|  |
| --- |
| **Name:**  Multipleinput issue |
| **Description:**  The system must support several types of input simulations as well as being able to add new types if the costumer requires it.  **Factors:**  P1.1.1 Siulate various input types  P4.1.1 Adding new input mechanisms |
| **Solution:**  Making the interface between the module handling input and the sub modules for different input types work the same no matter what input type it is. |
| **Strategies/Tactics: "***Generalize the module" "Software Architecture in Practice Second Edition"* Chapter 5.3 Len Bass, Paul Clements, Rick Kazman 2003 |

|  |
| --- |
| **Name:**  Multiplehardware emulations issue |
| **Description:**  The system must support several types of hardware emulations as well as being able to add new types if the costumer requires it.  **Factors:**  P1.1.3 Emulate various hardware devices  P4.1.2 Adding new hardware emulations |
| **Solution:**  Explore standards for hardware communications currently used or in development to support most hardware emulations without impacting the system. |
| **Strategies/Tactics:**  **"***Anticipate expected changes" "Software Architecture in Practice Second Edition"* Chapter 5.3 Len Bass, Paul Clements, Rick Kazman 2003 |

|  |
| --- |
| **Name:**  Multiplesoftware emulations issue |
| **Description:**  The system must support several types of software emulations as well as being able to add new types if the costumer requires it.  **Factors:**  P1.1.4 Emulate various softare applications  P4.1.3 Adding new software emulations |
| **Solution:**  Explore standards for software communications currently used or in development to support most hardware emulations without impacting the system. |
| **Strategies/Tactics:**  **"***Anticipate expected changes" "Software Architecture in Practice Second Edition"* Chapter 5.3 Len Bass, Paul Clements, Rick Kazman 2003 |

|  |
| --- |
| **Name:**  Multipletesting techniques issue |
| **Description:**  The system must support several types of testing techniques as well as being able to add new types if the costumer requires it.  **Factors:**  P4.1.4 Adding new testing techniques |
| **Solution:**  Keeping the semantics of the testing modules coherent so that further testing techniques can be added with minimal changes to the current structure |
| **Strategies/Tactics:**  "Maintain semantic coherence" and **"***Anticipate expected changes" "Software Architecture in Practice Second Edition"* Chapter 5.3 Len Bass, Paul Clements, Rick Kazman 2003 |

|  |
| --- |
| **Name:**  Not crashing with tested system issue |
| **Description:**  The MIB must not crash just because the tested system crashes. This means that the input must be stalled until the tested system is running again.  **Factors:**  P4.2.1 Reliable input and output components |
| **Solution:**  The input component must wait for clerance before sending the input through to the next component. This way its function will not be affected by a crash since it is always waiting before sending input. Similarly the output and log components will also be passive and wait for information instead of requesting it, thus not breaking the system. |
| **Strategies/Tactics:** |

|  |
| --- |
| **Name:**  Creating report issue |
| **Description:**  The MIB needs to be able to create a report once the testing is done. This report must contain data from both the output component as well as the log for all the test data.  **Factors:**  P1.1.2 Identify and compare output  P1.1.5 Logging all test data |
| **Solution:**  There must be a connection between the component handling the output varification and the component that logs all the test data so that data can be sent between them to be combined into the final report at the test end. |
| **Strategies/Tactics:** |

|  |
| --- |
| **Name:**  Keeping the system running through a test crash |
| **Description:**  The MIB needs to be able to standby further testing if the tested system crashes until it has been restarted from a earlier point.  **Factors:**  P4.2.2 Restart tested system on crash |
| **Solution:**  A component wrapped around the tested system will record the state of the tested system at regular intervalls. If a crash occurs the component will restart the system using the latest checkpoint as refrence. |
| **Strategies/Tactics:**  “Checkpoint/Rollback” *"Software Architecture in Practice Second Edition"* Chapter 5.2 Len Bass, Paul Clements, Rick Kazman 2003 |

|  |
| --- |
| **Name:**  Running system on all standard computers with required performance |
| **Description:**  The MIB needs to be able to run on all standard computers on the market which have enough performance to run both the MIB itself and the tested system.  **Factors:**  O3.1.1 Development platform |
| **Solution:**  Making the MIB cross-platform compliant |
| **Strategies/Tactics:** |