



Performance Standards and Test Procedures for Environmental Data Management Software

**Environment Agency
Version 1
June 2008**



Foreword

The Environment Agency established its Monitoring Certification Scheme (MCERTS) to deliver quality environmental measurements. The scheme is based on international standards and provides for the product certification of instruments, the competency certification of personnel and the accreditation of laboratories.

This document provides the performance standards and test procedures for validation of software used for processing monitoring data from:

- atmospheric emissions, etc.
- water treatment and distribution
- waste water treatment
- natural rivers, lakes and estuarine environments
- water storage reservoirs
- boreholes
- trade effluents

Computers are now an integral part of how environmental data is generated, stored, manipulated and reported. Problems with data management can have a number of serious adverse effects, for example:

- affect the quality of measurements as they are made (or input into the data system);
- corrupt data which has been gathered;
- apply inappropriate policy or technical judgements, for example in the context of applying a model, where limits to key model parameters have to be respected;
- remove data from its 'natural' context, that is, to use data without reference to: the chemical analytical methods used, the measurement uncertainty, the detection limit, the definition of what 'zero' means, etc;
- apply secret or obscure manipulations to data;
- make data management overly cumbersome and time consuming;
- make data management unnecessarily expensive;
- otherwise handle or present data in a way, which does not accurately and constructively inform decision-making.

MCERTS for environmental data management software provides a formal scheme for the product certification of data management applications conforming to these standards. We have appointed Sira Certification Service (the Certification Body) to operate this scheme on our behalf.

Product certification under this Scheme comprises:

- audit of the software development lifecycle by an independent software specialist appointed by the Certification Body;
- audit of the application software by the software specialist using the criteria laid down in this standard;
- submission of the audit report to Sira for review;
- issue of the MCERTS certificate.

The certification process and the role of the Certification Body and Certification Committee are explained in Annex A.

If you have any questions regarding the certification process, or would like further information on how to make an application, please contact:

Sira Environmental Ltd
12 Acorn Industrial Park
Crayford Road
Crayford
Dartford, Kent
DA1 4AL

Tel: +44 (0) 1322 520500

email: mcerts@siraenvironmental.com

If you have any general questions about MCERTS, please contact:

Environment Agency
Releases Monitoring
PO Box 519,
Preston
PR5 8GD

Tel: +44 (0)1772 714362

or visit the MCERTS website at www.mcerts.net

Status of this document

This document may be subject to review and amendment following publication. The latest version of this document is available on our website at:

www.mcerts.net

Contents

1.	Introduction	1
1.1	Background	1
1.2	Scope of the standard	1
1.3	Software to which this performance standard applies	2
1.4	The boundary between previous MCERTS certification and this standard	3
1.5	Previous software certification	3
2.	Background to this standard	3
2.1	Auditing principles	3
2.2	Manufacturer's published documentation	4
2.3	Background to part A of the standard – Generic software quality	4
2.4	Background to parts B and C of the standard– Data management applications	5
3.	The MCERTS performance requirements	8
3.1	Part A - generic software quality	8
A1	Software lifecycle definition	8
A2	Application requirements specification	11
A3	Software detailed design	11
A4	Software coding and unit testing	12
A5	Software integration and validation	12
3.2	Part B - data management application standard	13
B1	Introduction	13
B2	Application documentation	14
B3	Mathematical specification and the processing of measurement data	15
B4	Traceability and auditability of data	16
B5	Storage and security of data	16

B6	Interfaces to measurement devices	18
B7	Report generation and the display and presentation of data	18
B8	Confirmation of the certification of new versions and withdrawal of defective versions	19
B9	Software installation and acceptance	20
B10	Commercial status of the supplier	20
3.3	Part C – performance standards for specific applications	21
	Part C1 –Continuous Emissions Monitoring Systems (CEMS) data management applications – Generic Requirements	21
	Part C2 – Continuous Emissions Monitoring Systems (CEMS) data management applications – EN14181 Requirements	24
	Part C3 –Closed pipe flowmeter performance assessment applications using insertion probes (Dynamic Flow Profiling and Verification Applications)	24
	Part C4 – Closed pipe electro-magnetic flowmeter performance assessment applications (Verifier Applications)	25
4.	Testing requirements	25
4.1	Approach	25
	Bibliography	26
	Annex A – Certification Process	27
	Annex B - Definitions	31
	Annex C - Information about the NPL Best Practice Guides	35
	Annex D - Sample Headings for a Software Quality Plan	36

Standards for Environmental Data Management Software

1. Introduction

1.1 Background

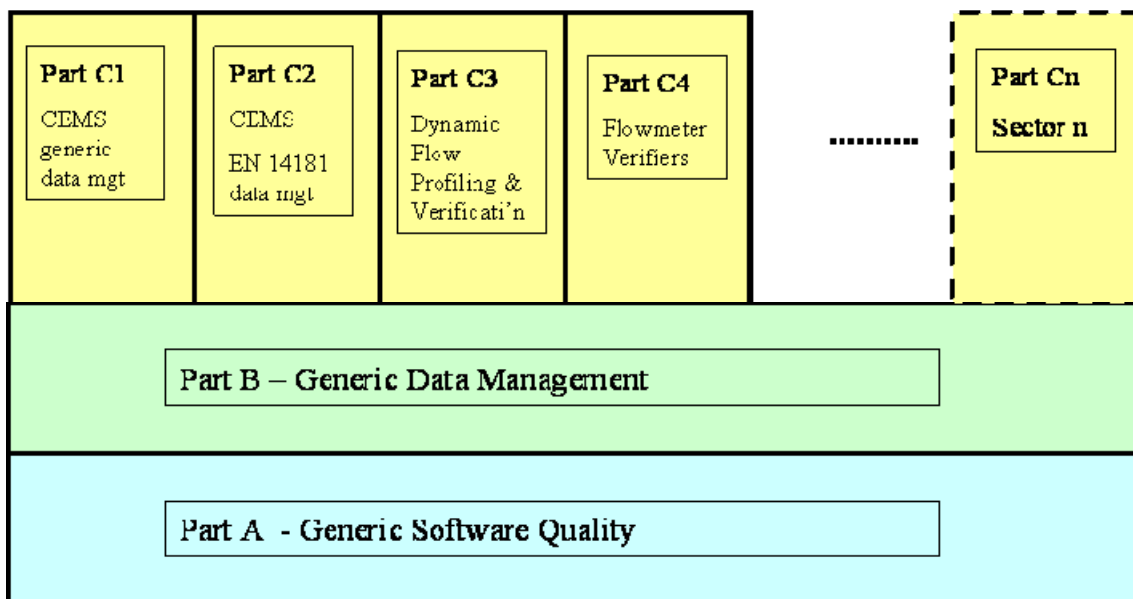
- 1.1.1 This document describes the performance standards, test procedures and general requirements for environmental data management software for compliance with the MCERTS performance standards.
- 1.1.2 The certification process is explained in Annex B.
- 1.1.3 The Environment Agency requires operators of regulated processes to utilise MCERTS certified equipment and, in accordance with this document, environmental data management applications software.

1.2 Scope of the standard

1.2.1 Organisation of the Standard

The standard is in three parts as shown in the diagram below:

- Part A covers the generic quality of the software and defines a standard for the lifecycle used to develop and maintain the data management application.
- Part B covers the performance of the data management application (the “application standard”).
- Part Cx covers the sector specific aspects of the application for sector x, so there are sector standards C1 – Cn. Each sector standard contains requirements that are additional to those in Part B.



The reasons for splitting the standard into three parts are:

- A number of software quality standards exist already and it is possible that some data management applications submitted for MCERTS certification may already have had their software quality assessed, in other words they may already comply with what we identify as Part A of this MCERTS standard.
- There are a number of generic data management requirements that fall naturally into Part B, that is, they are common to most monitoring data management applications.
- Discussions with suppliers of various environmental monitoring applications have shown that there is a diversity of application sectors with specific requirements. Moreover, it can be expected that additional ones will arise as the market develops. Hence we introduce a Part C standard for each significant sector.

A data management application will normally have a certificate covering Parts A, B and a relevant sector standard in Part C. Thus an application has to pass through three ‘quality gates’: the software has to comply with a basic quality standard (Part A); it shall comply with the generic qualities of a data management system (Part B); and it shall comply with the technical performance standard for its specific sector (Part Cx).

1.2.2 Part C Sector standards

So far the following sector standards have been identified:

- C1 – Continuous Emissions Monitoring Systems (CEMS) Data Management applications, generic items;
- C2 - Continuous Emissions Monitoring Systems (CEMS) Data Management applications, items relating to the EN 14181 standard;
- C3 – Dynamic flow profiling and verification applications in the water industry;
- C4 – Flowmeter calibration verifiers (‘Verifier Applications’).

Each Part C standard is presented at the end of this document. Further Part C sector standards will be added as required.

1.2.3 Guidance to the reader

Part A uses a number of software engineering terms that may be unfamiliar to some readers. Where these occur for the first time there are hyperlinks to the definition section of this document. Thus it is more convenient if the reader is reading an electronic version in Word then the ‘Web’ tool bar should be enabled in the ‘Options’ menu under ‘Tools’. This gives the reader the left and right arrows for resuming after following a hyperlink.

Readers who are more interested in the scope of particular environmental applications than in software engineering requirements may find it more convenient to read the relevant sector standard first, then Part B, and finally Part A.

1.3 Software to which this performance standard applies

1.3.1 Software that can undergo the validation process is typically anticipated to be:

- PC-hosted data management applications that perform functions including data monitoring, data storage and archiving, displays and reports;

- Embedded applications hosted on measurement instruments or sensors that perform data management functions;
- Distributed Control Systems - DCS's;
- Supervision, Control, and Data Acquisition - SCADA systems.

1.3.2 It is the responsibility of the user to ensure that the selection and operation of the software is appropriate to the application.

1.3.3 The requirements in this document are intended to be technology transparent to allow the certification of any technical solution that meets the requirements.

1.4 The boundary between previous MCERTS certification and this standard

1.4.1 Some measuring instruments will have been MCERTS certified already, for example against the *MCERTS Performance Standards for Continuous Ambient Air Quality Monitoring Systems*. Such instruments perform a certain amount of data manipulation: averaging, sampling, smoothing, calibration, etc that could be interpreted as falling within the remit of this standard. However, such instruments usually provide data to applications in the 'next layer up' that lie more obviously within the scope of this standard. Thus it would only be necessary for an instrument supplier to submit a data management application running as embedded software in the instrument if at least one of the following conditions applied:

- The instrument generates official reports directly without the participation of any other 'next layer up' application;
- The data manipulation is of a sophistication that raises questions as to whether the software is performing secret manipulations on the data;
- There are doubts about the provenance of the software, for example, its origin or maintainability is uncertain;
- The supplier is uncertain about the present status of the software in terms of its lifecycle, change control and other quality issues.

1.5 Previous software certification

1.5.1 As mentioned above, the results of previous validations/certifications may well be acceptable to the Certification Body, if equivalent to MCERTS and carried out independently. This applies particularly to Part A.

2. Background to this standard

2.1 Auditing principles

In defining the standard the following considerations apply:

- Many developers of data management applications are small companies and do not necessarily have a certified QMS such as ISO 9000:2000; a firm does not need to have an accredited QMS to gain certification for an application using this MCERTS standard.
- In order to gain an MCERTS certificate the software has to be audited by an MCERTS assessor who has the necessary competence. The audit report provides the evidence to the Certifying Body to support the award of the certificate. An audit should not entail independent testing, rather it should seek evidence that testing,

amongst other software validation activities has been carried out in a professional manner.

- Although this standard mentions documents that should be produced as evidence to an MCERTS assessor any audit will focus on the *information* that is provided rather than the organisation of the documents. The developers of the software must provide the necessary information, but how it is arranged into documents is their choice.
- To facilitate audit and to reduce its cost to the developer, applicants who intend to have software assessed against this MCERTS standard will be issued with checklists that can be used by the software developer to help provide the necessary evidence to an MCERTS assessor whose report justifies the issue of an MCERTS certificate.
- This standard can be developed in step with changes in the technology and use of data management applications. Thus, sector-specific additions to this standard may be made in the future in consultation with stakeholders such as industry, the Environment Agency, other interested parties etc. These will be implemented in Part C.

2.2 Manufacturer's published documentation

The developer of the application software shall make available a technical manual and an operating manual to the users of the application. As described in the standard, this documentation shall normally be part of a 'help' system if the host computer has the appropriate human-machine interface (HMI). This information must also be available to the MCERTS assessor involved in the certification process.

2.3 Background to part A of the standard – Generic software quality

2.3.1 Introduction

In the some cases the data management application may form part of a sophisticated measurement instrument control system and the software may have already been certified to standards such as:

- IEC 61508 – 3, when the instrument is used in safety related systems [1]
- The NPL Best Practice Guide No 1 “Validation of Software in Measurement Systems”
- IEC 12207 “Software Lifecycle Processes”.

Certification in this case means that the software has already been assessed by an independent expert whose report provides evidence of compliance with one of the above typical standards, and the report and any resultant certificate is available for scrutiny by an MCERTS assessor, who can form their own judgement of the rigour of the audit by comparing it with Part A of this standard. In general, it is expected that [IEC 61508-3](#) is more than adequate for MCERTS Part A certification while the other typical standards should be sufficient. Thus, to achieve MCERTS certification the data management software must be assessed against:

- all three parts A, B, and Cx of this standard;
- instead of Part A, IEC 61508-3 or another standard of rigour equal or greater than Part A such as the NPL Best Practice Guides [2], [3]; and parts B and Cx of this standard.

In the absence of any pre-existing assessment or audit of the quality of the same version of the software that comprises the data management application it will be necessary to audit the

software against Part A of this standard as well as against Parts B and C.

Many environmental data management applications are hosted on PCs and make use of a variety of software components to handle such functions as graphical displays, real time databases, etc. Often, these applications are written by following a “rapid development” or “agile” software lifecycle model where subsets of the functionality are successively developed and acceptance tested, each during a short period of time. Such approaches are acceptable: Part A of this standard does not require any particular software lifecycle model; rather, it stipulates that whichever lifecycle is chosen must be adhered to and documented so as to ensure that the application is of sufficient software quality throughout its life.

It is sometimes the case that environmental data management applications are not designed and written by professional software engineers but by engineers from other disciplines who have a deep knowledge of the application sector and who translate aspects of their knowledge into programs that grow into quite substantial applications. Part A of this standard is intended to assist such engineers in the task of bringing their applications to a quality where the software can be supported by the author and others over a period of several years so that the users of the programs and the recipients of reports from those programs, including the Environment Agency, can have confidence in the way the raw acquired data has been processed from its original input through to the final report.

As a general point, this standard is intended to encourage the adoption of better and more effective software development tools as they become available.

2.4 Background to parts B and C of the standard– Data management applications

2.4.1 Relevance

The objective, scope and application of any software package must be clear and refer to the environmental regulations to which it applies. The standard will support relevance by identifying a set of criteria that should apply to the data management application. An MCERTS assessor will confirm that the evidence exists to support a ‘fully compliant’ result for each relevant criterion. Those criteria that are not relevant to the application will be marked ‘N/A’.

2.4.2 Responsibilities of the software developer/supplier

It is the responsibility of the supplier of the data management software to fix defects, decertify defective versions, and to certify new ones. Equally, when the application is first produced the developer (or the body commissioning the development) has to obtain an MCERTS certificate, and that certification has to be based on an MCERTS assessor’s report that produces the evidence to support the issue of a certificate. The assessor’s report itself is subject to scrutiny during the certification process to ensure that the Certifying Body is satisfied the MCERTS criteria are met and that the supporting evidence exists. The software supplier must remedy defects in the software lifecycle and in the data management software itself that are found during the MCERTS assessment before a certificate can be issued by the Certifying Body.

2.4.3 Transparency

Because this scheme is for software, which is being used to comply with regulatory

requirements, the methods and data used must be stated and auditable. It is not generally acceptable for data, which has been subjected to secret calculations, assumptions and policies to be submitted in response to Environment Agency requirements. Secret or obscure manipulations to data shall be shared with MCERTS assessors for purposes of assessment only and reviewed under conditions of strict commercial confidence.

2.4.4 Data integrity

This standard has to address the following issues as to whether:

- data is stored and managed without becoming corrupted;
- appropriate calculations are faithfully implemented;
- reports faithfully represent raw or processed data;
- measurement uncertainty is being addressed and propagated to the degree required for the data and decisions which are being supported;
- limitations and qualifications are stated clearly.

If a mathematical specification does not exist already, it will be necessary for the developer to produce one that demonstrates that appropriate algorithms and arithmetic have been used. Likewise, an appraisal of the algorithms with sample calculations shall be available so that it is possible to demonstrate the robustness of the algorithms across the full range of data. Additionally, the appropriateness of manipulations such as smoothing/filtering techniques will be justified.

2.4.5 Completeness

It must be clear how the application fulfils the requirements that:

- data and other contextual information are stored and managed properly within the system;
- data are made available to the operator and used in the preparation of reports;
- where data has the potential to affect decision-making (that is, outlying data points, details of sampling conditions, etc) then this information is used correctly in the generation of interim and final reports;
- in treating outliers the application distinguishes between when it treats them as legitimate data, and when it deems them to be in error;
- the application makes the treatment of outliers clear.

2.4.6 Security

The standard must enforce provisions to protect against loss or corruption of data

2.4.7 Documentation

The standard must ensure that the application guides and enables its operator to use it properly. Conversely, it must discourage the potential for misapplication or improper use.

2.4.8 Embedded data

Data included in the application must be from an approved source such as a reference document. Where such data has to be kept up to date the Environment Agency needs to know whether updated versions have been applied correctly. Equally, embedded data must be correctly accessed and used only for approved and appropriate purposes.

2.4.9 Embedded methods, policies and assumptions

These shall reflect current Environment Agency policies. Methods and assumptions shall be technically valid, reasonable, clear, and have they been agreed with or approved by the Environment Agency.

2.4.10 Hardware and complementary software requirements

Computer architecture, operating systems, application program platforms (for example, MS Excel, Access) and supplementary software elements must be specified. The application developer should include a rationale of the choice of any software components that are incorporated into the application. This choice will be audited from the point of view of its generic fitness for purpose as well as its suitability for MCERTS certification.

2.4.11 Data management applications embedded within instruments

Data management is often an integral part of many modern instrument systems, some of which are already certified under the MCERTS scheme. If these instrument systems were to be submitted for MCERTS certification in their own right against other performance standards, then these previous certifications will be taken into account by the MCERTS assessor when the data management software in the instrument is assessed against this standard.

2.5 Continuity of certification

Upgrading software and/or reference tables should not cause the loss of certification. It is in no one's interest to lock an application into a particular and possibly defective version because it is feared that upgrading to a new version will cause the loss of certification. The policy for this standard enables suppliers to withdraw old and defective versions, while automatically certifying new versions under certain safeguards, including:

- A test specification that is part of the application documentation that was audited against the MCERTS criteria for data management applications.
- Automated installation of new versions.
- Automatic validation, using the audited test specification, of new versions using audited dummy data sets.
- Distinguishing between 'major' changes that may require external assessment by an MCERTS assessor and 'minor' changes that do not.
- Regular surveillance by the Certifying Body of the software supplier's continued adherence to the MCERTS standard.

Any change to an application that impinges on the integrity of reports or displays of regulatory information must be subject to this validation process.

3. The MCERTS performance requirements

3.1 Part A - generic software quality

The following paragraphs state the requirements of the standard in regular font. The software must comply with these requirements. Some requirements are followed by guidance notes in italic font.

A1 Software lifecycle definition

A1.1 The application developer shall justify the choice of the software quality standard used in Part A. This is required to confirm whether the choice of Part A standard is appropriate for the particular application. Each application shall be risk assessed to establish whether other norms that are more rigorous than this one should apply. If the application developer prefers, it is permissible to use another standard of at least equal rigour to Part A of this one.

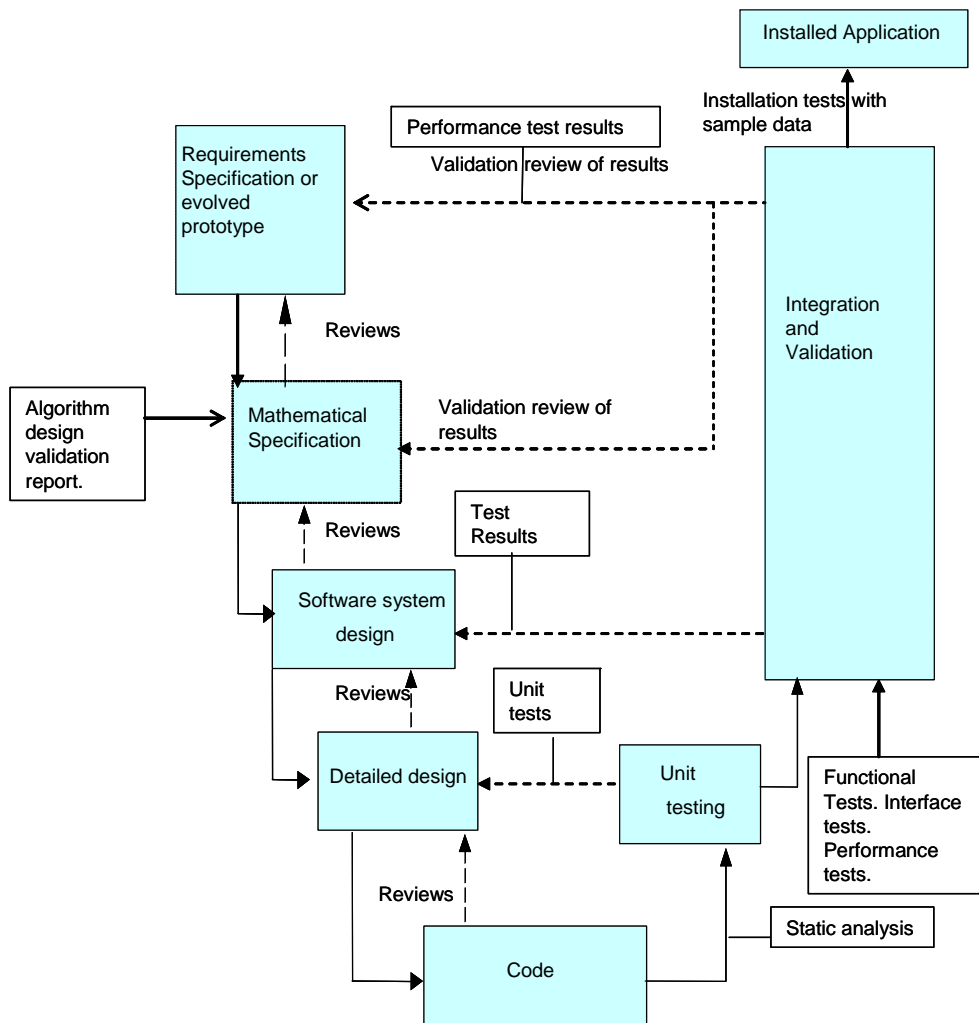
Note: For example instead of Part A below, the [NPL Best Practice Guide No 1 \[2\]](#) could be used. In the case of safety related applications then a more rigorous standard such as [IEC 61508](#) must be used.

A1.2 If the software quality of the application has already been certified then the application developer must supply a copy of the certificate and the existing report to the MCERTS assessor. Evidence referenced in the supplied existing report must also be available to the MCERTS assessor. The evidence must ensure that the previously certified version of the software is the version that is to be used in the MCERTS certification and that no unauthorised changes have been made since that previous certification.

A1.3 The application developer shall define the software lifecycle that applies to the application software. The software lifecycle shall be defined in a [Software Quality Plan](#).

Note: Some guidance is given in Annex D as to the scope and content of the SQP.

The following diagram illustrates the main elements of an acceptable software lifecycle:



A1.4 All documents produced and used in the application software development shall be listed. The list shall include reference numbers and version numbers.

Note: This identifies the documents to be produced and whether they need to be maintained as live documents throughout the lifecycle.

A1.5 The application developer shall apply a change control procedure throughout the application software lifecycle.

Note: A change control procedure will typically be described in a work instruction/company procedure that applies to all projects.

A1.6 All application software, reference data and other objects that constitute a release of the application software shall be held under [software configuration management \(SCM\)](#). An SCM tool shall normally be used to automate this process. Non-use of an SCM tool requires justification by the software developer in the Software Quality Plan.

Note: SCM is essential for the long term integrity and support of software products. All application software, documents, specialised software tools (including in house software tools, tools that are not maintained by others or which are not under a support contract), software deliverables, including test results will be archived under SCM. Tools and packages that are maintained by others under a support contract need not be held under SCM.

A1.7 The version of all [software components](#) and packages used by the application shall be defined and the components and packages maintained (see A1.11).

Note: A [package](#) is a body of software such as SQL Server or a [SCADA](#) package. It is important here to ensure that these packages are maintained and that the suppliers' bug fixes and updates are available so that the data management application makes use of up to date versions of packages and components.

A1.8 The application developer shall maintain a system for reporting, logging and tracking software defects and other alerts.

Note: Defect reports shall be logged and tracked either in a manual system or using one of the many database tools that are available.

A1.9 The software defect reporting system shall enable the application developer to produce accurate release notes with each release of the product. Each release of a new version of the application shall include release notes that include:

- a list of software defects that are cleared by the new version;
- a list of software defects that remain unfixed in the new version;
- a list of new features/facilities included in the new version;
- a statement as to the level of testing performed during the validation of the software.

A1.10 The application developer shall define the means used to verify the software at each stage of its development in the Software Quality Plan.

Note: [Verification](#) will typically be a combination of reviews and testing. The degree of testing shall be identified. Any audit requires evidence that verification has taken place so that the minutes of reviews and the results of tests have to be archived and available for inspection. Reviews can typically be logged in lab notebooks, copied and filed manually. Test results shall be held under [SCM](#).

A1.11 All software tools, [COTS](#) (commercial off the shelf) software components, and hardware used in the software lifecycle shall be listed in the Software Quality Plan. The version number of each software item shall be identified.

Note: This lists the software tools and any hardware used during the software lifecycle. Care should be taken to identify special hardware and specially configured PCs. Special in house utility programs, spreadsheets, VBA plug-ins running within Microsoft Office applications, and other such specialised tools must be identified. These utility programs, etc shall be maintained under SCM, while any platform applications such as MS Office shall be supported and their version numbers identified.

A1.12 The application developer shall backup all software and documentation. The backup procedure shall include:

- procedure for regular on site back ups;
- procedure for regular off site back ups;
- procedure for restoring files;
- procedure for regularly testing the backup arrangements.

Note: Normally, this should refer to a company work instruction or procedure rather than being specially developed for the project.

A1.13 Access to the application software development environment shall be password protected.

A1.14 Responsibilities of all working on the project shall be defined.

Note: This identifies all those involved in the project, how communications such as emails are to be logged,

who is to produce each document and code module and who is responsible for reviewing it. In very small firms there may not be sufficient people to enable independent code and document reviews to be undertaken. If such is the case this should be explained in the [Software Quality Plan](#) and reviews should either be subcontracted to an external reviewer during the development, or the MCERTS audit should be deepened to include reviews. In other words, if the application developer does not have evidence that reviews have taken place then that evidence must be produced as an addition to any MCERTS audit.

A2 Application requirements specification

A2.1 The functions to be performed by the application, the interfaces to be used, and the constraints under which the application must run shall be defined.

Note: In many applications of this type a full requirements specification document is not required. Often, the user interface is evolved through a succession of prototypes and the underlying functionality, interfaces and reports are described in the technical manual for the application. Again, if the application is not safety-related then full traceability of requirements is not required. This MCERTS standard therefore requires evidence that the functions and interfaces have been specified to the extent that validation tests can be specified for them. For example:

- *If a user interface includes menus, then functions are defined for each menu item and exception conditions and user input errors are trapped.*
- *For each communications interface the behaviour of the software is defined for normal and faulty operation.*

A2.2 The functions, interfaces and reports that have been specified by prototyping shall be reviewed by the specifier(s) and revised according to the results of the review. The results of the review shall be archived as part of the documentation.

A2.3 As soon as the application has been delivered by the developer to an outside entity it must come under change control.

Note: This means that release notes do not have to be produced during prototyping. However, [SCM](#) must be used throughout the development, including during prototyping.

A2.4 A technical manual shall be produced for the application. It shall be a live document that describes the application's functions, interfaces, and reports.

Note: The technical manual can form part of any "help" facilities in the application, it does not have to be a paper document. See Part B for further documentation requirements.

A3 Software detailed design

A3.1 The application developer shall make all design information available to the assessor.

Note: This clause of the standard is stated in general terms because the standard does not stipulate any specific design method; rather it demands a design that is appropriate to the application's requirements and the tools and components used to build the application. For example, if the application makes use of a database then its [data dictionary](#) should be available, appropriate measures should be included for ensuring data integrity, etc. Likewise, if a [SCADA](#) package has been used in the application then the [points or tags database](#) containing the measurements used by the SCADA package should be defined and available to the MCERTS assessor. It is for the developer and then the MCERTS assessor to check whether there is sufficient information.

A3.2 The application developer shall maintain sufficient detailed design information to facilitate the long term support of the software and avoid the decay of the software into unmaintainability. The developer shall justify to the assessor the steps taken to ensure that the software can be maintained. This justification shall be included in the detailed design document.

Note: This standard makes no statements about keeping detailed design information in step with the code, but the developer must ensure that the software remains maintainable. For example:

- A software interface to a [communications protocol stack](#) may be quite complex, state-based code that requires finite state diagrams to facilitate its understanding. Therefore these diagrams need to be maintained
- On the other hand, well commented user interface code driven by menus or soft buttons is usually straightforward and an experienced programmer can readily comprehend how the software works from exercising the user interface and studying the code without recourse to additional information.

A4 Software coding and [unit testing](#)

A4.1 A coding standard shall be used for all the language(s) used to develop the software. The coding standard shall define guidelines and conventions for naming, modularity, module complexity limits, defensive programming, testing, and ease of software maintenance.

A4.2 The coding standard and all other guidelines used by the application developer shall be included in any audit against this MCERTS standard.

A4.3 All code shall be reviewed by at least one competent person other than its author.

Note: It is useful to identify those parts of the code that are of greater criticality than others, to concentrate reviews on the more critical code and to perform less rigorous reviews of the non-critical code. Software tools should be used to support the reviewing process. The justification for selective levels of code review shall be made available to the MCERTS assessor, typically in the [Software Quality Plan](#).

A4.4 All code shall be subjected to [static analysis](#).

Note: If static analysis tools are available, then they should be used. In their absence, desk checking for data flow errors such as the use of uninitialised variables, etc should be performed.

A4.5 The principles and methods of [Defensive Programming](#), as defined in Annex C, shall be followed by the application developer.

Note: Code reviews will ascertain that these methods have been followed.

A4.6 The choice of the level of unit (module) testing shall be justified in the [Software Quality Plan](#) by the application developer. Unit testing shall include some or all of the following techniques:

- [structure testing](#) including testing of all branches
- [boundary value testing](#).

Note: Structure testing should at least exercise both sides of every branch although it is not necessary to exercise each side of every defensive programming check. Boundary value testing involves values just inside or just outside the specified limits for each input ('off-by-one values'). It includes special cases such as empty arrays, empty strings, and zero values.

A4.7 [Regression testing](#) shall be used.

Note: Each new version of a code module will have its tests repeated prior to it being used in a new application software build (see A6.3).

A5 Software integration and validation

A5.1 There shall be an Acceptance Test Specification (ATS) against which the application can be validated. [Validation](#) shall include functional and performance tests.

Note: Where automated test tools and harnesses are used the ATS can be specified as a set of scripts or similar means used to drive the tests.

The following clause is based on the NPL BPG No 12 for test and measurement software [3].

A5.2 The ATS shall specify tests including some or all of the following, depending on the

criticality of the application:

Note 1: It is recognised that some application modules are more critical than others, for example a data acquisition module that must execute within tight time limits and be able to handle interface faults and exceptions is more critical than a reporting module that takes input files, processes them and produces a summary for printing and filing. Hence this clause can be interpreted by developers in such a way that they use more testing for critical modules and less for lower criticality ones such as those performing offline reporting. In the latter case it would be expected that the developer will have a set of standard input files and a set of standard output files against which the results from the standard inputs can be compared. For example, these input files will be set up to perform 'realistic' tests, boundary values and unusual inputs.

- “realistic” tests that represent the likely values to be encountered when using the application
- boundary value testing (see above)
- unusual combinations of inputs, including physically unlikely input values
- error handling

Note 2: These include negative values where positive is expected, out-of-range inputs, missing files and bad path names, and tests that cause modules or functions within the application to return an error.

- user interface tests

Note 3: These should include cancelling dialogue boxes, pressing inappropriate buttons, aborting the application when it is running, dragging forms, etc.

- stress tests.

Note 4: These test the application under extreme operating conditions. They should include exposing the software to maximum data rates, writing large disk files, operating the software with high network traffic, etc.

A5.3 Each release of the application shall be validated against the Acceptance Test Specification (ATS) Although it is permissible to use a subset of the ATS in the case of a minor release, that is to say, partial regression testing is allowed when the changes are minor. The justification of the level of testing performed shall be included in the release notes.

A5.4 To ensure effective coverage the ATS shall be reviewed. The review shall provide evidence that the set of tests achieves an appropriate level of validation.

A5.5 The application-specific testing required to fulfil the detailed requirements defined in the relevant Part C sector standard shall comply with clauses A5.1 – A5.4 above.

3.2 Part B - data management application standard

B1 Introduction

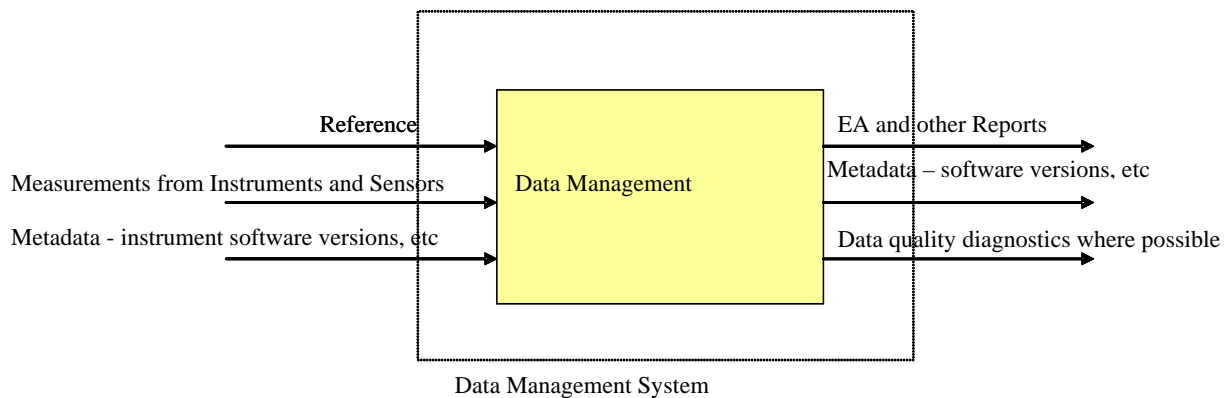
While Part A is concerned with the inherent quality of the software that makes the data management application, Part B covers the general aspects managing the environmental measurement data. Even if Part A shows that the software is well designed and built to an appropriate standard, the application can still fail to be certified if the application facilities cannot provide suitable means of ensuring the integrity of the measurement data and the metadata that is used in the processing and display of the measurements.

The main elements covered by Part B are:

- Application documentation

- Mathematical specification and the processing of measurement data
- Traceability and auditability of data
- The storage and security of data
- Interfaces to measurement devices
- Report generation and the display and presentation of data
- Certification of new versions
- Software installation and acceptance
- Commercial status of the supplier.

The scope of Part B can be summarised as follows:



It is important to note that as well as the measurement data management software, reference data and the data that describes the origin of the sources of the data ([metadata](#)) must all be included in the scope of the system, and are all subject to certification. [Metadata](#) such as software version numbers are very significant because the Environment Agency must be sure about which versions are certified and those versions, which are not. There is a strong connection between the general requirements for software configuration management described in Part A and the requirements in Part B for displaying the correct provenance of data such as chemical and physical constants, calibration curves, etc.

The following paragraphs state the requirements of the standard in regular font. Some requirements are followed by an explanation in italics.

B2 Application documentation

B2.1 Every installation must be accompanied by release notes as described in A1.9 above.

Note: this applies not only to changes in functionality but also to changes to any constant data that are 'plumbed into the software', for example, tables and conversion factors.

B2.2 An operations manual/help system shall be available. This must include guidance to novice users except in the case where the application is only to be used by previously trained staff.

Note: As mentioned in Part A, the operations and technical manuals do not need to be paper documents.

B2.3 There shall be a technical manual/help system which must include the details of the input measurements, data processing algorithms, data storage and databases, outputs to external interfaces, reports, and other functions and interfaces of the application.

B2.4 Requirements for any training in the use of the application shall be stated in the operating manual.

B3 Mathematical specification and the processing of measurement data

B3.1 The application developer shall produce a Mathematical Specification for the application even when another standard is used instead of Part A of this MCERTS standard.

Note: The [NPL Good Practice Guide 1](#) [2] insists that this document be produced and this standard follows suit for the same reason: all steps in the processing of measurement data have to be carefully specified and their associated sources of error identified.

B3.2 The Mathematical Specification shall include:

- Specification of numerical inputs and their ranges, their validation, and the handling of outliers.
- Definition of all formulae and specification of the algorithms used in the application, including their limits of validity and evidence of their suitability in numerical analytic terms.
- Use of stable formulae whose solution is not vulnerable to rounding errors.
- Validation of any non-trivial algorithms by means of an offline implementation using representative test data.

Note: For example, the mathematical specification must demonstrate:

- *The appropriate use of any floating point arithmetic and the avoidance of bad practices such as cancellation, division by very small quantities, exact comparison of floating point numbers, etc.*
- *The use of well conditioned formulae where small changes in the input data do not cause relatively large changes in the solution.*
- *Appropriate use of any scaled integer arithmetic.*

The mathematical specification typically forms part of a requirements specification.

During application software validation the results of the above off-line validation of non-trivial algorithms will be used to corroborate the results from the on-line version. It is very important that any test data sets and test software are as carefully designed and controlled as the main application.

B3.3 The Mathematical Specification shall specify how instrument status information is to be used (when it is available).

Note: If any of the following data are available then the application shall make use of it:

- *Measurement uncertainty and/or device statuses returned by measurement devices.*
- *Indications of the validity of measurements based on the status of the measurement devices and measurement uncertainties computed within the application.*

The principle is that, where possible, useful status information should not be ignored but used to indicate the validity of measurements.

B3.4 All input data shall be verified before being used in calculations, reports, displays or other application outputs. Unverified data must never be used in any output from the application.

Note: If the application receives data from another system that has already verified the data and marked it as such then there is no need to repeat the verification.

B3.5 Each measured value shall be accompanied by an indication of its validity, if the measured value is invalid then the reason for the invalidity shall be recorded. All conditions that can render one or more measured values invalid shall be recorded with the measured

values. The method of recording shall permit the affected data to be identified, together with the reason for their invalidity.

Note: The ability to recover this information from historical records is an important part of establishing an audit trail for performance of the application. The data processing shall use this information to determine which measured values are eligible for inclusion in the calculated values and reports.

B3.6 The application shall be able to ignore input data that is acquired during times and conditions when measurements are not required to be reported.

Note: This typically applies when a plant is running but not processing, for example, an incinerator may be required not to report measurements when nothing is being burned in order to prevent average measurements being distorted by low readings during the time when no waste is being burned.

B4 Traceability and auditability of data

B4.1 Measurement data records shall be traceable to the physical location and time of the measurement, the measurement device(s) used, and the status(es) of the device(s).

B4.2 The time and date of measurement shall use [UTC](#) and shall normally make use of an independent time and date source. If such a source is not used then its non-adoption shall be justified in the application design.

Note: Non-adoption of an independent time and date source would be justifiable in a simple stand-alone application that is running on one PC and which is responsible for the time-stamping of all its input data and where the application timing requirements allow a margin of error on the time of day that is used. Networked systems or applications using data from several different input devices that time-stamp their data shall use an independent time and date source.

B4.3 The status history of measurement devices shall be held so that measurement data can always be traced to the operational status of the device (if available) at the time of the measurement.

Note: If the calibration of the device is traceable to a standard then a reference to that information, or a similar indication, should be included in the operational status.

B4.4 All measurement data records shall be auditable so that the original input data and all subsequent modifications can be identified.

B4.5 All manual changes to data shall be auditable and traceable to the date, time, and operator responsible for the change.

Note: For example, a measurement database must be protected from attempts to export data into a spreadsheet application, to change the data, and then to import the changed data back into the measurement database. If such an operation is performed on database records then this fact must be logged and must appear in any audit trail pertaining to the custody of the measurement data. The principle that applies here is 'no secret manipulation of data'.

B4.6 Configurable applications shall include an audit trail showing changes to the configuration.

Note: Typically, applications that are set up with a configuration tool that adds, removes, and edits the definition of data inputs shall have those changes recorded so that the configuration at a particular time can be made available readily.

B5 Storage and security of data

B5.1 Access to the application and all its data shall at least be password protected. Where access to data has to be segregated between different user roles then separate protected passwords shall be used.

B5.2 All attempts at unauthorised access to the application shall be logged and the fact that such failed accesses have occurred shall be made evident to an appropriate application user or administrator.

B5.3 Data shall be encoded and stored in a form such that it is not susceptible to manual modification.

Note: Typically this means that human readable data representations should be avoided.

B5.4 All accesses to the data shall be checked against corruption or loss of data. In cases where these checks are infeasible the developer shall justify their exclusion to the assessor.

Note: Here, it is required that databases and other means used to store data shall apply data integrity checks as and when each record is accessed. This is to ensure that all data retrievals are accurate; [CRCs](#) (see definitions in Annex C) or other means of detecting corruption and inconsistency shall be used on data records.

B5.5 Databases and other repositories of data shall be checked by the application software for the loss or corruption of data at regular intervals.

B5.6 The application shall include functions that facilitate the auditing of its data and the tracing of data in terms of its origin and path, modification, and other manipulations.

Note: Audit checks are required in B4 above. These facilities shall assist auditability and traceability so that a user of the software can view a report and establish that data that contributes to the report has been through processes such as smoothing, averaging, manual editing, etc.

B5.7 The origin and path of the data shall include the physical location of the raw measurement or the point of sampling, the sample stream (if relevant), the instrument or sensor, the input channel and network node (if relevant). See also B4.1.

B5.8 The application and its data shall be backed up so that it is capable of being restored without undetected loss of data.

Note: If hardware failures or operating system failures cause stored data to be lost or measurements not to be made then the gaps in the record shall be advised to users of the data in any reports generated from the incomplete data.

B5.9 Calibration data, constants, operating parameters and all other supporting data shall be auditable so that the history of all modifications is recorded. The supporting data shall include the value, units, period of validity, and origin. If the end user chooses, history files can be archived off-line after a period of not less than three months.

Note: Where calculations make use of these supporting data it shall be possible to show whether, for example, an instrument has been recalibrated during a reporting period. This clause applies to data entered manually, data uploaded from instruments, or from other sources such as reference tables.

B5.10 An audit trail shall be established for each set of supporting data, with the time and date of each change, the name/role of the modifier.

Note: A calibration curve is an example of a set of supporting data. The data and the assumptions used for calibration curves shall be clearly traceable.

B5.11 The application shall support the display and reporting of audit trail information on demand. The audit trail should be able to go back at least six years.

B5.12 The application shall be designed to minimise the manual transcription of data and

other avoidable sources of error

B5.13 If the application runs on a PC on which other applications can be loaded, or is on a network that requires protective security measures then the operating system used for the PC must conform to an appropriate security level such as the Common Criteria Evaluation Assurance Level.

Note: See the definition of '[Common Criteria](#)' in Annex C for further explanation.

B6 Interfaces to measurement devices

B6.1 All failures leading to loss of data from each measurement channel shall be logged. The reason for the failure shall be logged if it is available.

Note: This is so that the validity/invalidity of time series of measurements and any resulting averages can be marked with an appropriate validity/invalidity code. Thus failures in network interfaces, instrument failures, etc shall be logged and made known to users.

B6.2 Where a group of measurements has to be synchronised within a time window any loss of synchronisation shall be logged and the corresponding measurements marked to that effect.

B6.3 Time based measurements shall normally be synchronised with the calendar/clock.

Note: This means that normally a measurement made every 15 minutes is made at HH.00, HH.15, HH.30, etc. However, in some applications measurements it is necessary to postpone the times of day when they are sampled in order to accommodate higher priority processing that is scheduled at fixed times, for example, end of day processing at midnight on some process control systems. Deferring any time based measurements by a fixed number of time units must be justified and the justification documented in the Technical Manual.

B6.4 Time and date formats shall avoid representations that lead to roll over or other misleading behaviour at any time or date within a 100 year period beginning at the earliest in 1980.

Note: The Technical Manual shall state the limits on the application's time and date handling.

B7 Report generation and the display and presentation of data

B7.1 All reports and displays shall indicate whether any measurement data or calculated results are based on invalid or incomplete data.

Note: Incomplete data would be for example an average measured value taken over fewer points than normal because an interface has been temporarily out of use.

B7.2 If Environment Agency policies apply to the calculation and presentation of data in displays and reports then the displays and reports shall comply with the relevant policies. References to the appropriate policies shall be available to the user.

Note: Typically, these references should be listed in the applications 'help' and in the Technical Manual.

B7.3 Statutory reports and displays shall be labelled with their official title.

B7.4 Only reports generated by MCERTS certified applications shall be marked as 'MCERTS compliant' reports, the MCERTS logo shall be used and it shall contain the number of the certificate issued by SIRA to the application software supplier. Reports generated from exported data on other applications shall be marked as 'Advisory'.

Note: MCERTS compliant reports will normally be identified by their official title in the header and be marked with the MCERTS logo. The inclusion of the supplier's certificate number in the logo enables a report to be traced to a particular software product. It is also intended to discourage any misuse of the MCERTS logo.

B7.5 Export of reports and displays shall not cause validity/invalidity information to be lost.

B7.6 Where test data is used to exercise the application any displays or reports containing such test data must be unambiguously marked to indicate that test data is being used.

B8 Confirmation of the certification of new versions and withdrawal of defective versions

B8.1 The application software supplier shall be responsible for the continued certification of new versions of the application under the condition that the software remains compliant with this standard.

Note: Once the MCERTS certificate has been issued the application supplier must ensure that the software is appropriately maintained using the audited software lifecycle.

B8.2 Each release of the application shall have version number of the form ‘major.minor’.

Note: A major release requires recertification, a minor release does not. Where developers deem a change to be ‘minor’ they must justify the decision, for example, by listing the affected modules and showing that the change is very localised. Typically, this can be done automatically by means of a code navigation tool that shows module dependency by cross-referencing. A minor release has the following characteristics:

- *A localised change within the code to fix a software defect, to reformat a display or report, or to add a feature, or*
- *Small modification for a site-specific feature that has to be introduced to meet the end-user’s requirements, or*
- *A change to include a new edition of reference tables, or*
- *A rebuild to take advantage of a new release of a function library*

A major release entails, for example:

- *Addition of significant functionality, for example, a new class of input devices, or*
- *A port to a different operating system or database package*

B8.3 In the event of a major release the developer shall provide brief supporting evidence to the Certification Body who will advise whether recertification is necessary.

B8.4 Each release shall be accompanied by release notes (see A1.9) which must include a justification as to why the release is major or minor. The justification shall be supported by reference to the number of source files changed/added and the number of lines of code that are changed or added.

Note: This information should normally be made available automatically from the software configuration management tool or from an automatic file comparison tool.

B8.5 An MCERTS certificate is valid for five years. When the period is close to expiry or when there has been a succession of ‘minor’ changes the application developer shall apply for an extension to the certificate by submitting the release notes (the change history) for the minor changes to the certifying body.

Note: The set of release notes will indicate whether the succession of ‘minor’ changes amounts to a significant change that demands recertification. A surveillance check of the software development process used to support the software will be required two years after the first issue of a certificate.

B8.6 Where an application is sold as a member of a family of applications and each member is in fact a subset of a master application then it is permissible to certify the master application and then state that the subset is also certified provided that the entire master application is audited and the subset is a strict subset of the master application code.

Note: Typically, the subset applications will have a common user interface where various on-screen options are greyed out in some subsets.

B8.7 A developer shall only withdraw a version of an application when there is evidence that continuing use of that version will compromise the quality of the data supplied by the application. This evidence shall be made available to all purchasers of the application. It is permissible to have different versions of an application in use at a time provided that each is certified.

B9 Software installation and acceptance

B9.1 Installation of the application software and its associated files shall be automated.

Note: Applications hosted on PCs are expected to be installed by end users; hence this standard requires that installation errors should be avoided by reducing the opportunity for user errors.

B9.2 De-installation and/or quarantining of previous versions of the application software and its associated files shall be automated.

B9.3 Each release of an application shall be accompanied by Release Notes (see A1.9 above) and be uniquely identified by a version number of the form 'major.minor' (in accordance with the software configuration management requirements and B8 above). The application shall display its version number whenever requested (see B9.5).

B9.4 Each release of the application shall be capable of being exercised by the user so that the installation can be shown to be correct. If the End User requires, there shall be a demonstration that the application algorithms are being applied correctly to audited sets of test data.

Note: Once the installation has been completed it shall be possible for the user to run a number of test cases automatically through the application in order to show that the installation has been successful. For high utilisation sites where downtime must be kept to an absolute minimum it may well be the case that a demonstration is not feasible, in such cases strong reliance shall be placed on the evidence supplied in the release notes that the installation has been tested properly.

B9.5 The release notes must be available on line and must include:

- list of all cleared software defects
- list of all outstanding software defects
- list of all fixed data files and the provenance of their data, this also applies to constants that are 'plumbed into the software', for example, physical constants and conversion factors, which must be displayed whenever requested.

Note: To facilitate software acceptance and support the version numbers of the application, its major components, and any reference data should be available to the end user (for example, via the "Help" – "About" menu).

B10 Commercial status of the supplier

B10.1 The application software supplier shall provide evidence that they can provide technical support for the application.

B10.2 Application suppliers shall place the application source code and its design information on a web file storage service so that it is available to others in the event of the supplier becoming unable to maintain the application. Evidence for this arrangement and instructions for retrieving the source and design information shall be available as part of the

certification information held by SIRA Certification Services. In the event of the supplier becoming unable to support the software bona fide customers of the software shall be given access to the source and design information by SIRA on request.

Note 1: Web file back up and storage services are available from Google, BT Digital Vault, and other providers. Some of the more limited services are free of charge to account holders. Typically the supplier will put each new valid release as a set of zipped files into a separate directory on the backup service. The supplier will delete releases from the back up service when they are withdrawn. These files can be password protected and the password must be included in the information held by SIRA.

Note 2: This clause applies with particular force to:

- *One- or two-man software development teams vulnerable to the loss of key staff.*
- *Small firms or small subsidiaries of large companies.*

3.3 Part C – performance standards for specific applications

Part C1 –Continuous Emissions Monitoring Systems (CEMS) data management applications – Generic Requirements

Objective

This annex specifies the MCERTS performance standards for Data Acquisition and Handling Systems (DAHS) used in acquiring data from Continuous Emissions Monitoring Systems (CEMS).

The scope of these performance standards includes application software designed to run on Personal Computers (PCs), smart instruments, and other processors, together with any associated data capture hardware.

Legal drivers

The MCERTS performance standards described here are designed to meet the requirements of EC Directives and the mandatory standards cited within these Directives. The Directives include:

- *Directive on the limitation of emissions of certain pollutants into the air from large combustion plants (2001/80/EC).*
- *Directive on the incineration of waste (2000/76/EC).*
- *Directive 2008/1/EC concerning integrated pollution prevention and control.*

Scope of the MCERTS scheme

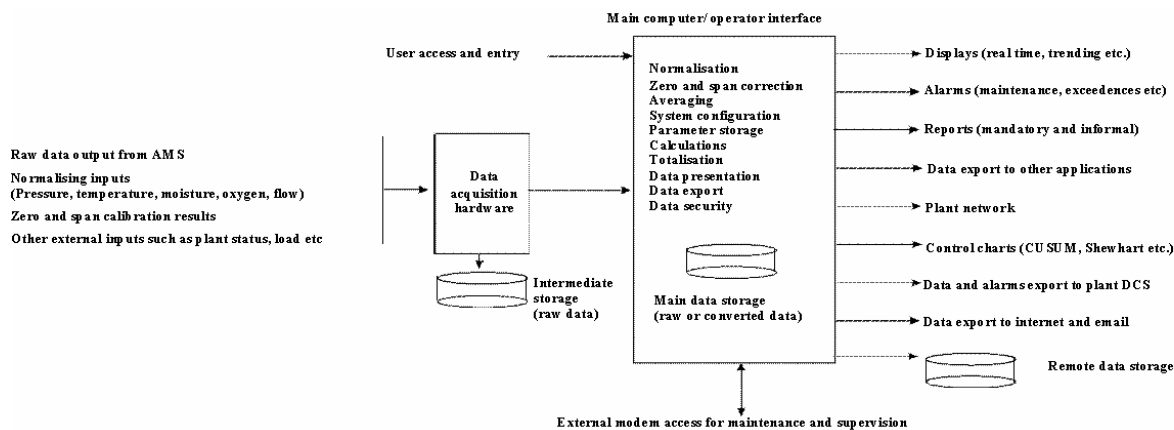
The scope of processes within the MCERTS scheme for DAHS for CEMS is as follows:

- **Incineration processes**, including those for hazardous waste, co-incineration, municipal waste and clinical waste. In short, those processes covered by the *Directive on the incineration of waste (2000/76/EC)*.
- **Combustion processes** covered by the *Directive on the limitation of emissions of certain pollutants into the air from large combustion plants (2001/80/EC)*. This also includes gas turbine monitoring applications which are also covered by this directive.
- **Solvent-using processes** covered by the *Solvent emissions Directive (1999/13/EC)*.

- **Other industrial processes** covered by the IPPC Directive (2008/1/EC).

A typical application scenario

The following diagram indicates the scope of an application that would typically fall within the remit of this MCERTS sector standard C1, and also C2.



General requirements for CEMS data acquisition and handling systems (DAHs)

The generic standards for DAHs for CEMs are covered in Parts A and B of this document. Where manufacturers utilise hardware to convert the output from a CEM into a more useable form, and do not produce all of these components themselves, they should supply components from other manufacturers that they consider suitable for the required purpose(s). In these cases:

C1.1 All hardware shall conform to all applicable EC Directives, including:

- Electro-Magnetic Compatibility Directive 89/336/EEC and its amendments 92/31/EEC and 93/68/EEC.
- Directive 72/23/EEC (known as the Low-Voltage Directive and its amendment 93/68/EEC) covering electrical equipment designed for use within certain voltage limits.
- Software producers or suppliers shall supply independently verifiable and traceable evidence of conformity to all the relevant Directives applicable to the equipment.
- The nominal temperature range for data acquisition hardware is $-20\text{ }^{\circ}\text{C}$ to $+50\text{ }^{\circ}\text{C}$, unless assemblies are installed within temperature-controlled environments, in which case the required range is $+5\text{ }^{\circ}\text{C}$ to $+40\text{ }^{\circ}\text{C}$.

General Acquisition and Reporting Requirements

C1.2 The DAHS shall be able to acquire, process, and report data in the formats required for the measurands specified in Waste Incineration Directive (WID), Large Combustion Plant Directive (LCPD) and Environmental Permitting Regulations, 2007 (EPR-PPC). The DAHS

shall be capable of acquiring data within over the entire ranges used in the CEMs..

C1.3 Data shall be stored in a suitable database as an averaged value and from this value only additive averages are permitted to be created.

Note: For example, from a summation of 10 minute averages, a 20, 30, or 40. minute average can be created, but not, for example, a 15 minute average, nor may an average over a period shorter than that of the stored value be created.

C1.4 The software should be capable of generating all the averages required in the Directives listed above in the paragraph 'Legal drivers'.

C1.5 The communication subsystem used by the data management application shall be capable of acquiring data from the AMS it serves within a 10 second cycle.

C1.6 Raw data from the AMS shall be converted from ppm etc to normalised mg/m³ using the normalising calculations that are given in section 3.4.3 of EA Technical Guidance note M2 and using recognised molecular weights.

C1.7 The system shall be capable of generating all relevant reports stated in the Directives listed above in the paragraph 'Legal drivers'.

C1.8 There must be procedures in place to prevent unauthorised adjustment of the recording device.

C1.9 The software shall be able identify and report the following events and conditions:

- a) Invalid averages
- b) Plant status (not in operation, start-up, normal operations, shut-down)
- c) Maintenance alarms
- d) Calibration events (zero and span)
- e) Attempts to adjust stored emissions data
- f) CEM operating status
- g) CEM malfunctions.

C1.10 The data acquisition and handling system shall be available for 99% of the time.

C1.11 The DAHS shall have the capability to store uncertainty requirements and emission limit values given in the Directives above. It must then be able to apply them and correct monitoring data by the measured uncertainty error.

C1.12 Hardware requirements

C1.12.1 The DAHS shall have hardware equipped with suitable input ports for accepting signals from the CEM, sensors, and field transmitters. These signals may be in the form of analogue currents or voltages, digital information or status signals.

C1.12.2 The data acquisition system shall have an output port for connecting to an external printer.

C1.12.3 The data acquisition hardware shall be secured against unauthorised access (see B5.1 and B5.2).

C1.12.4 The data acquisition hardware shall normally be equipped with an accurate timing system which is continuously updated by an external radio device which is in direct contact with MSF-60 (Rugby atomic clock) or a device receiving time from the Global Positioning System (see also B4.2)

C1.12.5 All computer based hardware shall be equipped with an Uninterruptible Power Supply which can sense when mains power has failed; signal this event to the controlling program, and give sufficient time for the system to close down all operating software in a controlled fashion and so prevent unnecessary data loss.

Part C2 – Continuous Emissions Monitoring Systems (CEMS) data management applications – EN14181 Requirements

The data management application shall comply with the specific requirements of EN14181, namely:

C2.1 During QAL 3 checks the application must be able to:

- 1) Acquire the output from the CEM when zero and span checks are performed under QAL 3.
- 2) Record both positive and negative values for drift.
- 3) Record any changes in readings from previous zero and span checks.
- 4) Record zero and span data results for at least one calendar year.
- 5) Have the means to generate control charts using the data gathered during zero and span checks. Control charts may include, but are not limited to; CUSUM (drift and precision), Shewhart and Exponentially Weighted Moving Averages (EWMA).
- 6) Maintain a summary of QAL 3 baseline resets.
- 7) Perform zero and span drift tabulation.

C2.2 The application shall only use calculations as specified in Annex C, EN14181, when applying the CUSUM procedure.

C2.3 The application shall apply the calibration function generated by QAL 2 calculations (section 6.4.2.).

Part C3 –Closed pipe flowmeter performance assessment applications using insertion probes (Dynamic Flow Profiling and Verification Applications)

This part of the standard is under development.

Part C4 – Closed pipe electro-magnetic flowmeter performance assessment applications (Verifier Applications)

This part of the standard is under development.

4. Testing requirements

4.1 Approach

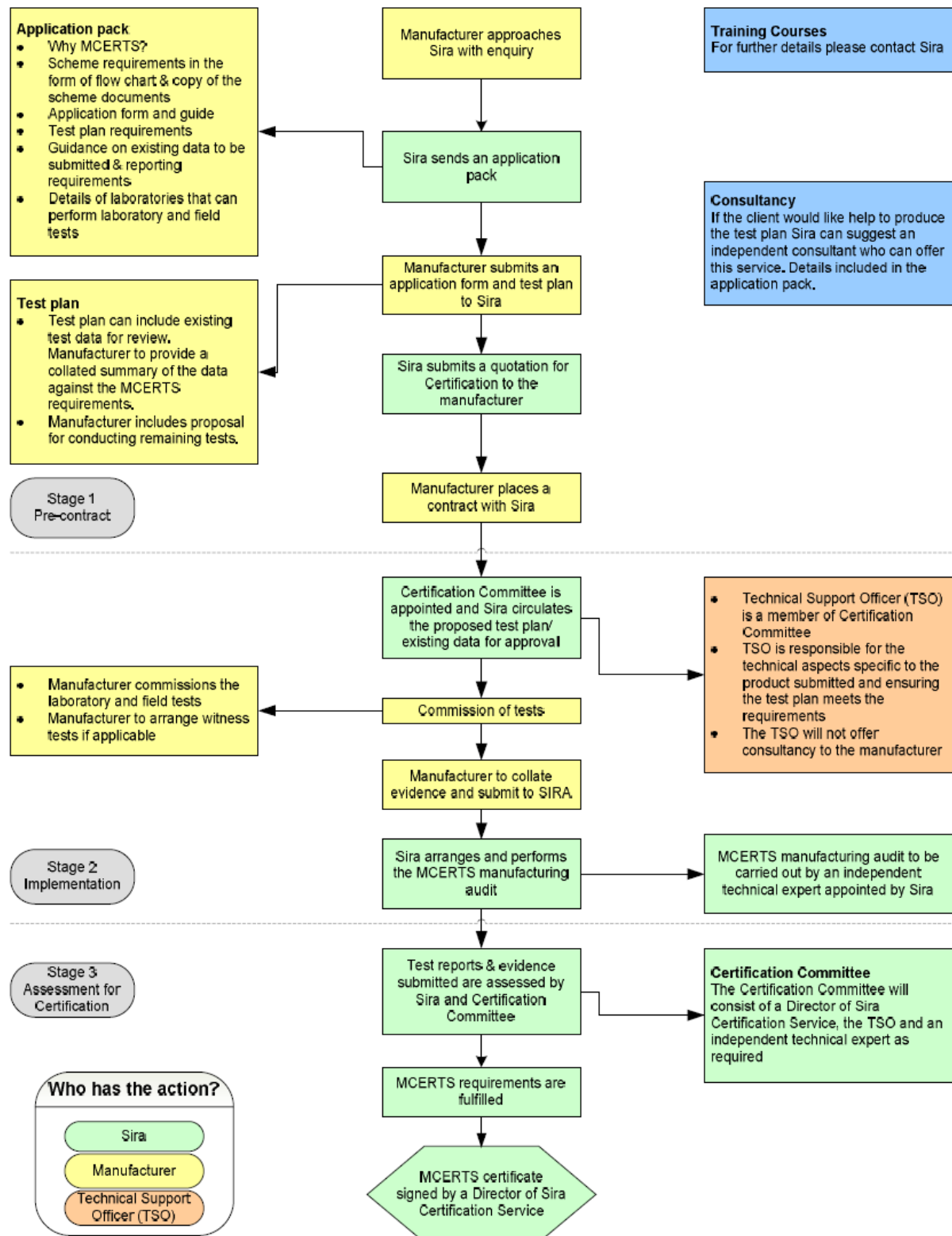
As described above in A5, A6, B8, B9, testing is the responsibility of the application developer who must provide evidence of all the levels of testing that have been performed on the application software. This evidence is archived and kept under software configuration management throughout the life of the software.

Bibliography

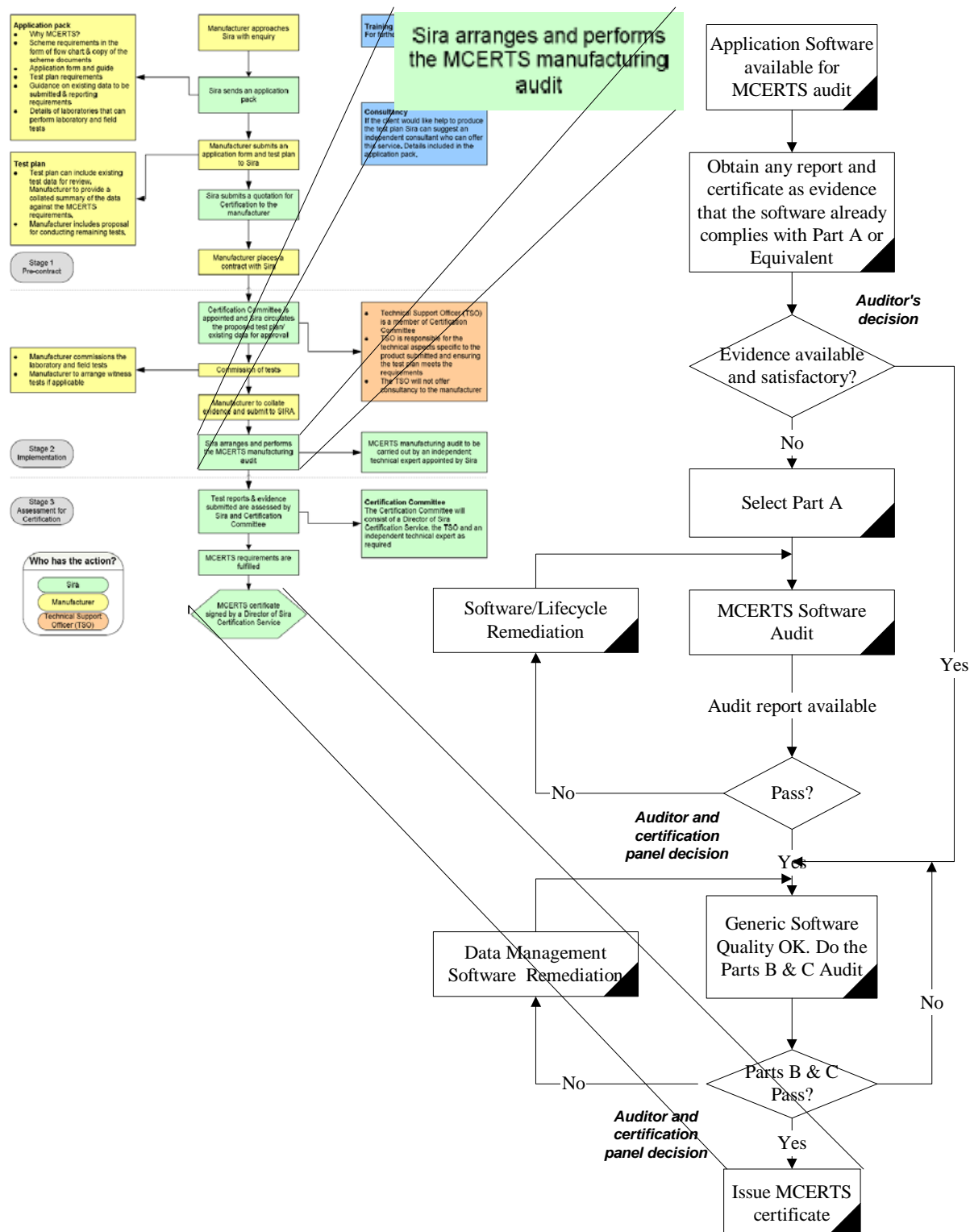
- [1] European Standard 2001, *Functional Safety of electrical/electronic/programmable electronic safety-related systems*, IEC 61508, 1998 + corrections 1999
- [2] Software Support for Metrology Best Practice Guide No 1, *Measurement System Validation: Validation of Measurement Software*, Version 2.1 Draft 4, Brian Wichmann, NPL, July 2003.
- [3] Software Support for Metrology Best Practice Guide No 12, *Test and Measurement Software*, NPL, October 2002.
- [4] Report to the National Measurement System Directorate, DTI, *Techniques for Validation of Measurement Software – without specialised tools*, Roger Barker and Graeme Parkin, March 2004
- [5] *Code Complete*, Steve McConnell, Microsoft

Annex A – Certification Process

1 General MCERTS Certification process



2 MCERTS certification process details for data management applications



The process for standards selection and the subsequent application audit is summarised in the above diagram.

Product certification comprises the following phases:

- part A audit, which can be replaced by an equivalent
- part B and C audit
- review of the audit report
- issue of the MCERTS certificate.

The audit can cover all three parts of the standard in the same process.

Manufacturers seeking certification should contact the Certification Body who will advise on any specific requirements for the software under consideration.

The standard states that where an application can be configured into subsets of the functionality certification must cover the entire application and thereby include all options.

Instruments that include a data management application must still have their instrument functions certificated to the appropriate MCERTS standard.

3 Certification Committee

The role of the certification body is to assess and certify compliance with the MCERTS standard for defined applications and or conditions.

In performing this role the MCERTS scheme requires the certification body to consider the relevance of the procedures defined in the MCERTS standard to the specific product to be certified. The technology or defined application of a specific product may make certain of the documented tests inappropriate. The certification body is required by the MCERTS scheme to exercise its technical judgement when considering these matters.

Any certification decision based on technical judgement of the standard shall be taken by an appropriately independent, competent person or group of persons, who in this MCERTS standard are referred to as the **“Certification Committee”**.

When the certification body exercises its technical judgement the rationale supporting any such decision shall be appropriately documented.

Any certificate issued by the certification body shall identify any variations from the normative MCERTS standard.

On request the certification body shall provide the MCERTS scheme owner with the rationale for any decision based on technical judgement, within the relevant confidentiality constraints.

4 Certification Scope

This is established in the body of the standard. Basically, minor revisions are permitted for a certified application provided that the criteria laid down in the standard are followed for all revisions.

5 Testing

The application software shall be tested (at a number of levels) by its developers. This

standard insists that evidence of that testing be archived by the developers and provided to the MCERTS assessor.

6 Auditing and surveillance

An audit of the software development process shall be conducted by the Certification Body to confirm that the application developer has followed this standard throughout the lifecycle of the application software.

Subsequent surveillance audits of the software development process are normally conducted every two years to ensure that the application is being maintained appropriately. Once this has been established the Certification Body may extend the interval between audits or require submission of specific audit data for review off site.

7 Certificate validity

MCERTS certificates for the data management application software are valid for five years. After this time, the certification is reviewed and any necessary auditing will be identified to maintain the certification. Assessment for this five-yearly recertification shall be carried out against the MCERTS standards current at the time of recertification (see clause B8.5).

8 Modifications to certified application software

Modifications to certified application software are allowable so long as the developers can demonstrate to the Certification Body that these software changes have the characteristics:

- They are minor changes, as defined in the standard.
- The certified application software development process is adhered to.
- The cumulative effect of a sequence of minor software changes does not amount to a major change from the originally certified software.

Manufacturers must keep detailed records of all design changes to the application software as stated in the standard, and have provisions for design verification and review to ensure that the software still meets the required performance standards. These records shall be available for independent review and verification so that the Certification Body can decide whether the scope of any changes is either major or minor.

Annex B - Definitions

AMS

Automated Measuring System

Boundary Value Testing

A testing technique using input values at, just below, and just above, the defined limits of an input range; and with input values causing outputs to be at, just below, and just above, the defined limits of an output range.

Branch Testing

Testing technique to satisfy coverage criteria which require that for each decision point, every possible branch (outcome) shall be executed at least once.

Common Criteria

Ratified as an international standard in 1999, the Common Criteria (CC) replaced several older evaluation schemes including the U.S. Trusted Computer Systems Evaluation Criteria (TCSEC), and the European Information Technology Security Evaluation Criteria (ITSEC). The CC are defined and maintained by an international body composed of nations that recognize CC evaluations and are recognized by the International Standards Organization (ISO) as ISO Standard 15408. CC certification does not ensure that a product is free of security vulnerabilities, it provides a degree level of security assurance by ensuring that the product performs as documented and that the vendor supports the product by fixing flaws when they are discovered. For example, Vista, Windows XP Professional and XP Embedded Service Pack 2 both comply with the Common Criteria, but other versions of Windows desk top operating systems do not.

COTS

Commercial off the shelf: typically applied to bought in hardware and software packages that are integrated into a product.

CRC (Cyclic redundancy code)

A method of error checking in which the properties of high-order binary polynomials are exploited to generate a relatively small number (the "checksum") by repeated binary operations on a data stream. The CRC algorithm is designed so that probability of the checksum from a data stream containing one or more errors being equal to that from an error-free data stream is statistically insignificant.

Data Dictionary

A data dictionary is a set of [metadata](#) (see below) that contains definitions and representations of the data in a database. It typically holds the following information: definition of data elements in the database; user names, and user's roles and privileges; the general database structure; space allocations; other technical information required by the database package that stores and accesses the data in the database.

Defensive Programming

A general set of programming techniques based on the idea that programs are responsible for protecting themselves against the effects of being passed bad data from diverse sources, including systems, users, and other programs. Effective defensive programming strikes a

balance between excessive protection, which makes a program bulky and inefficient and insufficient which leaves a program vulnerable to the effects of bugs in other programs that manifest themselves as unexpected input values, etc. More details are to be found in, for example [5].

Embedded software

Programs and data that permanently reside in ROM or flash memory. Embedded software may be immediately available to the CPU or, for faster execution, may be transferred to RAM first and then executed, this differs from PC software applications, which are stored on disk and must be loaded into RAM for execution. Embedded software controls an embedded system which is a special-purpose system where the computer is completely encapsulated by the device it controls, for example, a sensor or a measurement instrument. Some embedded software uses an embedded real time operating system (RTOS) while other implementations include their own real time executives that interface directly to the hardware.

EN14181:2004

Stationary Source Emissions standard covers the quality assurance of Automated Measuring Systems (AMS)

IEC 61508

This is a family of international standards published under the title *Functional Safety of Electrical/Electronic/Programmable electronic safety-related systems*. The standard covers the entire systems lifecycle and includes detailed requirements for functional safety, hardware, and software. Please refer to www.iec.ch for further details.

Metadata

Literally ‘data about data’: it provides the context for data so for a reference value its metadata may include a reference to the chemical tables from which the value was taken.

Protocol Stack

This is a software package that implements a communications protocol such as TCP/IP. It is called a ‘stack’ because communications protocols are often organised into layers such as: physical (hardware), data link, network message, transport, session, presentation, application.

QAL 1, 2, 3

These are quality assurance levels defined as part of EN14181 that cover the suitability of an Automated Measuring System (AMS) for its measuring task. QAL1 covers the suitability of an AMS for its measuring task, the QAL1 procedure is defined in EN ISO 14956; QAL2 covers the validation and commissioning of the AMS following installation; QAL3 is a procedure to check drift and precision to demonstrate that the AMS is in control during its operation and it therefore continues to function within the required specifications for uncertainty. This is achieved by conducting periodic zero and span checks of the AMS.

Regression Testing

Regression bugs occur whenever software functionality that previously worked as desired stops working or no longer works in the same way that was previously planned. Typically regression bugs occur as an unintended consequence of program changes. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged.

SCADA

Supervision Control and Data Acquisition: a SCADA package makes use of a points or tags data base that contains the data acquired by the package and which is also used to contain any output values that it uses to exert automatic control or to facilitate supervision by operators.

Software configuration management

The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software lifecycle.

Software Configuration Management involves identifying the configuration of the software (i.e., selected software works products and their descriptions) at given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software lifecycle. The work products placed under software configuration management include the software products that are delivered to the customer (e.g., the software requirements document and the code) and the items that are identified with or required to create these software products (e.g., the compiler).

A software baseline library is established containing the software baselines as they are developed. Changes to the baselines and the release of software products built from the baseline library are systematically controlled via the change control and configuration auditing functions of software configuration management. [*Quoted from the SEI Software Capability Maturity Model.*]

Software Component

A component is a program that can be re-used by a programmer and incorporated into diverse applications. Examples of software components are 'screen widgets' that provided buttons to click on or sliders to operate.

Software Package

A package is a body of software such as SQL Server or a SCADA package that is sold and supported as a product and which comes with its own documentation (see software component)

Structure-based testing

Testing that takes into account the internal mechanism/structure of a system or component. Types include branch testing, path testing, statement testing. Testing to ensure each program statement is made to execute during testing and that each program statement performs its intended function. Contrast with functional testing.

Static Analysis.

Analysis of a program performed without executing it. That is, the process of evaluating a system or component based on its form, structure, content, documentation.

Static Analyser.

Static analysers are tools that determine various properties of a program by inspecting its source code; unlike dynamic testing, the program is not compiled and run.

Unit testing

This is a test procedure used to validate that individual modules or other units of source code are working properly.

UTC

Coordinated Universal Time is a time scale that couples Greenwich Mean Time, which is based solely on the Earth's inconsistent rotation rate, with highly accurate atomic time. When atomic time and Earth time approach a one second difference, a leap second is calculated into UTC. UTC was devised on January 1, 1972 and is coordinated in Paris by the International Bureau of Weights and Measures. UTC, like Greenwich Mean Time, is set at 0 degrees longitude on the prime meridian.

Validation

Validation entails confirmation through the provision of objective evidence that the requirements for a specific intended use or application have been fulfilled. Simply put, validation has to answer the question: 'have we built the right product?'

Verification

Verification entails the evaluation, often by testing and/or review of the outputs (for example, specifications, designs, code) from each phase of the software lifecycle to confirm that the outputs are consistent with the inputs to the phase, for example that the design matches the input requirements, the code matches the design. Simply put, verification has to answer the question: 'is the product right?'

Annex C - Information about the NPL Best Practice Guides

In recent years there has been significant public investment in the UK in the production of “Best Practice Guides” (BPG) for measurement-related applications. At the same time there has been great activity on the standardisation of safety-related systems, particularly through norms like IEC 61508 [1], and its sector standards IEC 61511, IEC 62061, etc.

The National Physical Laboratory (NPL) as part of the Software Support for Metrology (SSfM) initiative has authored the BPGs. These publications [2], [3], [4], and others provide a valuable resource to developers, assessors, and users of measurement software; they are freely downloadable from <http://www.npl.co.uk/ssfm/download/index.html> . A particularly useful aspect of the BPGs is their exposition of the need for a Mathematical Specification, and the associated need for well-formed algorithms, the appropriate use of arithmetic, and an awareness of issues surrounding measurement uncertainty. Thus, in a measurement context there are two levels of guidance:

1. Generic software guidelines that must be followed according to the level of risk associated with the application.
2. Measurement specific guidelines.

Other guidelines have been produced and references to them can be found in the NPL BPGs. Experience with the BPGs shows that they can be adopted quite readily by SMEs (small and medium sized enterprises) and that they are not unduly burdensome.

Annex D - Sample Headings for a Software Quality Plan

This annex provides guidance on the information that should be made available in the Quality Plan.

Typical headings are as follows:

1. Introduction

Summary of project/product
List of deliverables

2. Risk Assessment

Choice of the software quality standard used in Part A

3. Quality System

Statement as to whether a QMS exists including copy of any certificate for the company's QMS, if one is available

4. Project Organisation and Control

Roles and responsibilities for all those involved in the project, including subcontractors

Outline Plan and milestones

Communications between members of the project:

- Policy on recording decisions
- Policy on storing emails and other correspondence
- Policy for meetings minutes, recording of reviews, etc

Change Control

Software configuration management

Progress meetings

Records and archiving

Training requirements

5. Software Lifecycle

Selection and justification of the software lifecycle that is to be used

Selection and justification of the tools and methods to be used, including a list of all tools and components

List of software standards to be used, including the coding standard

Procedure for handling problems (software defect reports) and non-conformances

Procedures for verification of the software, typically testing and reviews

Backups and security

List of reviews to be performed including:

- Independence Matrix (table showing that authors do not review their own work)
- Identification of the more critical parts of the application that require more detailed review

6. Design Phase

Definition of the contents of each of the design documents, depending on the methods used:

- List of prototype releases and their functionality
- Mathematical specification
- Software Design documents
- Software Acceptance Test Specification
- Software Design Verification Report and Code Reviews

7. Delivery Phase

Acceptance test policy, testing and sign off

Software release policy – release documents, method of delivery

Support/warranty commitments for the software