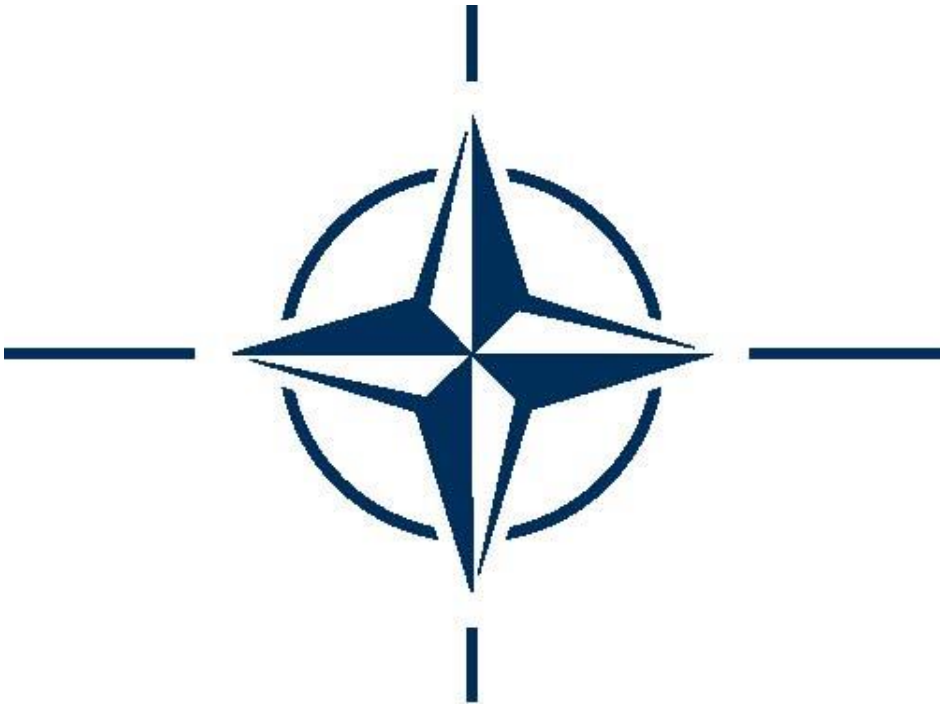
STANDARDS RELATED DOCUMENT

AQAP-2210-SRD.1

NATO GUIDANCE ON THE USE OF AQAP-2210 NATO SUPPLEMENTARY SOFTWARE QUALITY ASSURANCE REQUIREMENTS TO AQAP-2110 OR AQAP-2310

Edition A Version 1
NOVEMBER 2018



NORTH ATLANTIC TREATY ORGANIZATION

Published by the
NATO STANDARDIZATION OFFICE (NSO)

INTENTIONALLY BLANK

NORTH ATLANTIC TREATY ORGANIZATION (NATO)

NATO STANDARDIZATION OFFICE (NSO)

NATO LETTER OF PROMULGATION

8 November 2018

1. The enclosed Standards-Related Document, AQAP-2210-SRD.1, Edition A, Version 1, NATO GUIDANCE ON THE USE OF AQAP-2210 NATO SUPPLEMENTARY SOFTWARE QUALITY ASSURANCE REQUIREMENTS TO AQAP-2110 OR AQAP-2310, which has been approved in conjunction with AQAP-2110 by the nations in the Life Cycle Management Group, is promulgated herewith.

2. AQAP-2210-SRD.1, Edition A, Version 1, is effective upon receipt and replaces the content of Annex E to AQAP-2009, Edition 3, which has been cancelled.

4. This publication shall be handled in accordance with C-M(2002)60.

Zoltán GULYÁS
Brigadier General, HUNAF
Director, NATO Standardization Office

INTENTIONALLY BLANK

TABLE OF CONTENTS

ANNEX A – Guidance for a Software Project Quality Plan (SPQP)

FOREWORD

This document has been prepared and issued to provide information and guidance on the application of: AQAP-2210 "NATO Supplementary Software Quality Assurance Requirements to AQAP-2110 or AQAP-2310".

It aims to contribute to commonality of interpretation of these requirements between Supplier and Acquirer. It is not intended as a procurement document. Its content has no legal or contractual status, nor does it supersede, add to or cancel any of the AQAP-2210 requirements. Copies of this document may be made available to industry to facilitate the use and understanding of AQAP-2210.

Each paragraph (and subparagraph) of AQAP-2210 is listed in this document only with his title in bold italic, followed by the related guidance (or by the sentence "No guidance required"). The guidance offers some suggestions as to subjects and factors to be considered.

Because of the multiplicity of conditions that can exist (dependent on such factors as the type of work or process, the devices used and the skill of personnel involved), this guidance should not be considered as all-encompassing, nor should it be considered as imposing specific means or methods for meeting contract requirements. Managers must be aware that other means or methods could be used to meet these requirements.

The fundamental requirements of AQAP-2210 are mandatory for all software projects, but the sub-level application of tools, methods and procedures can be implicitly tailored to the needs of individual projects.

This publication supersedes the Annex E to the AQAP-2009 Edition 3.

CHAPTER 1 INTRODUCTION

1.1 PURPOSE

In addition to the requirements (a. through e.) strictly related to the software development process, this Publication also addresses the system-software relationship. The additional requirements (f. and g.) provide for the meaningful participation of software engineering in system engineering, and for addressing the system/software critical issues, like safety and security.

1.2 APPLICABILITY

No guidance required.

1.3 REFERENCED DOCUMENTS

No guidance required.

1.4 DEFINITIONS AND ACRONYMS

No guidance required.

CHAPTER 2 REQUIREMENTS

2.1 SOFTWARE QUALITY SYSTEM (SQS)

AQAP-2210 normally presumes the existence of a documented Software Quality System (SQS); the SQS includes not only the technical processes of software development but also the managerial processes.

The company-wide SQS should address the range of software that the Supplier produces. Different methods, procedures and tools may be called for dependent on the type of application, size of project, number of people involved etc.

Review of the SQS is defined as a periodic, systematic and documented evaluation of the status and adequacy of the system elements. Such a review is conducted by or on behalf of top management to ensure that their objectives are reached, and to reveal non-conformances or irregularities in the system elements that require improvement.

For the SQS to be effective it should support the requirements of AQAP-2210 and any additional requirements imposed by the contract.

2.2 PROJECT SOFTWARE QUALITY MANAGEMENT ACTIVITIES

2.2.1 General

The Project Software Quality Management Activities should comprise the planning and implementation activities necessary for the successful execution of the project. The project activities mentioned in paragraphs 2.2.1 a, b, c and d are elaborated in paragraphs 2.2.3 through 2.2.7. Guidance on these activities is given on each subparagraph.

The depth of Project Software Quality Management Activities will be influenced by the contractual requirements and constraints, like complexity, criticality, size, Acquirer involvement etc. Therefore, as a prerequisite to the planning of the activities, the Supplier should undertake a formal contract review, to ensure all requirements and constraints are clearly defined and understood.

Evaluation of the activities by the Acquirer, should initially make use of objective evidence of the Supplier's own reviews. Where no objective evidence exists that such reviews have been conducted, it should be regarded as a serious quality system shortcoming and consequential risk.

2.2.2 Software Project Quality Plan (SPQP)

The SPQP and its contents should be recognized by Acquirer and Supplier as an indication of the understanding, commitment and compliance with the quality requirements of the contract.

Suppliers should begin to plan their quality related activities at the earliest possible phase of the contract.

The SPQP should address "contract-specific" quality activities and should not be a reiteration of the SQS requirements, as detailed in the Supplier's Quality Manual/Documentation. However, reference to these requirements in the SPQP may be necessary.

An SPQP may be required in response to an Invitation-to-Tender / Request-for Proposal, or under the contract, and should be prepared as a precursor to the software development process.

A possible layout for the SPQP may be found at Annex A.

2.2.3 Identification and Review of Software Requirements

The software requirements may be derived from an expressed need (but not necessarily specified) by the Acquirer. Often the Supplier does not fully understand the Acquirer's problem and field of application; both contractual parties may work together to come to a formal contractual agreement on what the completed software must do.

The key to achieving effective software development is for both the Supplier and Acquirer to achieve a common understanding of the requirements. Therefore, the Supplier should ensure that the software requirements are described in such a way that their interpretation is not in doubt. Any omissions, misunderstandings or inconsistencies in the requirements should be addressed as early as possible in the software development process when they are easier to correct. The Supplier should also be satisfied that each requirement is defined in such a manner, that its achievement can be ultimately subjected to validation by a prescribed method. If this is in doubt, the matter should be brought to the attention of the Acquirer.

Often the software requirements are derived from higher level (i.e. system or subsystem requirements), in that case the task at hand consists in ensuring that all applicable higher level requirements have been correctly translated into software requirements and no new requirements have been introduced.

"Development constraints" are restrictions on the development process, which shift greater design responsibility to the Organization setting the restrictions and need to be separated from the software product characteristics. Examples are: Design standards and conventions, languages, computer hardware and Acquirer supplied software.

Consideration should be given to the provision of training of personnel (both Acquirer and Supplier).

Definitions of software quality characteristics can be found in ISO/IEC 25010: 2011 (confirmed in 2017). These include Functional suitability, Performance efficiency, Compatibility, Usability, Reliability, Security, Maintainability and Portability.

2.2.4 Management

2.2.4.1 Software Development Process

The software development has a strong impact on the quality of the software product.
Software development models are simplified, abstract representations of a systematic approach to the software development process and, together with methods and tools, are the most important quality management elements.

Development models are a basis for detailed planning of project software quality management activities, including time and budget aspects, and support continuing improvement of the software development process.

The models structure the processes in logical and coordinated activities and tasks, and clearly relate development activities to the associated evaluation activities.

There are several types of development model, e.g. Waterfall Model, Spiral Model etc.; AQAP-2210 gives the Supplier freedom in the choice of development model. The selection, definition and application of a specific model depends on the complexity, criticality and type of software to be developed. Whatever model is selected, it may be tailored to meet the specific contract requirements. However, the model should achieve the issues mentioned in paragraph 2.2.4.1 (a. through o.) of AQAP-2210. Where possible, account should be taken of International or National standards defining these models.

It should be noted that whilst paragraph 2.2.4.1 is under the parent paragraph 2.2.4 (Management), the software development process also includes the technical processes described in paragraph 2.2.5 (Software Engineering) and paragraph 2.2.6 (Evaluation, Verification and Validation).

The model should clearly describe all the primary processes, e.g. design, coding, testing etc., together with all the supporting and organizational processes, e.g. project management, quality management, configuration management etc., undertaken throughout the software life-cycle. The description of the processes should not only include the identification of the tasks, but also the roles (architect, tester, etc.), inputs, outputs, the start and end criteria, control points, related measurement (when applicable) and all the technical and managerial aspects. This is in order to reduce the complexity of the software development process, thus giving improved visibility, integrity and control of the software product itself.

A developed software integration strategy should include verification criteria for software items, consistent with the software design and the prioritized software requirements:

i. items are developed that ensure compliance with the software requirements allocated to the items;
ii software items are verified using the defined criteria;
iii software items defined by the integration strategy are produced;
iv results of integration testing are recorded;
v consistency and traceability are established between software design and software items;
vi a regression strategy is developed and applied for re-verifying software items when a change in software units (including associated requirements, design and code) occur.

2.2.4.2 Organization

It is important to define the inter-relationship of organizational elements and groups, since activities of the development process may overlap and be executed iteratively. It is also important that the organizational structure indicates the co-operation and consultation between elements or groups, and also indicates the point(s) of contact with the Acquirer.

The degree of independence required for personnel performing evaluations, verifications and validations, may depend upon the circumstances of the particular contract and/or Supplier concerned. In most instances suitably, independent personnel may be found amongst the peers of those who developed the software product or performed the activity being subjected to evaluation, verification or validation. Sometimes it may be necessary to seek such personnel within other areas or organizations, internal or external to that of the Supplier. Where special independence requirements pertain, such as for safety critical software, these should be defined in the contract.

A determination of the necessary independence of the verification effort should be required based on the potential of an undetected error in a system or software for causing:

i death or personal injury;
ii mission failure;
iii catastrophic equipment loss or damage;
iv the maturity of and risks associated with the software technology to be used;
v financial loss.

2.2.4.3 Non-conforming Software

Non-conforming software should be clearly identified as such and segregated from conforming software. Once "conforming" software is released and made available for use, e.g. to test areas or placed in the software library, its status should be clearly indicated and made known. Upon becoming non-conforming software, e.g. after failing a test or a confirmed customer fault report, it should be segregated by clearly indicating its non-conforming status and taking appropriate action to control access to the software.

2.2.4.4 Corrective Action

The primary aim of the corrective action process should be to prevent the recurrence of a problem. It will also be a source of data for the review of the SQS. The analysis of problems should consider the effectiveness of any processes involved, be they technical or managerial.

2.2.4.5 Sub-supplier Management

The main Supplier is responsible for ensuring that sub-contracted products and services comply with the requirements and conditions of the main contract, even if the entire software package is sub-contracted.

The Supplier should select Sub-suppliers, using an appropriate procedure, based on their ability to meet sub-contract requirements, including quality. The Sub-suppliers previously demonstrated performance should also be taken into account.

The Sub-supplier's SPQP should be related to the main Supplier's SPQP. This relationship of plans is necessary for configuration management and specifically to coordinate changes to configuration items.

2.2.4.6 Software Configuration Management (SCM)

In software development and/or maintenance, a strong relationship exists between SCM and software quality assurance. Without a disciplined SCM process, one of the means for quality assurance is missing.

Configuration management is a discipline for identifying, controlling, tracking and auditing the versions of each software configuration item. SCM should be applied in a cost-effective manner, in terms of organization, methods, tools and procedures, whilst ensuring the necessary integrity and traceability of the software product. Configuration management can be automated or undertaken by manual methods.

Temporary changes to delivered software, sometimes known as patches, should be strictly controlled. Where such changes are introduced into software they should be carried out in accordance with defined procedures. In any event, follow-up action should confirm the validity of the change and where appropriate formally introduce it under normal configuration management procedures.

2.2.4.7 Off-the-shelf Software

The reason why off-the-shelf software should be placed under Configuration Management is because it affects the integrity of the developed software. This is true, whether it is a component of the software under development or a tool to assist the development of such software.

Off-the-shelf software by definition includes "government furnished software" (see paragraph 1.4.1, item 6, of AQAP-2210). Government furnished software places constraints on the Supplier in terms of development freedom and responsibility.

The evaluation and validation of the ability of off-the-shelf software to perform the required functions may include such considerations as Intellectual Property Rights, licensing arrangement and configuration management controls.

The Supplier should be able to provide objective evidence (e.g. validation reports, configuration reports, etc.) that the use of off-the-shelf software has been evaluated and is under control.

Documentation requirements for off-the-shelf software may include functional and interface specifications.

2.2.4.8 Non-deliverable Software

Examples of non-deliverable software which may be employed in the development of deliverable software are: emulators, test harnesses and driver programs, stub routines etc.

It is essential that all such software is placed under configuration management, since it directly affects the integrity of the developed software.

2.2.4.9 Quality Records

Quality records may be in the form of EVV reports, test results, corrective action reports etc. They can be the formal results of both main Supplier and Sub-supplier activities.

2.2.4.10 Documentation

There are several reasons for documentation retention, e.g. to:

i facilitate the correction of faults;
ii allow traceability of product;
iii provide evidence in liability disputes; and
iv allow for the re-creation of the software development environment.

The Supplier should therefore identify all necessary documentation to allow the successful completion of such tasks.

Documentation should include but not be limited to:

i requirement specifications;
ii architectural and design documents;
iii user documentation;
iv testing documentation;
v quality records; and

vi software licenses, e.g. seats, number of platforms, number of users, reuse, interfaces, replication etc.

Documentation can be in electronic or hard copy.

2.2.4.11 Handling and Storage of Software Media

Any media on which software is stored should be handled in such a way that the integrity and confidentiality of the stored information is assured. It is therefore necessary that activities likely to influence the quality are recognized and steps taken to avoid degradation of the material or the information. The Supplier should describe the storage, storage security, environment, access to and release from storage, in a procedure that also indicates how these activities are controlled.

Software may be considered as "critical" because of its safety, security or other implications. However, it may also be considered as critical if, for example, its loss would seriously delay the successful completion of the software development programme.

Adequate antivirus and firewall protection should be provisioned.

2.2.4.12 Replication and Delivery

No guidance required.

2.2.5 Software Engineering

Software engineering is the defined, documented and controlled, engineering discipline which develops the software products, with the use of methods, tools and procedures that are established and documented. The methods and procedures applied in software engineering should be consistent with the development model and criteria defined in paragraph 2.2.4.1.

Software tools may be related to the specific methods or techniques identified or provide support to other aspects of the software life-cycle. Some tools may be phase-independent, for example those associated with configuration management or quality assurance activities.

Software tool validation may entail one or more of the following:

i certification provided by a recognized body that the tool has been subject to specified tests or validation processes;
ii establishment, with the tool supplier, that the tool meets required criteria through the Supplier quality system and evidence of appropriate tests;
iii identification of appropriate tests to be applied to the tool and any upgrades;
iv monitored usage of the tool during support to the development of the software product;
v feedback from a user group.

To facilitate software product maintenance, the availability of longer-term support for software tools is an important aspect that should not be ignored in their evaluation.

2.2.6 Evaluation, Verification and Validation (EVV)

Although EVV is an integral part of the management and technical process (paragraphs 2.2.3, 2.2.4 and 2.2.5), due to its importance in Quality Management, the EVV process is addressed in this discrete paragraph.

Due to the inter-relationship of Evaluation, Verification and Validation, these activities should be planned as a whole. The allocation of resources and time, and the selection of methods and techniques should be done in such a way that the entire EVV process is optimized.

The correct execution of EVV tasks has a considerable impact on the quality of the end product. This process requires, in general, the use of a considerable amount of resources, so that it should be carefully planned in terms of availability of qualified personnel, schedule, cost and test environment.

The level of EVV should be tailored to the level of complexity and/or criticality of the software and to the requirements of the contract and should involve optimum use of existing techniques and standards available.

This paragraph is also related, as a by-product, to the evaluation and improvement of the Software Quality System, e.g. it monitors the application of the established procedures and measures the correctness and efficiency of those procedures. This evaluation process is based on data provided by project groups and is a contract-independent activity (see paragraph 2.1).

2.2.6.1 Testing

In general, tests are much more effective the earlier they are addressed/conducted in the software development process. Planning for testing and the specification of tests should therefore take place as early as possible.

During test planning, if required by the contract, consideration should be given to the involvement of Acquirer personnel in test activities.

2.2.6.2 Reviews

Software-related review activities may be known under various headings, including design reviews, peer reviews, walk-throughs, inspections, document reviews, desk checks etc.

Experience has shown that significant software errors are introduced during the early phases of the software development process. Emphasis should therefore be placed on design reviews at these stages to promote the early detection and resolution of errors.

2.2.7 Maintenance

Software Maintenance is the process of maintaining software after initial delivery and installation, e.g. to correct defects, modify/adapt functions, or improve/augment performance.

2.3 HUMAN RESOURCES

Personnel performing specific assigned tasks shall be qualified on the basis of appropriate education, training and/or experience as required:

i design methods;
ii specific programming languages;
iii tools, techniques;
iv computer platforms and target environment.

2.4 ACQUIRER ACCESS AND INVOLVEMENT

No guidance required.

ANNEX A – Guidance for a Software Project Quality Plan (SPQP)

In AQAP-2210 (paragraph 2.2.2) a Software Project Quality Plan (SPQP) is required and requirements for the SPQP may be found throughout AQAP-2210.

As written in the same paragraph, the SPQP may be a discrete document or part of another plan that is prepared under the contract.

A possible layout of the SPQP may be found below. It should be noted however, that it is a guide and that the SPQP, documenting the software management activities related to a specific project, is the sole responsibility of the Supplier.

The requirements for evaluation of the software quality management activities by the Acquirer, are laid down in AQAP-2210, paragraph 2.2.1.

If stipulated in the contract, the SPQP shall be offered to the Acquirer for agreement, as called for in AQAP-2210, paragraph 2.2.2.

0 COVER SHEET

The cover sheet should carry the signature of approval of those organizational elements having responsibilities identified in the SPQP. It may also indicate the name(s) of the organization(s) for whom the SPQP is prepared.

1 INTRODUCTION

1.1 Purpose

A graphic presentation of the project may be included, or reference to where else it may be found. This could e.g. summarize the milestones and the numbers and positions of sites (or subsystems).

1.2 Scope

If the entire project is covered by plans, such as:

- project management plan,
- time schedule/milestone,
- work breakdown structure,
- list of deliveries,
- risk assessment,
- test plan and specifications,
- installation handbook,

reference to the plans can be made in paragraph 1.2 and the only reference needed in paragraph 1.4 may be AQAP-2210.

1.3 Maintenance of the SPQP

e.g. change control procedures.

1.4 Referenced Documents

1.5 Relationship to other plans
1.6 Definitions and Acronyms

2 PROJECT DESCRIPTION
2.1 Project Overview (or reference to where else it may be found)
2.2 Assumptions
2.3 Deliverable Products

3 MANAGEMENT
3.1 Software Development Process
3.2 Organization
3.3 Non-conforming Software
3.4 Corrective Action
3.5 Sub-supplier Management
3.6 Software Configuration Management (SCM)
3.7 Off-the-Shelf Software
3.8 Non-deliverable Software
3.9 Quality Records
3.10 Documentation
3.11 Handling and Storage of Software Media
3.12 Replication and Delivery

4 SOFTWARE ENGINEERING
4.1 Software Engineering Environment
4.2 Methods, Procedures, Standards
4.3 Development Documentation

5 EVALUATION, VERIFICATION AND VALIDATION (EVV)
5.1 Testing
5.2 Reviews

6 MAINTENANCE

7 HUMAN RESOURCES

8 ACQUIRER ACCESS AND INVOLVEMENT

INTENTIONALLY BLANK

AQAP-2210-SRD.1(A)(1)