

# Mathmet Software Quality Assurance Plan

Draft Version 6. See Disclaimer in section 7

## 1. Software details

- Enter the software name, a brief description etc. Mandatory fields are indicated by \*
- **Customer** is not always easy to identity. See the [accompanying document](#) to this software quality assurance plan.
- **Software location**, for example, could be a Git repository or an MS-Windows folder.

Software name *	
Brief description *	
Developer(s)	
Customer	
Software location *	

## 2. Document control

This section concerns the plan itself, not the software.

Status (choose)	DRAFT	ISSUED
Version of plan		
Date of plan		
Plan prepared by		

## 3. Software integrity level (SWIL) calculation

- The **software integrity level (SWIL)** is a value that helps quantify the risk associated with the software. It is used to determine the software quality requirements listed in section 4.
- A SWIL is a number between **1** and **4**, where **1** indicates the lowest level of risk and **4** the highest (typically safety-critical).
- A SWIL value is calculated by assessing the **criticality of usage** and the **complexity of the software**. The criticality quantifies the impact of the software (for example, will a calibration certificate containing incorrect values be generated?). The complexity quantifies how the development team views the software.
- Further details are available in the [accompanying document](#) to this plan.

# Mathmet: Software Quality Assurance Plan

<b>Criticality of usage (choose)</b>	1	2	3	4
	Not critical			Life critical
<b>Complexity of software (choose)</b>	1	2	3	4
	Very simple			Complex
<b>Recommended SWIL</b>	Is recommended SWIL suitable?			
<b>Factors that justify increasing or decreasing the SWIL</b>				
<b>Reviewed SWIL</b>	Confirm SWIL			

#### 4. Software quality requirements

- This section of the plan lists the quality requirements, determined by either the reviewed SWIL or recommended SWIL (if reviewed SWIL not entered).
- Fill in the relevant sections indicating actions taken to meet requirements. For example, state location where documents held.
- Mathematics must be specified clearly and unambiguously. Further details are available in the [accompanying document](#) to this plan.
- **NOTE:** Mandatory quality requirements are indicated by \*

The diagram consists of a light blue vertical bar on the left and a white area on the right. The light blue bar contains the text "User requirements". The white area contains three text elements: "Documented user requirements" at the top, "Review by team" in the middle, and "Review by customer or proxy" at the bottom. A large, faint, light blue watermark is visible in the background, featuring a stylized 'D' shape and a diagonal line.

<b>User requirements</b>	<b>Documented user requirements</b>
	<b>Review by team</b>
	<b>Review by customer or proxy</b>

# Mathmet Software Quality Assurance Plan

<p><b>Functional requirements</b></p> <p><b>NOTE:</b> In this plan, <b>function</b> does not mean a software module, for example, a MATLAB script, a C function or a database form.</p> <p>It means a task, action, or activity that must be accomplished to achieve a desired outcome.</p>	<p>Documented functional requirements</p> <p>Review by team</p>
<p><b>Design</b></p>	<p>Informal design</p> <p>Review by team</p>

# Mathmet Software Quality Assurance Plan

<b>Coding</b>	<p>Header to identify software name, author, date and version number</p> <p>Program history</p> <p>Review by team</p>
<b>Verification</b>	<p>Module testing as coding progresses</p> <p>Verification of complete software against functional requirements</p> <p>Review by team</p>
<b>Validation</b>	<p>Validation of complete software against user requirements</p> <p>Review by team</p>

# Mathmet Software Quality Assurance Plan

<p><b>Delivery, use and maintenance</b></p> <p><b>NOTE:</b> Maintenance is defined as correct bugs, improve performance, improve other attributes (e.g. user interface layout) or adapt to a changed environment (for example, an operating system upgrade, a new version of MATLAB, MS-Office, Python etc.).</p>	<p><b>Version control on release</b></p> <p><b>Traceability of output</b></p> <p><b>User documentation</b></p>
---	--

## 5. Other information

<b>Platform</b> (for example, operating system)			
<b>Type</b> (choose)		<b>Language(s)</b>	
<b>Responsibilities for testing</b> (give details)	<b>Developer:</b>		<b>Customer:</b>
<b>Backup regime</b> (for example, is software source code stored somewhere that will be automatically backed up?)			
<b>Release rules</b> (i.e., instructions for how to release a new version of the software)			

# Mathmet Software Quality Assurance Plan

## 5. Other information (continued)

<b>Configuration management</b> How will various components of the software be managed? E.g., if MATLAB libraries are required how will user know which versions?	
<b>Maintenance plan</b> E.g. how will bugs and enhancement requested be logged and tracked?	
<b>Mathematical area(s)</b>	<b>If "other" selected enter details</b>
<b>Metrology area(s)</b>	

# Mathmet Software Quality Assurance Plan

## 6. Version history of this quality plan

Version	Date	Author(s)	Change(s)	Approver(s)

# Mathmet Software Quality Assurance Plan

## 7. Disclaimer

The Quality Assurance Tools for data, software and guidelines have been provided by the Members and Partners of the European Metrology Network for Mathematics and Statistics (Mathmet). EURAMET has no influence on its correctness and completeness and does not assume any liability for it.

## 8. Acknowledgements

The European Metrology Network for Mathematics and Statistics is supported by the Joint Network Project 'Support for a European Metrology Network for mathematics and statistics' (18NET05 MATHMET). The project 18NET05 MATHMET has received funding from the EMPIR programme co-financed by the Participating States and from the European Union's Horizon 2020 research and innovation programme.