Master of Technology

Individual Project Report

DRAM Customer Return System

by

Ong Boon Ping

**1. Abstract**

In DRAM backend plant, QA department oversees the handling of customer return parts. At first, QA re-run the DRAM testing flow on the customer return parts. Then, test engineering department will be asked to provide further technical intent of the failing parts.

After gathering the all the information, QA will decide whether further industrial action to be taken based on the finding of the customer return part.

**2. Problem Description**

QA lacks a multi-user platform that allows QA, test engineering and sales representatives to key in each finding. The platform should be able to run with simple rules that can help QA to do decision making.

This platform should save the time used for coordination works and decision making.

**3. Knowledge Modelling**

**3.1 Knowledge identification**

The knowledge is identified through manual extraction of QA customer return procedure and interviews of test engineering and sales expert.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Source | Insights from the Source | Acquisition Technique |
| 1 | QA | How the cases involve customer return DRAM parts can be closed:   1. Degrader 2. Guardbanded Parts 3. Temperature Violation 4. Additional testing 5. Customer system analysis 6. Requesting parts with higher specification | Manual extraction from Customer Return Procedure |
| 2 | DRAM Test Engineering | Type of analysis that can be done on the returned DRAM parts | Interviewing test engineers |
| 3 | DRAM Sales | How to communicate to customer based on the analysis result.   1. Check on how the customer is using DRAM on their system 2. Check on whether it is possible for the customer to upgrade the DRAM. | Interviewing sales personnel |

**3.2 Knowledge Specification**

Based on closed cases in the past, we will categorize cases based on various testing/analysis that can be done

1. Rerun=Fail AND PastResult EXISTS AND !FlowChanged -> Degrader
2. Rerun=Fail AND PastResult EXISTS AND FlowChanged -> Guardbanded Parts
3. (Rerun=Pass || PastResult NOT EXISTS) && ExtendedTemperatureFlow ==Fail -> Temperature Violation
4. (Rerun=Pass || PastResult NOT EXISTS) && ExtendedTemperatureFlow ==Pass && StringentTesting==Fail && ProductionOverkillPercentage<1.0 -> Additional testing
5. (Rerun=Pass || PastResult NOT EXISTS) && ExtendedTemperatureFlow ==Fail && StringentTesting==Pass -> Customer system analysis
6. (Rerun=Pass || PastResult NOT EXISTS) && ExtendedTemperatureFlow ==Pass && StringentTesting==Fail && ProductionOverkillPercentage>1.0 -> Requesting parts with higher specification

**4. Workflow and Features**

The rules are the basis of the KIE workflow. With the activity diagram, the process is being defined. Ruleflow group is then assigned to the rule task respectively. User QA, TestEngineer and DRAMSales are defined.

**4.1 Activity Diagram**

DRAMSales

System

TestEngineer

QA

START

Part Detail Requesting parts with higher specification

Rerun Result

Past History and Flow Changes

Guardbanded/Degraders?

Yes

No

Close as Degrader or Guardbanded Parts

Extended Temperature Flow Result

1

END

QA

TestEngineer

DRAMSales

System

1

Failing temparature flow?

Yes

Close as Temperature Violation

No

Stringent Flow Result

Passing?

No

Yes

END

Check with customer on system

Overkill Result on limited samples

Overkill < 1.0%?

Yes

END

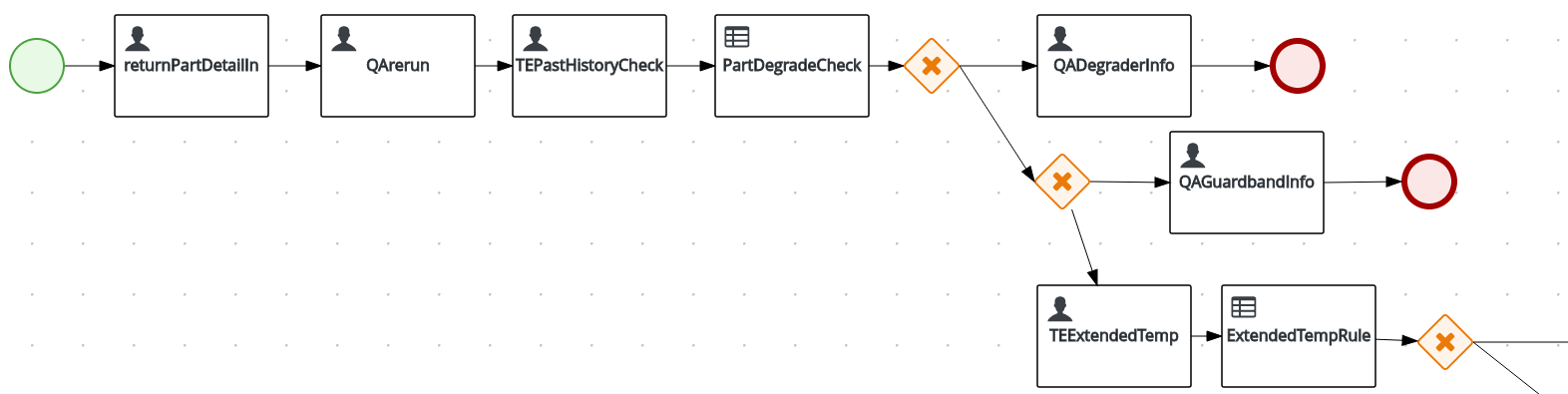
Close with additional test

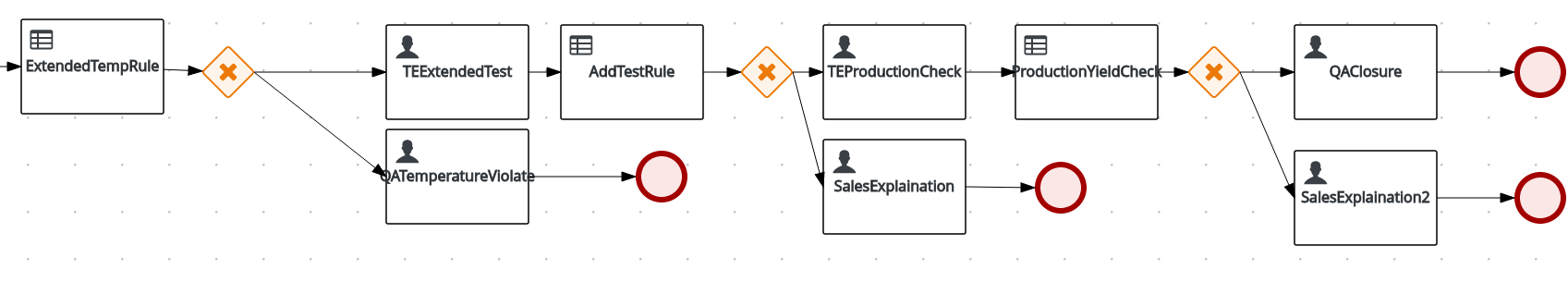
No

Check with customer on higher specifications

END

END

****

****

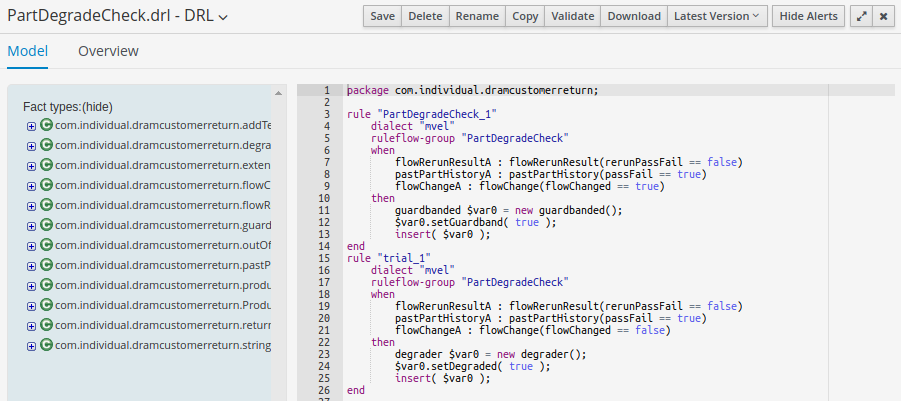
KIE Process Flow

**4.2 Data Object**

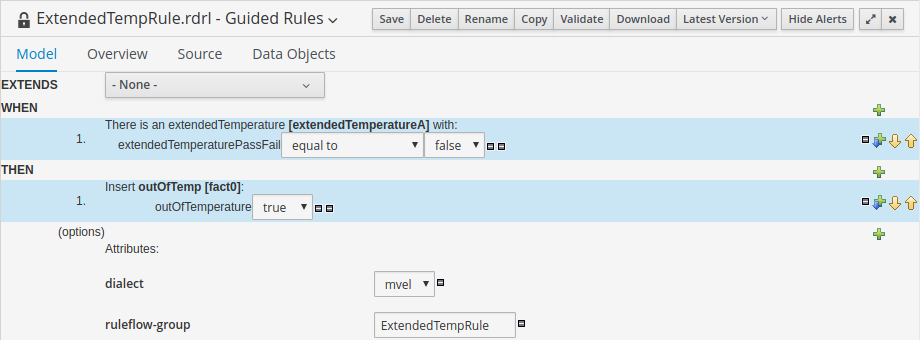
|  |  |  |  |
| --- | --- | --- | --- |
| Data Object | Field | Type | KIE Task |
| returnPartDetail | serialID | String | returnPartDetailIn |
| returnPartDetail | customerName | String | returnPartDetailIn |
| flowRerunResult | flowRerunPassFail | Boolean | QARerun |
| flowChange | flowChanged | Boolean | TEPastHistoryCheck |
| pastPartHistory | pastPartPassFail | Boolean | TEPastHistoryCheck |
| extendedTemperature | extendedTemperaturePassFail | Boolean | TEExtendedTemp |
| stringentFlow | stringentFlowPassFail | Boolean | TEExtendedTest |
| productionOverkill | productionOverkillPercent | Float | TEProductionCheck |

**4.3 Rules**

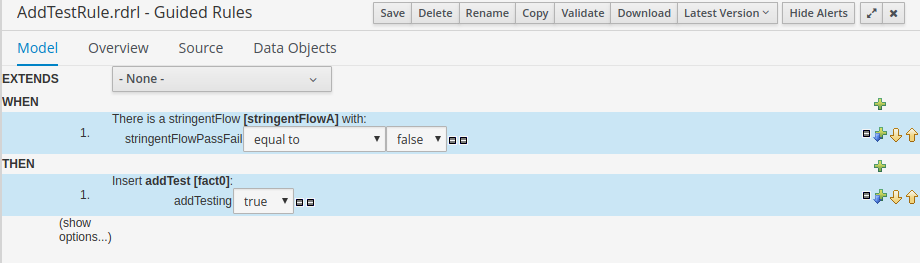
The rules obtained from the knowledge acquisition and specification are defined in drools language. Each rule is assigned with specific ruleflow-group that will also be declared in each KIE rule task. When there are multiple rules, using the DRL file. When there is single rule then using the guided DRL to make the rule definition easier.

****

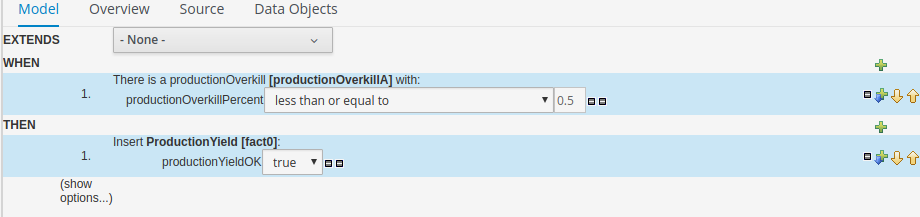
Assigned to PartDegradeCheck rule task.

****

Assigned to ExtendedTemperatureRule task.

****

Assigned to AddTestRule task.

****

Assigned to ProductionYieldCheck rule task.

**4.4 Features**

The system is designed for the QA to record the return part details and it will trigger test engineers to check on the parts.

At the beginning, QA will key in the part detail and the test re-run (on the part) result.

Test Engineer will check whether the past data are available and the part is passing.

If the part is passing re-run, the analysis will be continued by test engineers. If it fails re-run and past testing record still in test engineering database, if the testing flow remain unchanged, the part will be considered as degrader. However, if it fails re-run while the testing flow has been changed after the parts goes to customer, it will be defined as guardbanded parts. QA will get the result and enter the comment for degrader/guardbanded parts.

However, if the parts passing the re-run, analysis continues. In common, customer system can get too hot and so the parts fail. Thus, temperature range is extended for the test flow. If it fails, the part is a customer violation case. If it passes, the returned part will be going for further analysis with stringent test flow. If it passes the stringent flow, sales team should discuss with customer on whether the customer system has limitation or abusing the parts.

If it fails stringent test, the test engineer will run the stringent test on limited samples. If the stringent test will result in fallout (>1%), the sales representative should discuss with customer in getting higher performance parts and specify new requirement. If the overkill fallout is acceptable, QA can close the case while TE can continue to implement the test flow.

**5. Validation**

Validation Snapshots:

Test Case:

1. Process started. User “QA” receive task to complete form on parts’ detail.
2. “QA” will then receive next task to fill in rerun result.

(In test case, submit without tick to indicate failing the test)

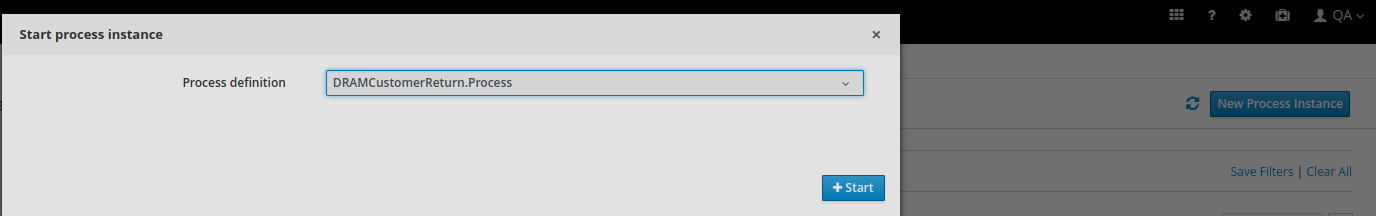
1. “TestEngineer” will then receive task to fill in past history information. “TestEngineer” should also indicates whether the flow is being changed.

(In test case, submit without tick to indicate that the flow info is not available so the process will continue on extended temperature flow.)

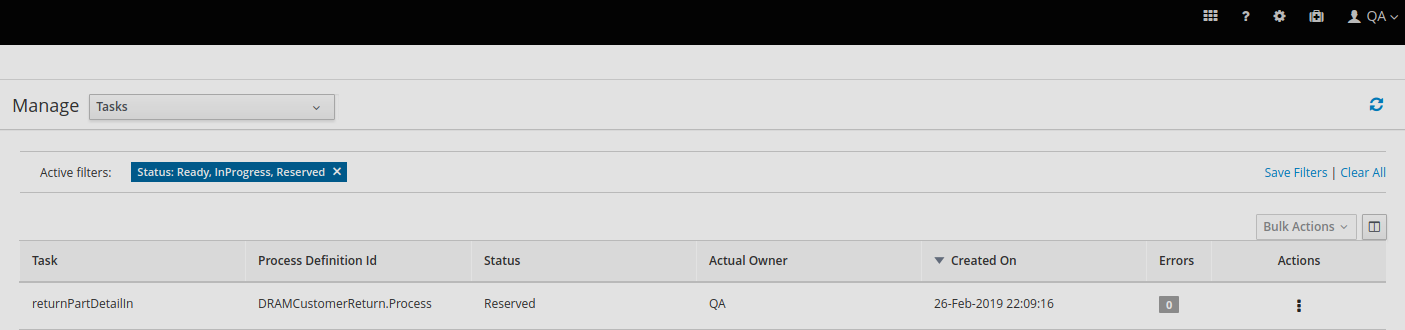
1. In the test case, it will goes to extended temperature flow after rule is firing based on info passed in step 2 and step 3.
2. “TestEngineer” will fill in extended temperature flow result. (In test case, submit with ticking the checkbox to indicate the temperature flow is passing. As a result, the part is not shown to be failing under extreme temperature, will go through stringent flow)
3. “TestEngineer” will fill in stringent flow result. (In test case, the flow is failing. Checkbox is not ticked. It will goes to next step so TestEngineer should feedback on overkill check)
4. “TestEngineer” will fill in overkill check result. (In DRAM testing, if customer returned part showing fallouts, then the test it fails should be examined thru engineering flow in production environment. This will give how much overkill failure in production environment, this is known overkill check.). In the test case, overkill check fails, hence, production test flow will not be check. “DRAMSales” should communicate with customer. Form will be sent to DRAMSales.

Validation result (Using Process instance/Tasks in Business Central):

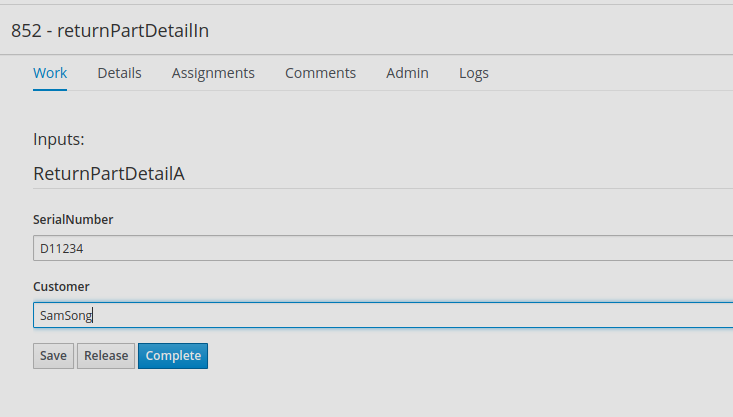
Start by user “QA”

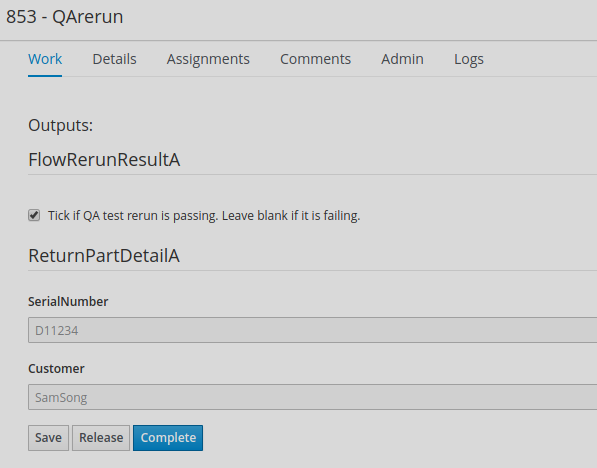


QA receives task

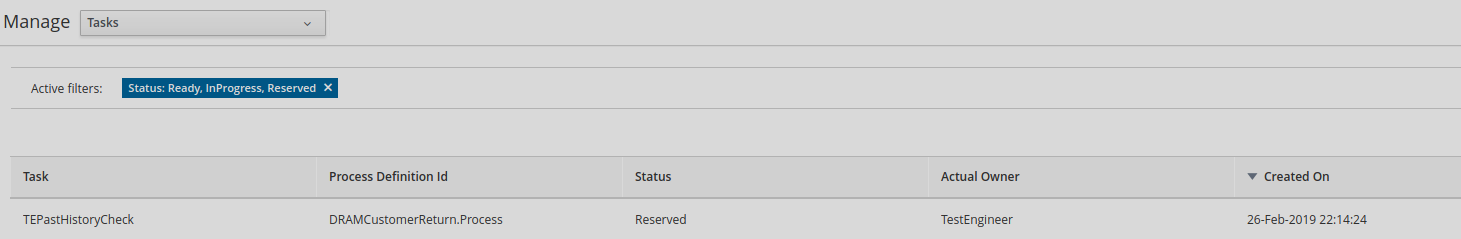


QA fills in part detail and flow rerun result

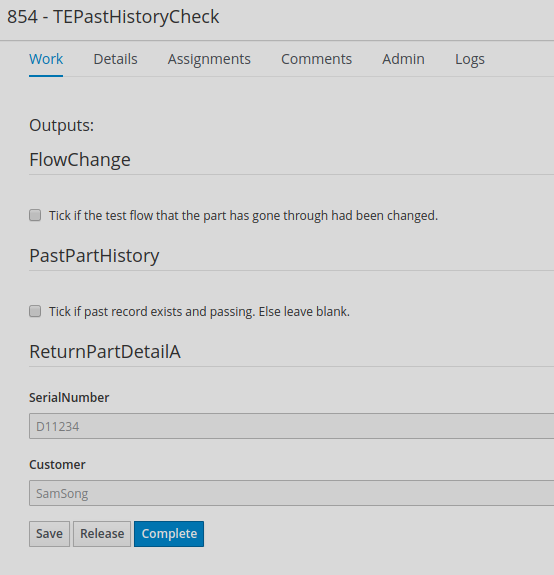


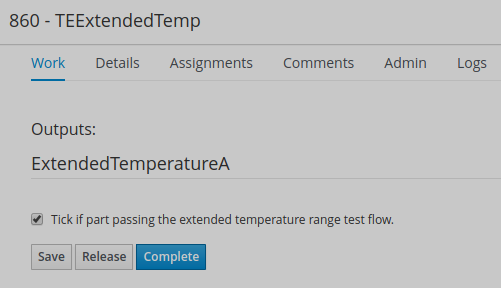


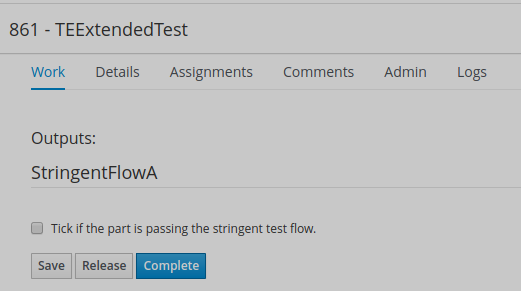
TestEngineer gets the task

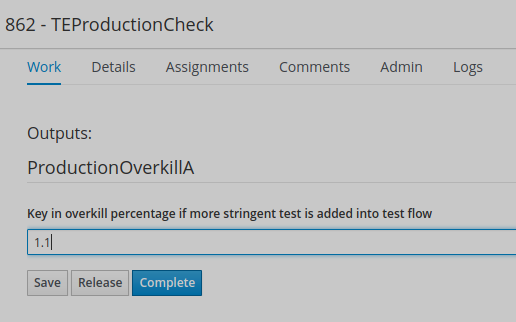


TestEngineer fills in past part history, extended temperature flow (after proved that the part is not resolved by re-run or past history given), stringent flow and production overkill result.

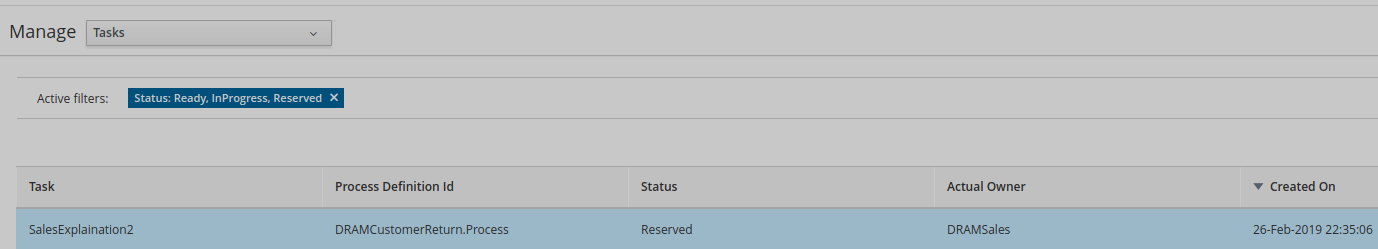


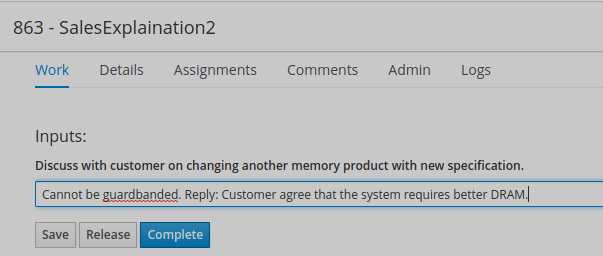






DRAMSales get the task when the additional test is having high overkill.





**6. Conclusion**

The system was validated, and it can run with the expectation. (Using KIE 7.16 downloaded from JBOSS, space name individual and project DRAMCustomerReturn).