## Chosen Changes

Scripting

Looped Scripting

Test -j

Quis custodiet?

### Motivation to not choose the other changes

All thumbs and Emulate Schemulate are not chosen since our analysis is that they would have a impact too significant to refactor within the timeframe given by the deadline.

Emulate Schemulate would require us to add Crash detection and checkpoints to the emulation component. This would break the current standard we have for seperating different functionality in different components.

With All thumbs we felt that we lack the knowledge of mobilephones and touchscreen for finding valid solutions to this change. With enough time to researcht the subject it could be solved but this does not fit the staff assumptions made by our team.

## Analyzis

There was no need to change to current factors, instead we added four new factors, one for each change. These are marked in the factor table with red. All the new factors are product factors.

To cope with the changes we have designed two new issue-cards with corresponding strategies and solutions. These are also marked with red like the new factors.

There will be some changes to the architecture to accommodate the new functionallity. Thankfully we already designed the system to handle generating new input from output received for different parts of the MIB so most changes will be localized to the testing wrapper and the new scripting component.

A scripting component will be added along with the corresponding modules for the module and execution view. It will be connected to the input/output component to enable it to feed new input into the system as well as define new output validation rules.

Running parallel tests on the same intance will require the testing wrapper to keep track of several testing techniques at the same time. With our current model this should not be a problem as long as they both have unique identifiers so that the correct input/output channels can be maintained.

All other components of the system should be unaffected on a architectual level.

## Evaluation

The evaluation will focus on the new changes made to the MIB, not the old requirements already established at the beginning of the course. the new requirements from the changes are divided into the functionality that they describe.

## Evaluation scenarios

#### Scripting

* Customer asks us to implement a specific script-language for the input-scripting.

We had already foreseen this and thus chose to make our scripting into a whole layer in the module view so that we could easily extend it with functionality to support our own scripting languages. The changes from a existing language and our own should therefore be very local. The rest is all a matter of budget and time.

* The customer wants 10 different scripting languages implemented.

This is not a realistic but doable. At an architectural level the scripting component should be able to hold an potentially indefinite number of languages, though we might have to point out the lack of gain for having so many instead of a selected few.

* How deep will the looped scripting go? Will you be able to script just for input and output

validation or will you be able to implement new rules for the other components as well?

It is not a requirement from the customer to be able to script anything besides the input and output-validation, as such it is nothing we will implement into our MIB since we do not work for free. However with the high changeability in our MIB it should be possible to some extent to implement such scripting in other components as well.

#### Parallel testing

* Output validator receives output to be validated from two parallel tests on the same tested system, how does it tell them apart?

Each testing technique will have a unique identifier when it arrives. Thus the output validator can match this identifier with the corresponding identifier that has been set at the start of the test or during scripting. It can then log the different outputs using these identifiers as well.

* We want to run 100 different test techniques in parallel.

The main concern with this scenario is the strain it would put upon the data broker for sending and receiving output/input to the different tests. We would probably have to limit the amount of parallel testing to a much smaller number so the system does not break. The number of techniques will also be dependent of the types of techniques to be run and how large their throughput are. As such it is a difficult question to answer since different customers may want different testing techniques. the maximum number can probably not be defined until a prototype can be built.

* While the MIB is running parallel testing techniques one of them crashes the tested system and the system is restarted from a checkpoint. What happens to the testing technique that did not crash the system?

All testing techniques will have to redo their testing from the last checkpoint. This will give some testing overhead but the alternative is a test that is less deterministic unless we can assume that the different techniques have no sustained impact on the tested system. The reason for this is that if a testing technique can change system variables, then the reason for the crash may not lie with the technique that triggered the crash, but instead with a previous technique that changed the system.

#### Monitoring the test

* A tester starts the MIB in automatic mode and later wants a hands on approach, thus wanting to switch into a monitoring mode.

There are two different modes for the MIB to switch between, automatic and monitored mode. The tasks given to the data broker will vary slightly depending on which mode is being run but there are no other changes to the rest of the system at this level.

* We are testing a system with a extreme throughput of data and the tester wants to run in monitored mode.

This will only change the amount of data being transferred in real-time since the testing data can no longer be stored in the datalog until the test is complete. Instead that data will have to be transferred to the output validator constantly so that a report can be made for the tester monitoring the system. Without hardware specifications and a working prototype it is hard to evaluate the effect of this increase in data.

- A tester want to only script without looking at the test results in real-time

The monitored mode will have to be able to toggle different functionalities on and off to make it possible to script without getting the testing result. A separate win for this is that it takes away the performance issues that could arise from monitoring the test results from scripting. We will add a new strategy to adapt this decision.

### Evaluation

From the scenarios we have found one new strategy that needs to be implemented. We cannot see this will have any effect on the different views and as such will only be added to the list of issue-cards we have. It could affect later stages but as of now the architectural structure remains the same. Here follows a short summary:

- It will be possible to implement our own scripting languages.

- Several different scripting languages can be implemented.

- The customer does not require us to script anything beside input and output-validation.

- Each testing technique will have a unique identifier.

- The performance issues with multiple testing techniques is hard to map without making a working prototype.

- If parallel testing is being done, all tests will have to redo their work from the last checkpoint if the tested system crashes.

- The MIB will have two different modes, one automatic and one monitored.

- In monitored mode it will have to be possible to toggle certain functionalities on and off depending on what the tester wants to do, the following strategy was created for solving this:

|  |
| --- |
| **Name:**  Toggle in monitored mode |
| **Description:**  A tester should not be forced to view the testing data if all he wants to do is script new input into the MIB. This is so that the performance loss from monitoring the test can be skipped for scripting.  **Factors:** |
| **Solution:**  We will make it possible to toggle certain functionalities in the monitored mode on and off to give the tester a higher degree of control. |
| **Strategies/Tactics:**  "function toggler" adapt certain functions to be toggled on and off. |