

See guidance document for details of how to complete this plan.

1. SOFTWARE OVERVIEW

Enter the software name, a brief description etc. Mandatory fields are indicated by *

Customer is not always easy to identify. See the guidance document for further details.

In **Location of Software and Documentation** provide, for example, a hyperlink to a Git repository, a shared folder or a networked drive.

Q1.1 Software name *	
Q1.2 Brief description *	
Q1.3 Developer(s)	
Q1.4 Customer	
Q1.5 Location of software * and documentation	

2. VERSION CONTROL OF PLAN

This section of the software quality assurance plan concerns the plan itself, not the software.

Q2.1 Status of plan	DRAFT ISSUED
Q2.2 Version of plan	
Q2.3 Date of version	
Q2.4 Plan author(s)	

3. SOFTWARE INTEGRITY LEVEL

A risk analysis results in the calculation of a Software Integrity Level (SWIL). The SWIL is a numeric value, between 1 and 4, that determines the quality requirements for the software. 1 indicates the lowest level of risk and 4 the highest (typically safety-critical).

The SWIL is calculated using two other parameters, criticality of usage and complexity of software, which also take values between 1 and 4 (where 1 indicates the lowest level of risk and 4 the highest). The selection of values for these parameters is to some extent subjective. Further details are available in the guidance document (a link is provided at the start of this PDF).

Q3.1 Criticality of usage (choose)	<div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> </div> <div> <div>Not critical</div> <div>Life critical</div> </div>
Q3.2 Complexity of software (choose)	<div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> </div> <div> <div>Very simple</div> <div>Complex</div> </div>
Q3.3 Recommended SWIL	Q3.3a Is recommended SWIL suitable?
Q3.3b Factors that justify increasing or decreasing the recommended SWIL	
Q3.4 Reviewed SWIL	Confirm SWIL

SOFTWARE QUALITY REQUIREMENTS

The following sections up to 11, list the tasks and documentation required for the application scenario quantified by the SWIL.

The plan supports long and formatted text which could be sufficient for storing details of the requirements for **very small** pieces of software.

Mathematics must be specified clearly and unambiguously. Further details are available in the guidance document

NOTE: Mandatory quality requirements are indicated by *

4. User Requirements	Q4.1 Documented user requirements
	Q4.2 Review by team
	Q4.4 Review by customer or proxy

<p>5. Functional Requirements</p> <p>NOTE: In this plan, function does not mean a software module, for example, a 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>Q5.1 Documented functional requirements</p> <p>Q5.3 Review by team</p>
<p>6. Design</p> <p>For much of the software developed with the assistance of this plan, a simple block diagram may be sufficient for both informal and well-structured documented design (see below).</p> <p>For other software, particularly SWIL 3 or 4 software, formal design languages such as the Unified Modelling Language (UML) or the Systems Modelling Language (SysML) could be helpful.</p>	<p>Q6.1 Informal design</p> <p>Q6.3 Review by team</p>

7. Coding	<p>Q7.1 Identify software name, author, date and version number</p> <p>Q7.2 Program history</p> <p>Q7.4 Review by team</p>
8. Verification	<p>Q8.1 Module testing as coding progresses</p> <p>Q8.2 Verification of complete software against functional reqs.</p> <p>Q8.3 Review by team</p>
9. Validation	<p>Q9.1 Validation of complete software against user requirements</p> <p>Q9.2 Review by team</p>

10. Delivery, use and maintenance NOTE: 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 Python etc.).	Q10.1 Version control on release
	Q10.4 Traceability of output
	Q10.5 User documentation

11. PLATFORM ETC.

Q11.1 Platform (for example, operating system)			
Q11.2 Type (choose)	Program	Q11.3 Language(s)	
Q11.4 Responsibilities for testing (give details)	Developer:		Customer:
Q11.5 Backup regime (for example. is software source code stored somewhere that will be automatically backed up?)			
Q11.6 Release rules (i.e., instructions for how to release a new version of the software)			

12. CONFIGURATION AND MAINTENANCE

<p>Q12.1 Configuration management</p> <p>How will various components of the software be managed? E.g., if Python libraries are required how will user know which versions?</p>	
<p>Q12.2 Maintenance plan</p> <p>E.g. how will bugs and enhancement requested be logged and tracked?</p>	
<p>13. Other Information: Mathematical area(s) and Metrology area(s)</p> <p>Q13.1 Mathematical area(s)</p>	<p>If "other" selected enter details</p>
<p>Q13.2 Metrology area(s)</p>	

14. VERSION HISTORY OF THIS QUALITY PLAN

Version	Date	Author(s)	Change(s) from previous version	Approver(s)

15. DISCLAIMER

No representation is made, nor warranty given that this document or the information contained in it will be suitable for any particular purpose. In no event shall EURAMET, the authors or anyone else involved in the creation of the document be liable for any damages whatsoever arising out of the use of the information contained herein. The parties using the guide shall indemnify EURAMET accordingly.

16. 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.

Please also see the acknowledgements in the guidance document that accompanies this PDF form.