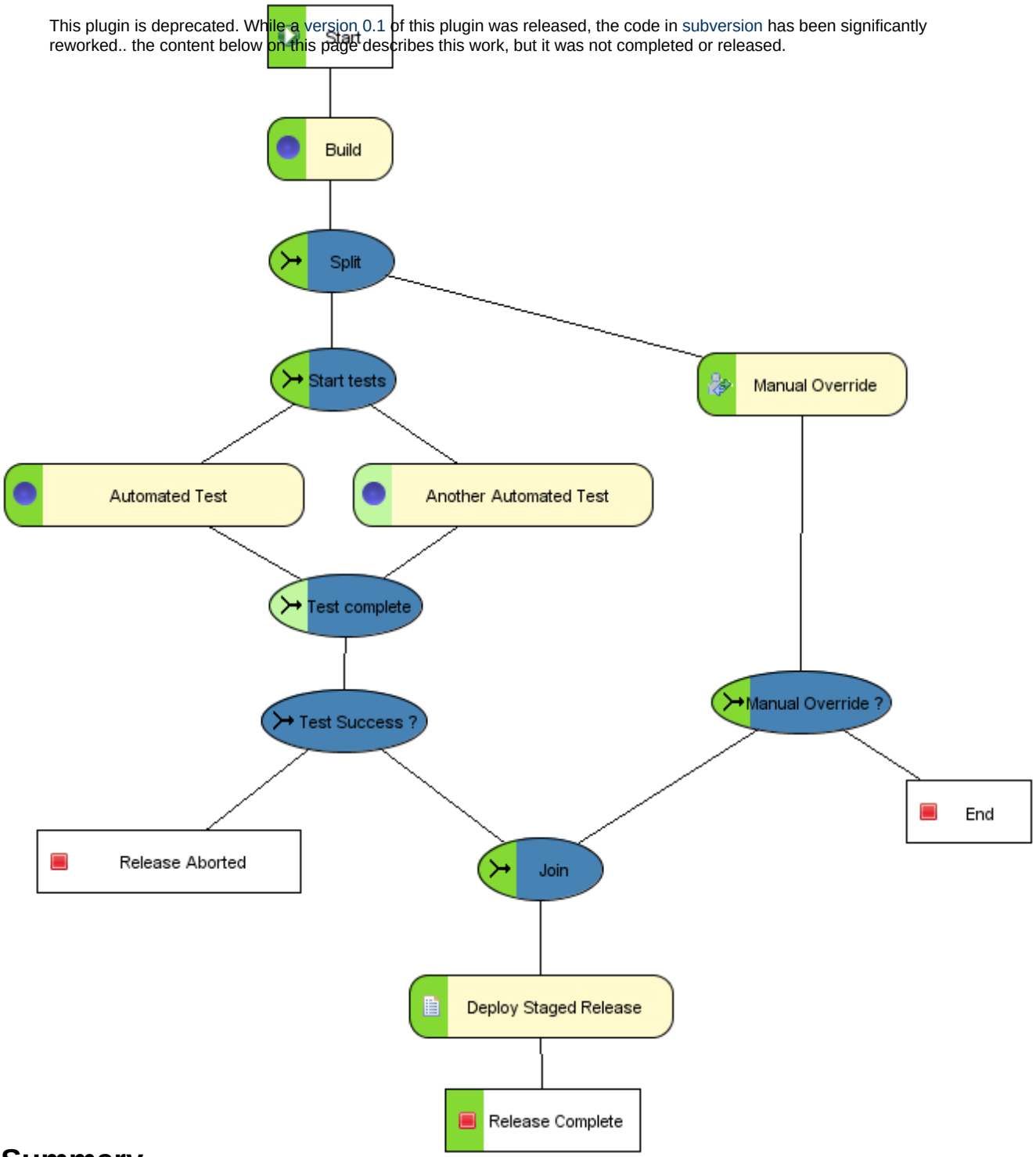
Drools Plugin

Plugin Information
No information for the plugin 'drools' is available. It may have been removed from distribution.

This plugin is deprecated. While a version 0.1 of this plugin was released, the code in subversion has been significantly reworked.. the content below on this page describes this work, but it was not completed or released.



Summary

This plugin integrates Drools with Hudson. It lets you define your own build, test and release workflows in a graphical designer, and execute them inside Hudson. The graphical representation allows non-developers to reason about and help design processes. It also gives a much better view of the state of a process at runtime. At a glance anyone can get an idea of wat is happening with a build.

Workflows can create tasks for human users too, e.g. manual testing or signing off on releases. This is also a way to input data during the workflow.

Getting Started

This assumes some familiarity with Hudson, Eclipse and Maven.

First, install the following

- The Drools Plugin for Hudson using the plugin manager (version 0.2)
- Eclipse 3.5
- The Drools Workbench (update site: http://downloads.jboss.com/drools/updatesite3.4/)
- The Hudson extension to the Drools Workbench (update site: https://svn.dev.java.net/svn/hudson/trunk/hudson/plugins/drools/hudson.drools.updatesite, user:guest, no password)

Next, we'll create a Maven/Eclipse project to develop the workflows in.

- Put this pom.xml into a new directory.
- Use 'mvn eclipse:eclipse' to create an Eclipse project, and then import it. (Or use m2eclipse -- I didn't test this).

And then our first workflow...

- Create a new 'RuleFlow File' in the project and call it MyFirstWorkflow.rf. You should see three Workitems on the bottom left called 'Script', 'Build' and 'E-Mail'
- Open the properties view and select the workflow. Set Package to MyFirstWorkflow and Id to MyFirstWorkflow
- Add a 'Build' node. Select it and set properties Project to 'Project1' and Name to 'Project1'
- Add an End node
- Add connections from Start to Project1 and from Project1 to End
- Click the green checkmark to validate your workflow

To deploy it:

- Right-click the file, Hudson > Create Project
- Hudson URL should be filled in, enter username and password if necessary, enter project name: "MyFirstWorkflow"
- Go to Hudson, create a freestyle project called 'Project1' (no extra settings needed)

Then try it:

- Browse to Hudson
- Click on the MyFirstWorkflow project. You should see a the workflow graph.
- Start a MyFirstWorkflow build. On the build page you should see a graph for the workflow process instance.
- A Project1 build should be started.

Manual

The Drools plugin defines some custom work items you can use in your workflows.

Build

This is a custom work item for Hudson that starts a build and waits for completion.

Node properties:
- Project: name of project to build
- Complete when failed: complete the work item (advancing the workflow) if the build failed
- Complete when unstable: complete the work item (advancing the workflow) if the build is unstable
When the work item is not completed after the build, the user will have the choice (in the UI) to either accept the result or retry the build.
* All work item parameters of type string, boolean or RunWrapper are passed as parameters to the Hudson build.
* When build and work item are completed, the build will be available to the workflow as result parameter 'Build', wrapped in a RunWrapper.

Script

This work item represents a script, witten in Groovy that runs inside the Hudsn JVM and has access to all Hudson internals.

Scripts have access to the following parameters:

- session (the Drools StatefulKnowledgeSession)
- hudson (Hudson.getInstance())
- args (a map with the parameters passed by the workflow)
- out (a PrintWriter that logs to the build log)

In the script properties, only a 'Name' is specied. The actual scripts can be configured in the global Drools Management page, or on the project page (TODO).

If the script returns a Map, that is passed as a result to the workflow. Any other object will be wrapped in a Map with key 'result'.

When scripts fail (throw an exception), the workflow does not advance, and the user can manually restart them.If you don't want this, then don't throw an exception.

Human Task

A human task is a way to get user input during the workflow. Using a Groovy DSL you can define some parameters the user should enter.

The DSL code goes into the 'Content' property of the work item.

Syntax:

```
task(title:"Question?", private:false) {
    id_of_parameter_1 (
        type: "string",
        description: "What is your reply ?",
        defaultValue: "default reply"
    )
    id_of_parameter_2 type:"boolean", description: "bla", defaultValue:
true
    id_of_parameter_3 type:"choice", description: "choice parameter",
choices: [ "foo", "bar" ]
}
```

- the single line and multiline parameters are equivalent
- valid types are string, boolean and choice
- defaultValue is optional (defaults to null or false)
- When completed, a result parameter will be created for every question, with the same id.
- If ActorId is present, only a Hudson user with the same id can complete the task
- If private is true, only the actor can see the task (not tested yet)

Events

On every build completion in Hudson, an event is created, called 'BuildComplete:projectName', which you can subscribe to in a workflow. However you should realize that this event may be triggered multiple times, and every event will be received by every running process instance. Consider starting this build from the workflow itself.

Additional information

- When a Run object (the record of a build) needs to be passed as a parameter or a result, it is always wrapped into a hudson.drools. RunWrapper object.
- The renderer of workflows supports WorkItem, HumanTask, Build, Script, Split, Event, Start, End and ForEach nodes.Other nodes work are rendered as a black&white box showing only the node name. Connections are always rendered as straight lines from center to center.