# Multi Agent System for Autonomy in Testing and Verification

*Final Presentation*

11/12/2013

Quirien Wijnands
ESA/ESTEC – TEC-SWG

# Outline

- **Some history**
- **MASATV**
  - Study Objectives and Tasks
  - Software Quality
  - The MASATV Test Process
  - The Demonstrator
- **Conclusions**
- **Questions**

# Some History

# DAFA History (1/2)

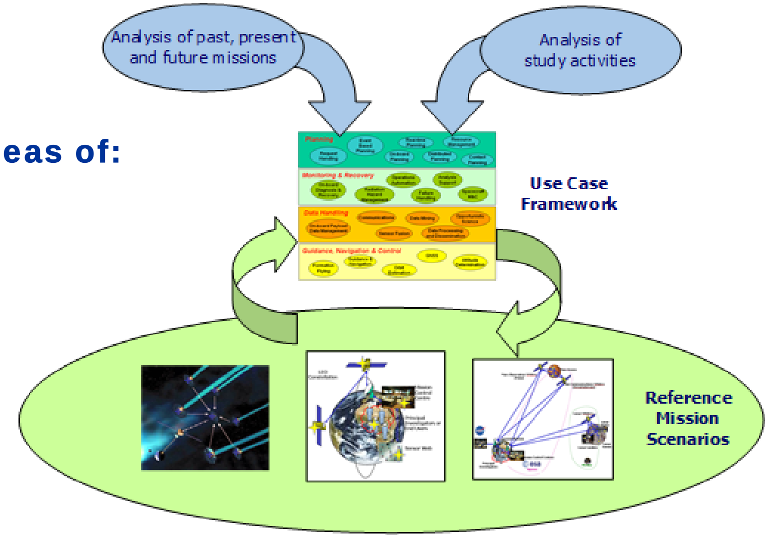
- In recent past, 2 parallel studies were done on **distributed agents for space autonomy (DAFA) to** demonstrate the advantages of using Distributed Agents in Space:
  - To demonstrate that it ***can*** be applied.
  - To identify an **appropriate methodology** for system design of agent-based systems.
  - To demonstrate the **added value** by applying a MAS-autonomy framework in a reference scenario.
  - In this respect:
    - Mission/System level including **Ground** and **Space Segment**
    - Mainly focused at **operational phases.**

# DAFA History (2/2)

- **The DAFA tasks**:
  - Survey of **use of autonomy** in space missions
  - Determination of **performance parameters** for measuring eventual improvements.
  - Survey of existing **Agent Frameworks** for development and execution.
  - **Design** of MAS for different suitable mission scenarios.
  - **Demonstrator and comparison**

**Identified Use Cases in areas of:**

- Planning and Scheduling
- Data Handling
- GNC
- Monitoring and Recovery



# Agent: one definition

- In these studies **an agent** is defined as:
  - **a stand-alone software entity,**
  - defined in terms of **behaviours,**
  - which is capable of acting with a certain degree of **intelligence and autonomy** in order to accomplish tasks.

> - In its **simplest form agents** can be seen as **traditional software functions or programs.**
> - **However agents usually are more complex and exhibit rational behaviour** such as to “maximise expected outcome”.
> - This definition can include concepts like **persistency, autonomy, social ability and reactivity and proactivity** (including capabilities like **adaptation** to the environment **and learning).**
> - A single agent can be used to accomplish a single task but more interesting are systems in which a **number of agents** are interacting with each other **(a Multi Agent System (MAS))**, and as such demonstrate **overall** intelligence (autonomy and automation).

# MASATV
# Study Objectives

European Space Agency

# MASATV: Objectives

- Application of MAS into the **S/W engineering process**.
- **Problem area: Software Systems' Testing and Verification:**
  - Major phase in S/W development life-cycle (often underestimated).
  - Time Consuming but also often under time pressure.
  - Elaborate, multi-domain, multi-level and complex.
  - Expensive to cover ALL s/w requirements and exclude ALL error sources.
  - Manpower allocation (type of tester vs. need for knowledge).
- **Possible Solution: Multi-Agent System for bringing intelligence and automation to the Testing and Verification process** (e.g. Test sequence optimalisation or failure diagnosis)

- **Objective:** Definition, Development and Demonstration of a MAS bringing Autonomy to the Testing and Verification phase of a Software Product.

T · · Systems · NLR

# MASATV: Original Main Tasks

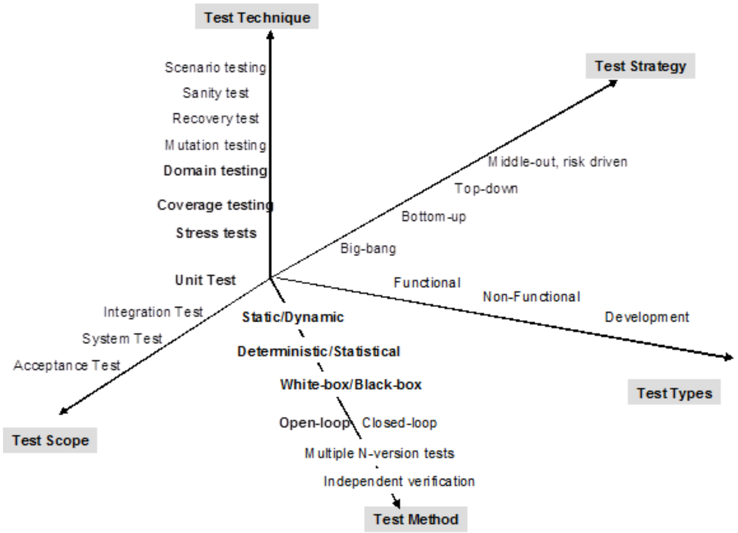
- **Task 1:** Analysis and identification of **types** of software tests and corresponding procedures most suitable for being supported.

- **Task 2:** Definition of **knowledge-base** and **Ontology**.

- **Task 3: MASTV design** using the "JADE" methodology.

- **Task 4: Implementation** of the rule-based **behaviours** of the Agents of the MASTV.

- **Task 5: Demonstration** by testing a simple Equipment Model Simulation Model.




**(non-orthogonal) Test Qualifications**

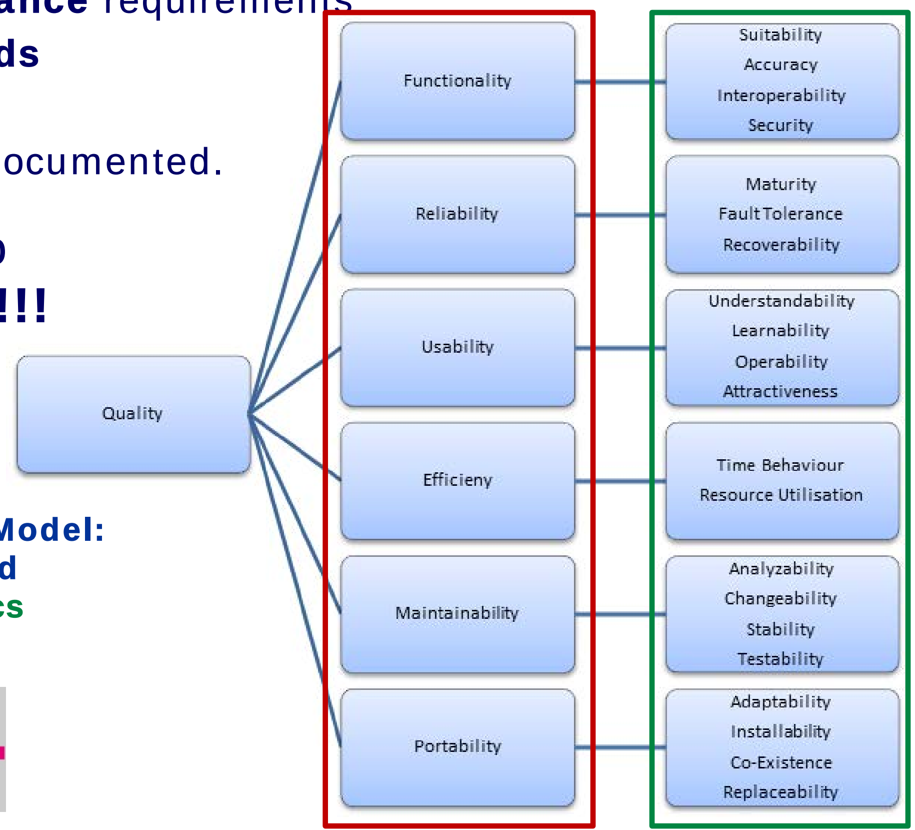
# MASATV

# Software Quality (results of Task 1)

European Space Agency

# Software Quality (1/4)

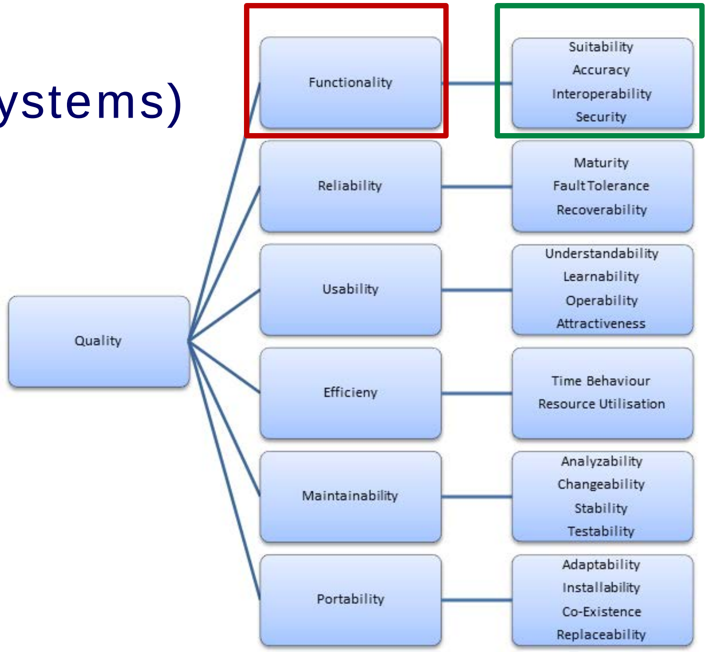
- **First we had to make some steps back:**
- The definition of **Software Quality** states that it's a **characteristic** of a product to have the **capability to satisfy needs** which are either *stated needs* or *implied needs* [ISO 8402].
  - Stated needs:
    - Documented **functional** or **performance** requirements
    - Documented **development standards**
  - Implied needs:
    - **Expected** characteristics but rarely documented.
- **Software Quality Models:** a way to **define**, **understand** and **measure !!!** software quality.




**Software Quality Model: Characteristics and Sub-Characteristics [ISO / IEC 9126]**

# Software Quality (2/4)

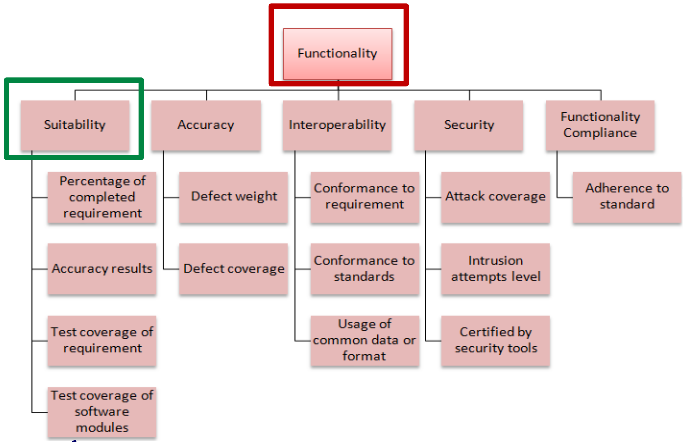
- **E.g: Functionality Characteristic**: For a software function or module to be high in quality aspects, it should be **well satisfying functional requirements**. It should perform its implied needs or stated needs. Functionality characteristic is one of the most important high-level characteristics. It contains the following **sub-characteristics**:
  - **Suitability** (how software functions comply to the services it needs to performs)
  - **Accuracy** (matching results versus reference results)
  - **Security** (how secure from misuse)
  - **Interoperability** (interfacing to other systems)




**Software Quality: Characteristics and Sub-Characteristics [ISO/IEC 9126]**

# Software Quality (3/4)

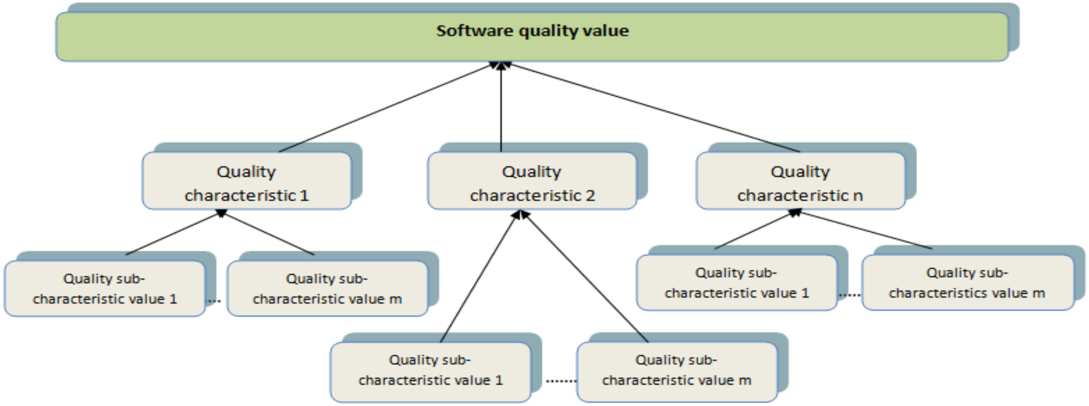
- **Metrics**: Needed for evaluation of each Sub-Characteristic.




**Metrics for Characteristic: Functionality Sub-Characteristic: Suitability**

| Functionality Characteristics | Metrics | Formula/Rules/Logic | References |
|---|---|---|---|
| Suitability (S) | • Percentage of completed requirement (S1)<br>• Accuracy result (S2)<br>• Test coverage of requirement (S3)<br>• Test coverage of software modules (S4) | S1 = completed requirement / total number requirement<br>The closer the S1 value is to 1, the better it is<br><br>S2 = Accuracy (calculated below)<br>The closer the S2 value is to 1, the better it is<br><br>S3 = Sum of all requirements covered* / total number of requirement.<br>The closer the S3 value is to 1, the better it is<br><br>S4 = Sum of all software modules covered** / total no. of software modules.<br>The closer the S4 value is to 1, the better it is.<br><br>*Requirement covered R1= no. of test case passed for R1/total no. of test cases for R1<br><br>**Software modules covered (1/0)= if there is a requirement which tests SM then its value is 1 else 0 | [BF09]<br>[ISTQB] |

# Software Quality (4/4)




**General model to evaluate software quality [Balz98]**

- **Steps** to follow:
  1. **Identify Characteristics** and **Sub-characteristics** of interest (depending on project information)
  2. Chose **metrics** (for evaluation of the Sub-Characteristics)
  3. Chose the **Evaluation Method** and **values** for the **metrics** (formula, fuzzy logic, ...)
  4. Combine **values to get quality of the characteristic** and put into matrix
  5. **Determine the rules for the quality evaluation**

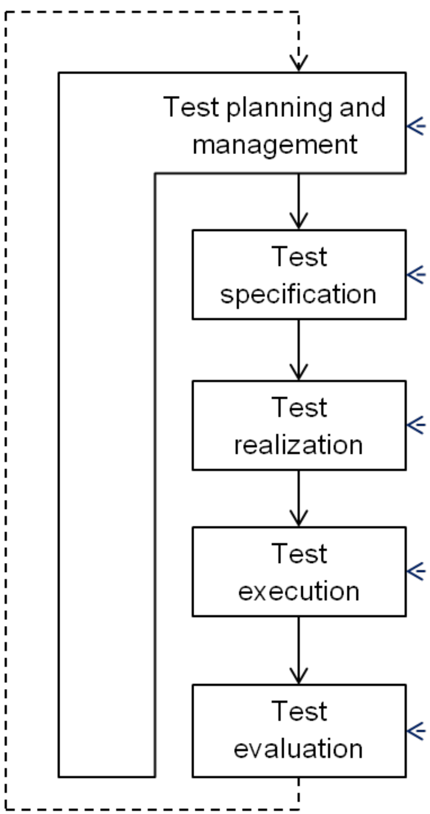
# MASATV

# The identified Test Process (results of Task 2 and 3)

# Test Process: possibilities for support by MASTV

MASATV demonstrator

**Test process**




Test case prioritization
Software quality determination

Test coverage determination

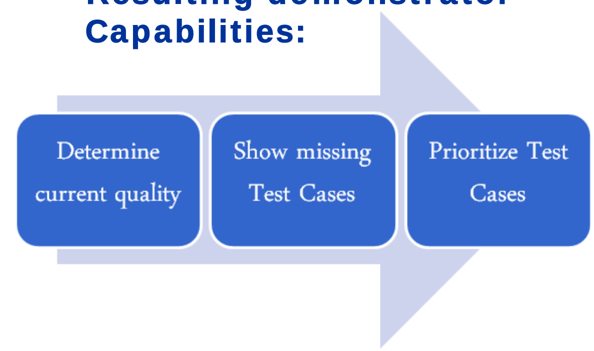
Test case generation
Test script generation

Test case execution
Test execution monitoring

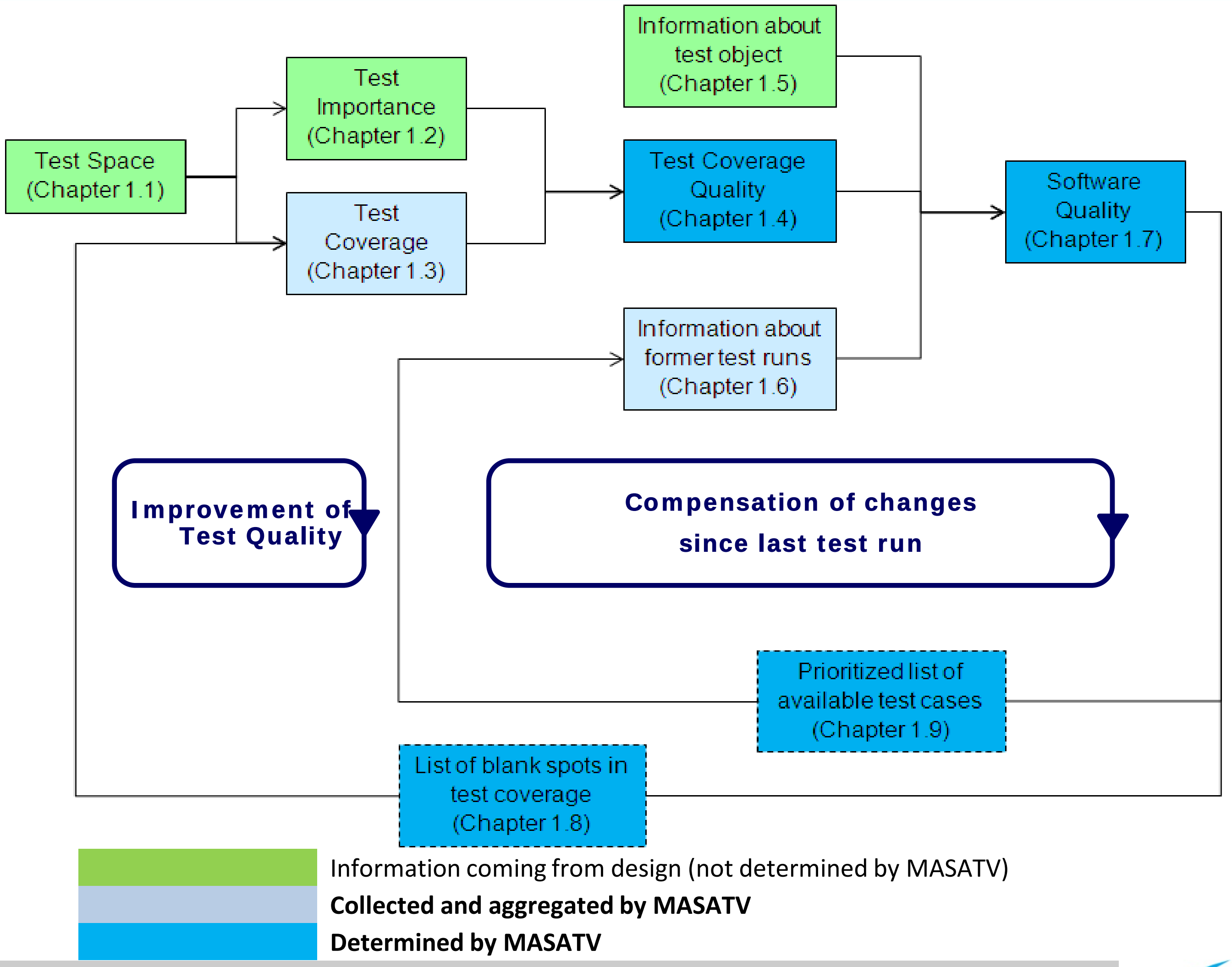
Test protocol analysis
Fault detection

Possibly supported by Test Automation

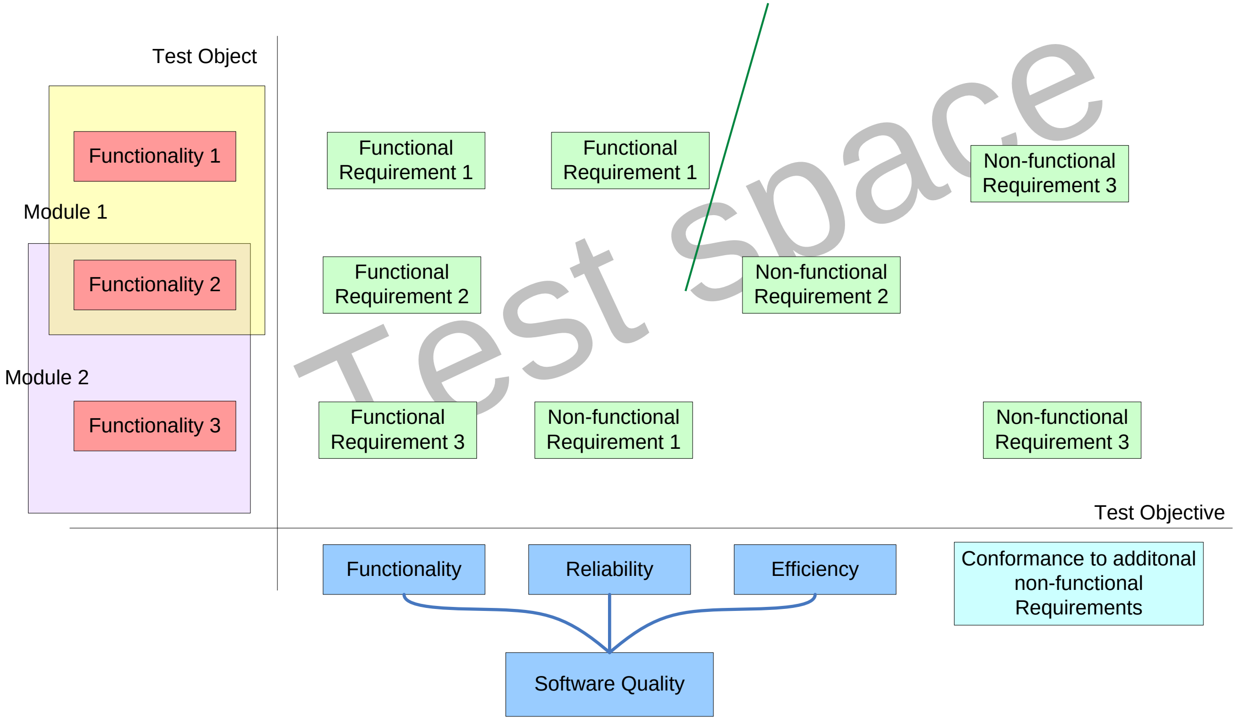
**Resulting demonstrator Capabilities:**

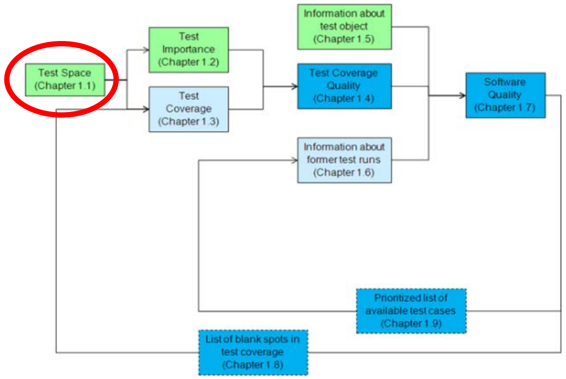


# Test Process and MASTV Main Functionalities



# Test Process and MASTV Information: The Test Space



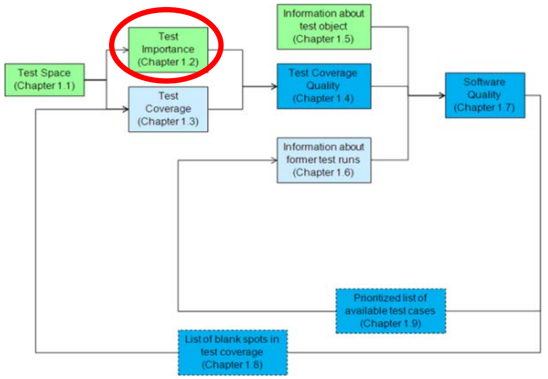




**MASTV Test Process**

- Functionalities derived from requirements which are to be tested
- Requirements (functional and non-functional)
- Software Quality aspects as defined in ISO/IEC 9126
- Additional test objectives that are not contained within ISO/IEC 9126

# Test Process and MASTV Information: The Test Importance

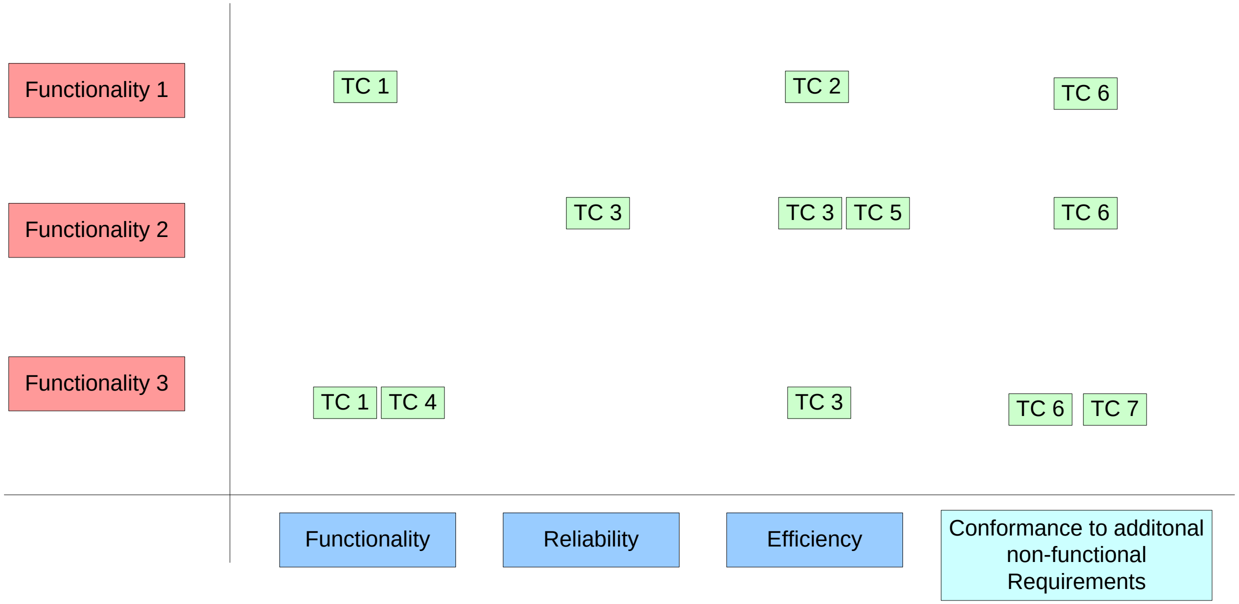
| | Functionality | Reliability | Efficiency | Conformance to additonal non-functional Requirements |
|---|---|---|---|---|
| Functionality 1 | 1.0 | 0.8 | 0.0 | 0.9 |
| Functionality 2 | 0.8 | 0.0 | 1.0 | 0.0 |
| Functionality 3 | 1.0 | 0.6 | 0.0 | 0.2 |

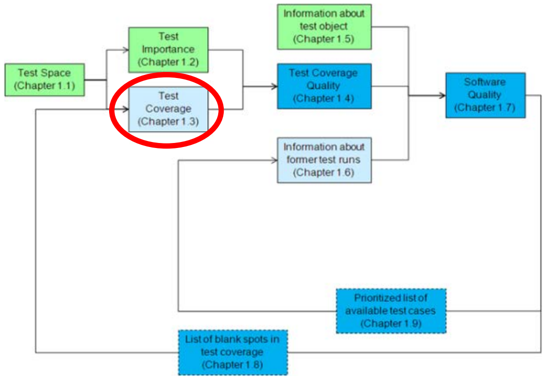



**MASTV Test Process**

- Contains the **importance** of every quality aspect for every functionality
- Is determined during the **Design Process** and **immediately** captured

# Test Process and MASTV Information: The Test Coverage



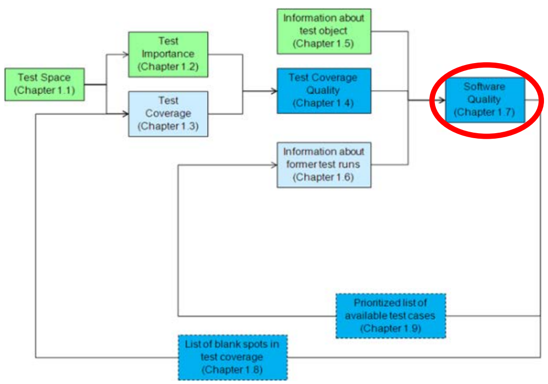




**MASTV Test Process**

- Assigns a **test-case (TC)** to one or several places in the Test Space.
- Contains the information which **Test Case** is testing which quality aspect of which functionality.
- Assignment is given by Test Engineer

# Test Process and MASTV
# Resulting: The Software Quality Matrix

| | Functionality | Reliability | Efficiency | Conformance to additonal non-functional Requirements |
|---|---|---|---|---|
| Functionality 1 | + | − − | + | − |
| Functionality 2 | − | + | − | + |
| Functionality 3 | + + | + | − | − |

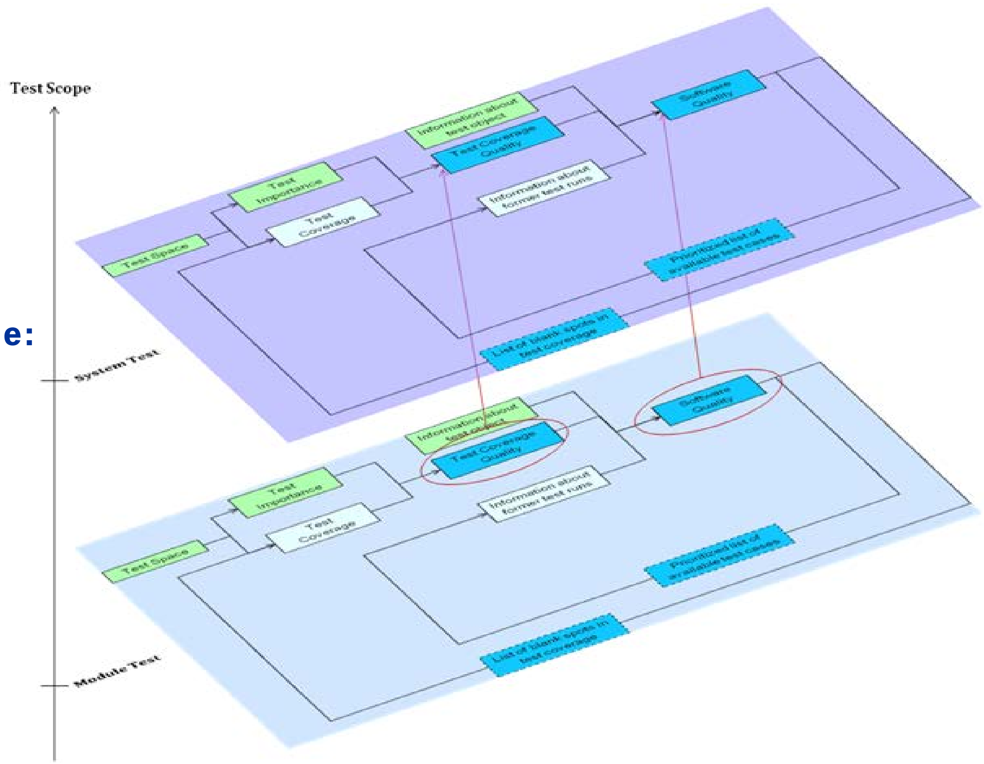


MASTV Test Process

- The Software Quality gives the actual quality of functionalities
- The Quality determination is the result of the combination of the previous matrices.

# Test Process and MASTV
# Integration of different levels of Test Scope

**“traditional” Test Scope:**
- **System Tests**
- **Integration Tests**
- **Unit Tests**



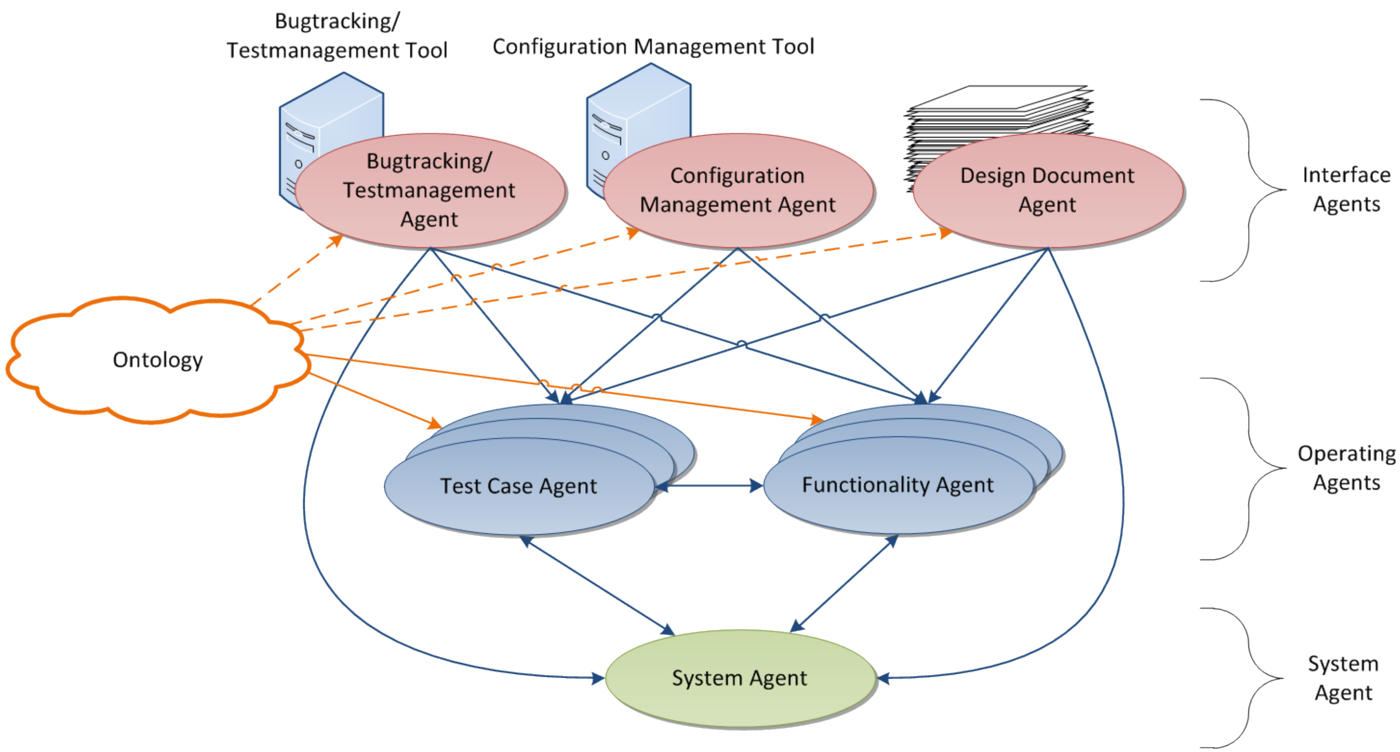

- For every test level, the Test Space can be generated and evaluated.
- The test levels depend on each other.
- In particular resulting data from lower levels is considered in higher levels

# MASATV

## The Demonstrator
## (results of Task 4 and 5)

European Space Agency

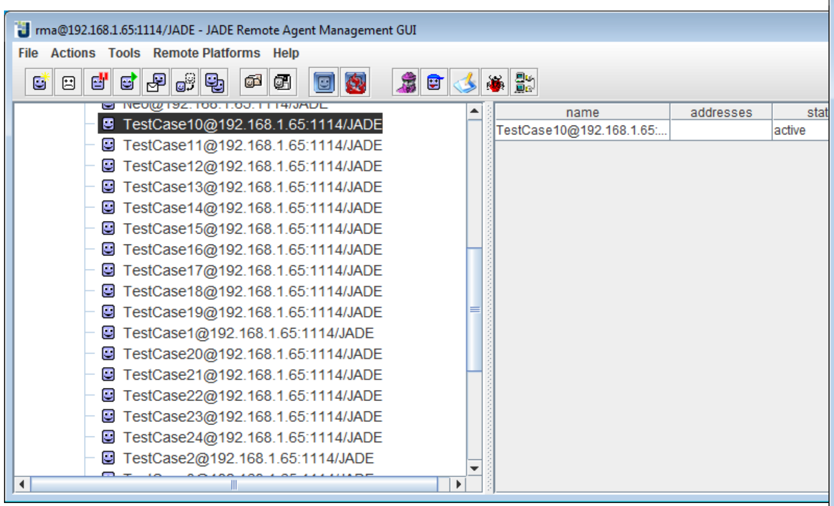
# MASTV
# MAS Architecture




**MASATV: different types of agents for different roles**

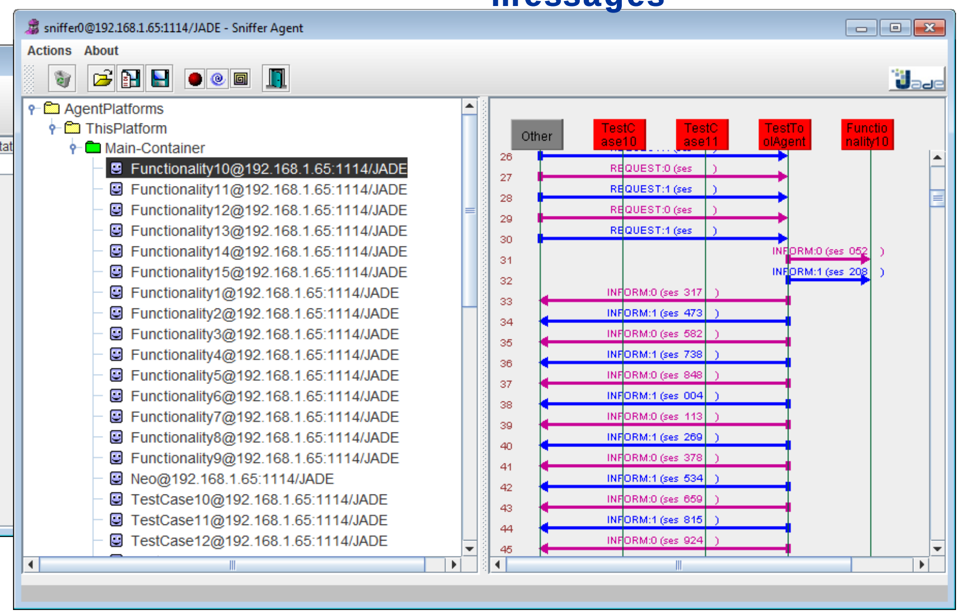
# MASTV
# The Implemented Prototype (1/3)

- Implementation in Eclipse using JAVA and using the JADE Agent Framework:
  - **Functionality Agent**: 7 behaviours, 418 LOC
  - **System Agent**: 5 behaviours, 467 LOC
  - **TestCase Agent**:8 behaviours, 485 LOC
  - **TestToolAgent**: 10 behaviours, 658 LOC

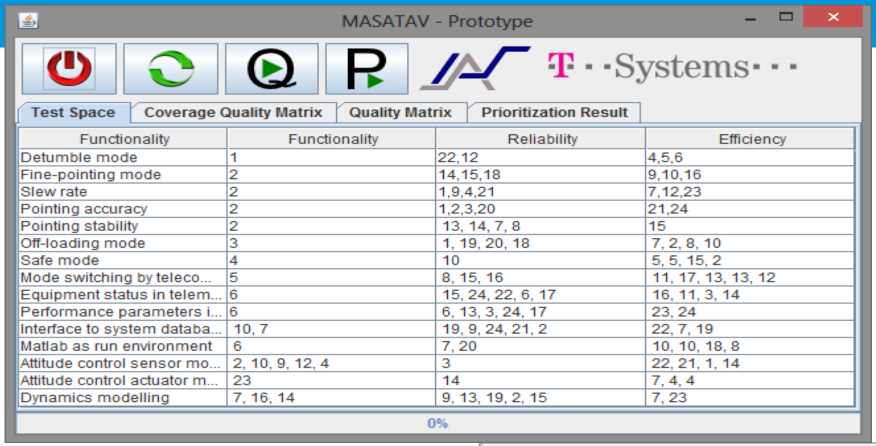



JADE execution environment

JADE sniffer and inter-agent messages

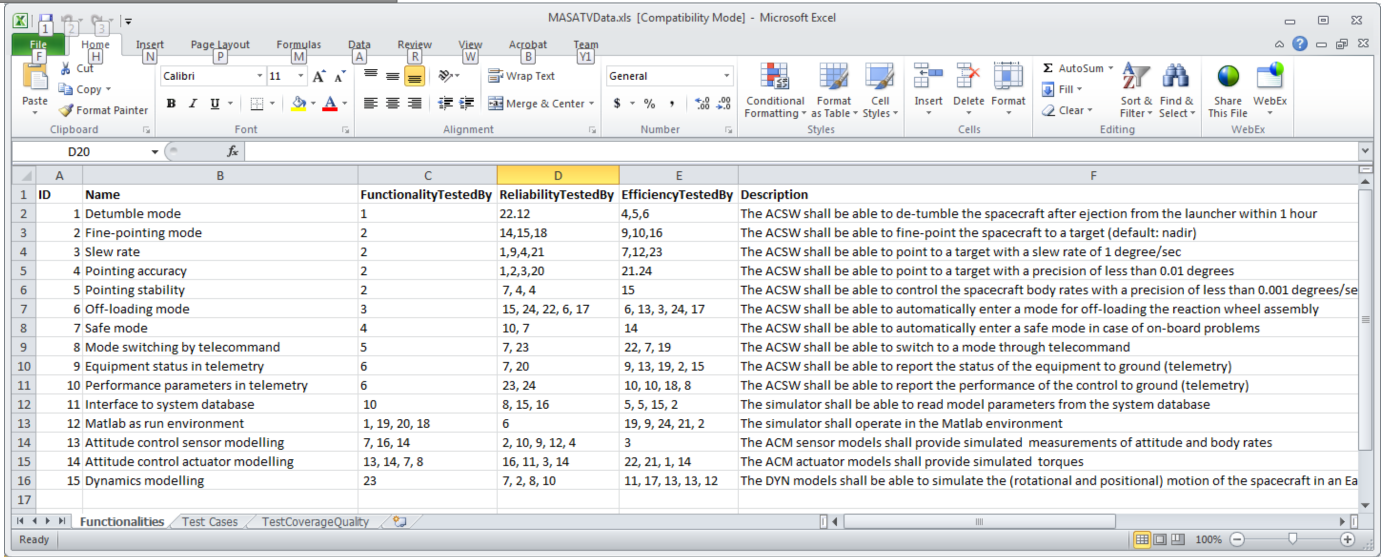


# MASTV
# The Implemented Prototype (2/3)




| Functionality | Functionality | Reliability | Efficiency |
|---|---|---|---|
| Detumble mode | 1 | 22,12 | 4,5,6 |
| Fine-pointing mode | 2 | 14,15,18 | 9,10,16 |
| Slew rate | 2 | 1,9,4,21 | 7,12,23 |
| Pointing accuracy | 2 | 1,2,3,20 | 21,24 |
| Pointing stability | 2 | 13, 14, 7, 8 | 15 |
| Off-loading mode | 3 | 1, 19, 20, 18 | 7, 2, 8, 10 |
| Safe mode | 4 | 10 | 5, 5, 15, 2 |
| Mode switching by teleco... | 5 | 8, 15, 16 | 11, 17, 13, 13, 12 |
| Equipment status in telem... | 6 | 15, 24, 22, 6, 17 | 16, 11, 3, 14 |
| Performance parameters i... | 6 | 6, 13, 3, 24, 17 | 23, 24 |
| Interface to system databa... | 10, 7 | 19, 9, 24, 21, 2 | 22, 7, 19 |
| Matlab as run environment | 6 | 7, 20 | 10, 10, 18, 8 |
| Attitude control sensor mo... | 2, 10, 9, 12, 4 | 3 | 22, 21, 1, 14 |
| Attitude control actuator m... | 23 | 14 | 7, 4, 4 |
| Dynamics modelling | 7, 16, 14 | 9, 13, 19, 2, 15 | 7, 23 |

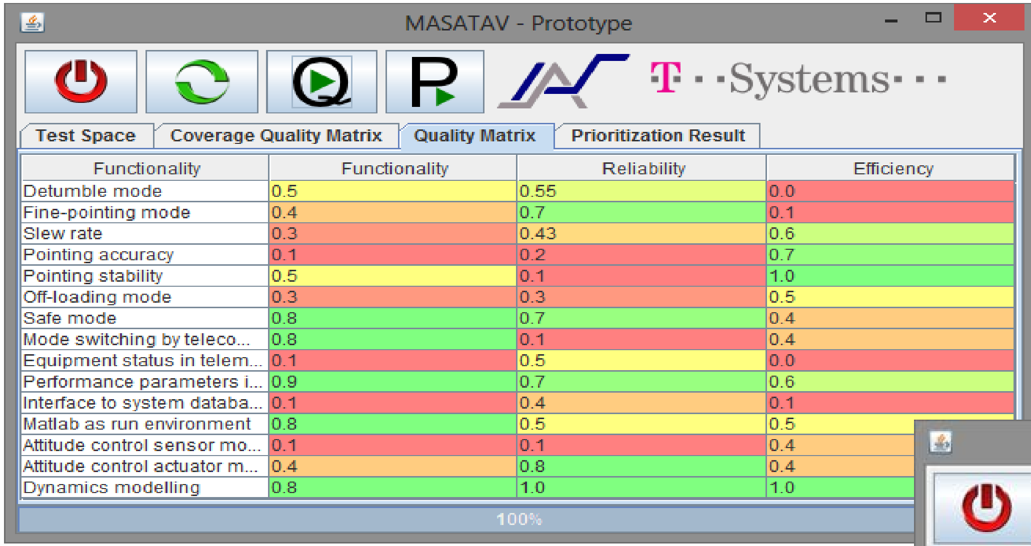
**GUI on top of the System Agent**




| ID | Name | FunctionalityTestedBy | ReliabilityTestedBy | EfficiencyTestedBy | Description |
|---|---|---|---|---|---|
| 1 | Detumble mode | 1 | 22.12 | 4,5,6 | The ACSW shall be able to de-tumble the spacecraft after ejection from the launcher within 1 hour |
| 2 | Fine-pointing mode | 2 | 14,15,18 | 9,10,16 | The ACSW shall be able to fine-point the spacecraft to a target (default: nadir) |
| 3 | Slew rate | 2 | 1,9,4,21 | 7,12,23 | The ACSW shall be able to point to a target with a slew rate of 1 degree/sec |
| 4 | Pointing accuracy | 2 | 1,2,3,20 | 21.24 | The ACSW shall be able to point to a target with a precision of less than 0.01 degrees |
| 5 | Pointing stability | 2 | 7, 4, 4 | 15 | The ACSW shall be able to control the spacecraft body rates with a precision of less than 0.001 degrees/se |
| 6 | Off-loading mode | 3 | 15, 24, 22, 6, 17 | 6, 13, 3, 24, 17 | The ACSW shall be able to automatically enter a mode for off-loading the reaction wheel assembly |
| 7 | Safe mode | 4 | 10, 7 | 14 | The ACSW shall be able to automatically enter a safe mode in case of on-board problems |
| 8 | Mode switching by telecommand | 5 | 7, 23 | 22, 7, 19 | The ACSW shall be able to switch to a mode through telecommand |
| 9 | Equipment status in telemetry | 6 | 7, 20 | 9, 13, 19, 2, 15 | The ACSW shall be able to report the status of the equipment to ground (telemetry) |
| 10 | Performance parameters in telemetry | 6 | 23, 24 | 10, 10, 18, 8 | The ACSW shall be able to report the performance of the control to ground (telemetry) |
| 11 | Interface to system database | 10 | 8, 15, 16 | 5, 5, 15, 2 | The simulator shall be able to read model parameters from the system database |
| 12 | Matlab as run environment | 1, 19, 20, 18 | 6 | 19, 9, 24, 21, 2 | The simulator shall operate in the Matlab environment |
| 13 | Attitude control sensor modelling | 7, 16, 14 | 2, 10, 9, 12, 4 | 3 | The ACM sensor models shall provide simulated measurements of attitude and body rates |
| 14 | Attitude control actuator modelling | 13, 14, 7, 8 | 16, 11, 3, 14 | 22, 21, 1, 14 | The ACM actuator models shall provide simulated torques |
| 15 | Dynamics modelling | 23 | 7, 2, 8, 10 | 11, 17, 13, 13, 12 | The DYN models shall be able to simulate the (rotational and positional) motion of the spacecraft in an Ea |

**MASATV prototype information data file: defining which SW functionality is tested by which TC for which Quality Characteristic**

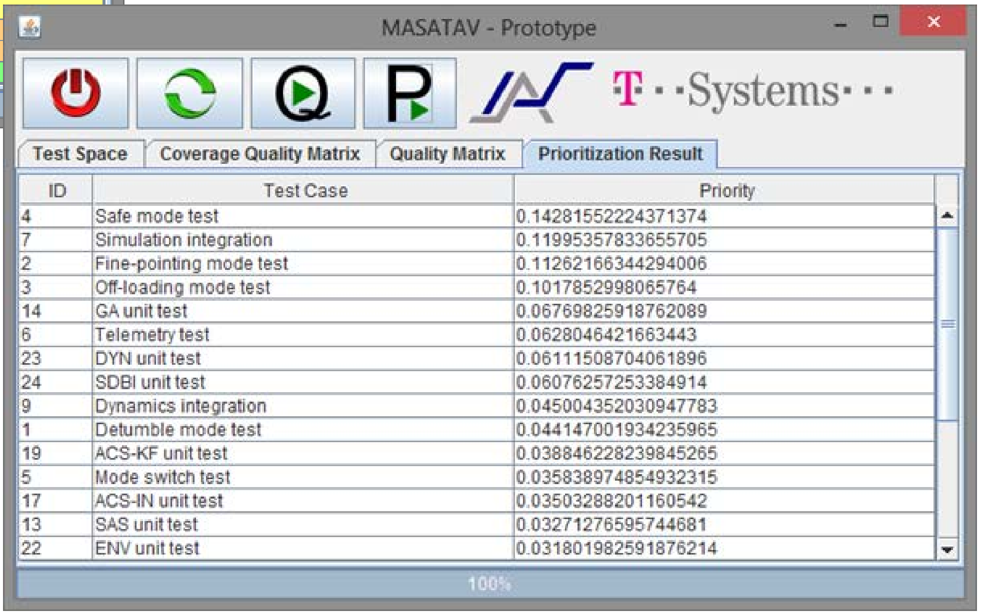
# MASTV
# The Implemented Prototype (3/3)




| Functionality | Functionality | Reliability | Efficiency |
|---|---|---|---|
| Detumble mode | 0.5 | 0.55 | 0.0 |
| Fine-pointing mode | 0.4 | 0.7 | 0.1 |
| Slew rate | 0.3 | 0.43 | 0.6 |
| Pointing accuracy | 0.1 | 0.2 | 0.7 |
| Pointing stability | 0.5 | 0.1 | 1.0 |
| Off-loading mode | 0.3 | 0.3 | 0.5 |
| Safe mode | 0.8 | 0.7 | 0.4 |
| Mode switching by teleco... | 0.8 | 0.1 | 0.4 |
| Equipment status in telem... | 0.1 | 0.5 | 0.0 |
| Performance parameters i... | 0.9 | 0.7 | 0.6 |
| Interface to system databa... | 0.1 | 0.4 | 0.1 |
| Matlab as run environment | 0.8 | 0.5 | 0.5 |
| Attitude control sensor mo... | 0.1 | 0.1 | 0.4 |
| Attitude control actuator m... | 0.4 | 0.8 | 0.4 |
| Dynamics modelling | 0.8 | 1.0 | 1.0 |

**Resulting Quality Matrix for the Simulation Model**

**Resulting Test Cases Prioritization Result**




| ID | Test Case | Priority |
|---|---|---|
| 4 | Safe mode test | 0.14281552224371374 |
| 7 | Simulation integration | 0.11995357833655705 |
| 2 | Fine-pointing mode test | 0.11262166344294006 |
| 3 | Off-loading mode test | 0.1017852998065764 |
| 14 | GA unit test | 0.06769825918762089 |
| 6 | Telemetry test | 0.0628046421663443 |
| 23 | DYN unit test | 0.06111508704061896 |
| 24 | SDBI unit test | 0.06076257253384914 |
| 9 | Dynamics integration | 0.045004352030947783 |
| 1 | Detumble mode test | 0.044147001934235965 |
| 19 | ACS-KF unit test | 0.038846228239845265 |
| 5 | Mode switch test | 0.035838974854932315 |
| 17 | ACS-IN unit test | 0.03503288201160542 |
| 13 | SAS unit test | 0.03271276595744681 |
| 22 | ENV unit test | 0.031801982591876214 |

· · T · · Systems · · · · · · · · · · · · · · · · · · · · · NLR

# MASATV

# Conclusions

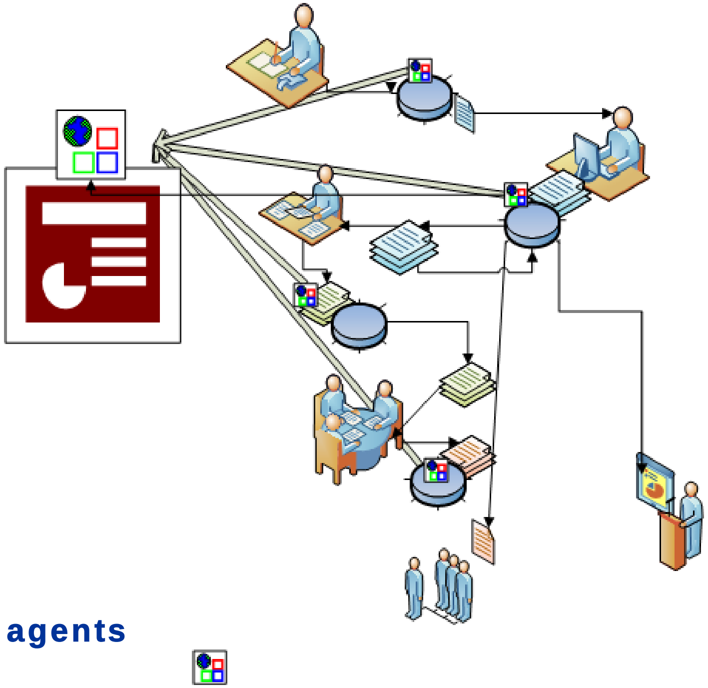
# Conclusions

- An **Agent-supported Test Process** has been defined based on **Software Quality**, using **Characteristics**, **sub-Characteristics** and (configurable) **Metrics**.

- In the implementation of the demonstrator use is made of **Agents** for:
  - **Mastering the Complexity**
    - Solving complex tasks by distributing them to different agents (for which each can be relatively simple).
    - By cooperation of different Agents the task is solved.
    - The Agent Knowledge base is populated by human knowledge and experience.
  - Its "easy" **Expandability**, by adding new (relatively simple) Agents. In this respect the quality of the **Ontology** is important.
  - Allowing **integration** of distributed information sources and **classic CASE** tooling.

# Outlook

- Merging with **ECSS S/W development** standard.
- Which **type of data** is available at which **moment** and is coming from which **entity**.
  - The format here is less important
- **Collaboration/Integration** with other used tooling

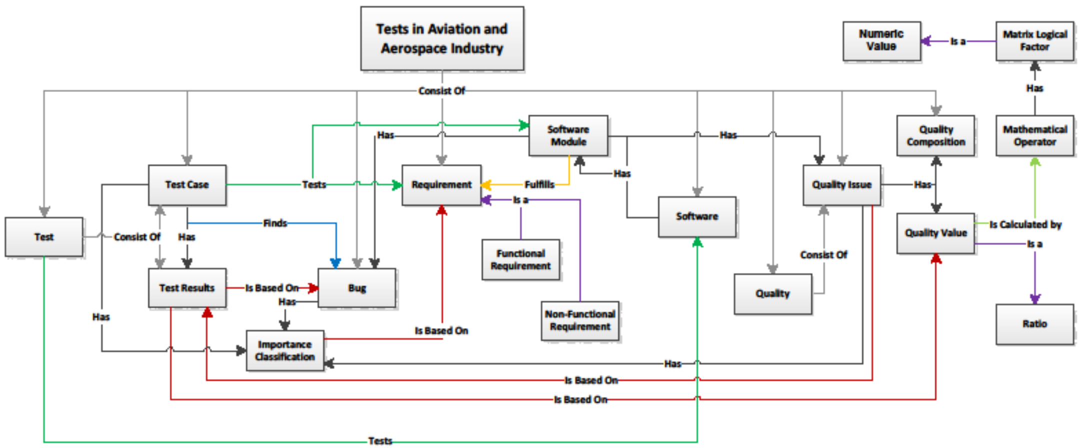


# MASATV

# Questions?

*For more information please contact:*
quirien.wijnands@esa.int

# MASTV
# The used Ontology




**MASATV: Ontology for exchange of information**