## Module view

### Emulations

This layer gathers the modules that emulate hardware and channel the data from software. We took the descision that these will benefit from ease of communication as well as similar data transfer to the rest of the system since an emulated software might be required to send data to software and vice versa. Data to the rest of the system is sent through the data transfer module.

### Testing

This layer holds the modules tightly connected with the actual system to be tested. When the tested system crashes the wrapper will need quick access to the checkpoint module so that the system can be restarted with minimal latency. Data to the rest of the system is sent through the data transfer module.

### Data Log

Here all the data from the testing is categorized and stored into a log-module until it can be sent up to the report module at the end of the test. Data to the rest of the system is sent through the data transfer module.

### Data Channeling

Holds the data broker subsystem that handles transfering data between different parts of the system that do not have knowledge of eachother as well as the message handler that takes care of informing the rest of the system in case of a crash in the tested system.

### Input/Output

This layer hold the module for the Input managing as well as the subsystem for validating the output and writing the report. We put the report module and the validation module in the same subsystem since the output wont have to be validated until the end of the test for the report.

## Execution view

The software platform will house two processes. One for the MIB itself and one for the tested system. These will communicate via IPC(intraprocess communication). The MIB will have one thread for each major component in the system so that they can work independently. We also mapped which resources each entity needs. The processes consumes Memory and Address-space where as the threads consume CPU-time.

In the diagram we have first described this relationship as well as the relationship with the platform resources. After this we have devided each thread-entity into single diagrams containing those modules that are coupled to that entity thus giving a better view that is not to cluttered.

The division on the thread are based on the division already made in the conceptual and the module view. Components and modules that will work closely together are grouped on the same thread.

We also have two separate entities for specific requirments on the MIB. One file-entity for the input manager since this module will require file handling. One Queue for the Data distrubutor so that it can handle the vast amounts of data that will be sent simultaneously over the MIB.