FP6-2004-27020

Access-eGov

Access to e-Government Services Employing Semantic Technologies

Instrument: STREP

Thematic Priority:
SO 2.4.13 Strengthening the integration
of the ICT research effort in an enlarged Europe

D1.3 Project Quality Plan

Start date of project: January 1, 2006 **Duration:** 36 months

Date of submission: July 17, 2006

Lead contractor for this deliverable: University of Regensburg

Revision: Final 2.2

Dissemination level: PU

Acknowledgement: The Project is funded by European Commission DG INFSO under the IST Programme, contract No. FP6-2004-27020.

Disclaimer: The content of this publication is the sole responsibility of the authors, and in no way represents the view of the European Commission or its services.

FP6-2004-27020 **Access-eGov**
Access to e-Government Services Employing Semantic Technologies

D1.3 Project Quality Plan

Workpackage:	WP1	**Task:**	T1.4
Date of submission:	July 17, 2006		
Lead contractor for this deliverable:	University of Regensburg (UR)		
Authors:	Jan Kolter (UR) Stefan Dürbeck (UR) Marek Skokan (TUK) Kamila Dec (EMAX) Ralf Klischewski (GUC)		
Version:	2.2		
Revision	Final		
Dissemination level:	PU		

Project partners:

Technical University of Kosice (TUK), Slovakia (Coordinator); University of Regensburg (UR), Germany; German University in Cairo (GUC), Egypt; Intersoft, a.s. (IS), Slovakia; EMAX S.A. (EMA), Poland; Kosice Self-Governing Region (KSR), Slovakia; Cities on Internet Association (COI), Poland; e-ISOTIS (ISO), Greece; Municipality of Michalovce (MI), Kosice; City Hall of Gliwice (GLI), Poland; State Government of Schleswig-Holstein (SHG), Germany.

Abstract:

The intention of this Project Quality Plan is to define quality management procedures to validate that all deliverables are completed with an acceptable level of quality. It defines mechanisms of how to deliver products that meet the quality standards as expected by all project partners.

All procedures described herein are tailored to meet the requirements of the Access-eGov Project.

This Quality Plan will be applicable to written deliverables (reports, working papers) as well as software and adjoining types of deliverables. All project partners will have to adhere to the quality processes as outlined in this document.

The procedures described in this Project Quality Plan will come into effect after submission of this document to the European Commission.

Document Sign-off

Nature of Sign-off (Reviewed/Approved/Submitted)	Name	Role	Participant short name	Date
Reviewed	Jan Kolter	TL	UR	July 17, 2006
Approved	Tomas Sabol	WPL	TUK	July 17, 2006
Submitted	Tomas Sabol	PM	TUK	July 17, 2006

Document Change Record

Date (d/m/y)	Version	Contributor(s)	Change Details
17/05/2006	Draft 0.5	Jan Kolter, Stefan Dürbeck	Major changes to the overall structure
12/06/2006	Draft 0.6	Kamila Dec, Marek Skokan, Stefan Dürbeck	Updated chapters 3, 4 and 6
23/06/2006	Final 1.0	Jan Kolter, Ralf Klischewski	Added chapter 5
06/2006	Final 2.0	Jan Kolter, Stefan Dürbeck, Marek Skokan, Kamila Dec	Updated chapters 3, 4 and 5
06/2006	Final 2.1	Jan Kolter, Kamila Dec, Stefan Dürbeck	Minor changes
06/2006	Final 2.2	Jan Kolter, Stefan Dürbeck, Ralf Klischewski	Updated chapter 5

Files

Software Products	User files / URL
Microsoft WORD	

Content

Table of Figures

List of Acronyms and Participant short names

EC	European Commission
IST	Information Society Technologies
PM	Project Manager
PCC	Project Co-ordination Committee
SDC	Software Development Committee
LOPA	Liaison Officers to Public Administration
WP	Work Package
WPL	Work Package Leader
TL	Task Leader
QAM	Quality Assurance Manager
COI	Cities on Internet Association
EMA	EMAX S.A.
GLI	City Hall of Gliwice
GUC	German University in Cairo
IS	Intersoft, a.s.
ISO	e-ISOTIS
KSR	Kosice Self-Governing Region
MI	Municipality of Michalovce
SHG	State Government of Schleswig-Holstein
TUK	Technical University of Kosice
UR	University of Regensburg

1 INTRODUCTION

The purpose of this Project Quality Plan is to establish Quality Assurance Procedures for Access-eGov Project to ensure a consistent quality level of all deliverables for the whole project runtime.

Therefore, deliverables in the sense of this Project Quality Plan are:

- written documents (papers, reports, minutes), as well as
- software (source code, executables and related documentation)

This document will be applicable to both types of deliverables. All project partners will have to implement the quality processes as outlined in this document.

The first part provides an overview of the quality management structure in Access-eGov and defines the responsible project members. Chapters 3 and 4 outline the quality strategy for written documents and software deliverables. Change management procedures are described in Chapter 5 and the overall auditing process for Access-eGov is specified in Chapter 6.

Relevant document templates for use in daily quality assurance procedures are enclosed in the Appendix (Chapter 8) to this Project Quality Plan.

Supporting the daily quality assurance work, Access-eGov will make use of an online project management tool called dotProject[1]. It allows other project members to control the overall and task-related work progress.

The Quality Plan has been developed in consultation with all the project partners.

The procedures as outlined in this Project Quality Plan will come into effect after its final version is made accessible to all the project partners.

[1] http://esprit.ekf.tuke.sk/dotproject/

2 QUALITY MANAGEMENT STRUTURE

All project partners of Access-eGov are committed to the quality assurance procedures as outlined in this Project Quality Plan. Project members of Access-eGov hold the following responsibilities as of 30th of June 2006 (agreed at the Kick-off meeting and subsequent discussion).

Project Manager (PM)

Tomas Sabol (TUK)

Project Co-ordination Committee (PCC)

The PCC consists of:

1) Tomas Sabol (TUK) – chair
2) Guenther Pernul (UR)
3) Ralf Klischewski (GUC)
4) Julius Kovac (IS)
5) Jacek Polrolniczak (EMA)
6) Gabriela Hajdukova (KSR)
7) Darius Wozniak (COI)
8) Karel Van Isacker (ISO)
9) Jozef Bobík (MI)
10) Krzystof Zbrozek (GLI)
11) Christiane Coenen (SHG)

Project Administrator

Ivana Gerhartova (TUK)

Financial Manager

Maria Geralska (TUK)

Software Development Committee (SDC)

1) Marian Mach (TUK) –chair
2) Daniel Wiechzinski (EMA) – co-chair
3) Rolf Schillinger (UR)
4) Martin Tomasek (IS)
5) Jakub Staniewicz (COI)

6) Norman Wittmüss (SHG)
7) Stefan Ukena (GUC)
8) Karel Van Isacker (ISO)
9) Gabriela Hajdukova (KSR)

Liaison Officers to Public Administration (LOPA)

1) Dariusz Wozniak (COI)
2) Piotr Karasewicz (GLI)
3) Michal Kmec (KSR)
4) Tomáš Bielovský (MI)
5) Christiane Coenen (SHG)

Workpackage Leader (WPL)

1) WP1 – Tomas Sabol (TUK)
2) WP2 – Ralf Klischewski (GUC)
3) WP3 – Martin Tomasek (IS)
4) WP4 – Rolf Schillinger (UR)
5) WP5 – Daniel Wiechzynski (EMA)
6) WP6 – Jan Hreno (TUK)
7) WP7 – Karol Furdik (IS)
8) WP8 – Dariusz Wozniak (COI) and Gabriela Hajdukova (KSR)
9) WP9 – Karel Van Isacker (ISO)

Task leaders (TL) – for the tasks running in first six months of the Project:

1) T1.1, T1.2, T1.3 – Tomas, Sabol (TUK)
2) T1.4 – Guenther Pernul (UR)
3) T2.1, T2.4 – Ralf Klischewski (GUC)
4) T2.2 – Dariusz Wozniak (COI)
5) T2.3 – Jan Kolter (UR)
6) T3.1, T3.3 – Martin Tomasek (IS)
7) T3.2 – Rolf Schillinger (UR)
8) T7.2 – Stefan Ukena (GUC)
9) T7.3 – Karol Furdik (IS)
10) T9.1, T9.3 – Karel Van Isacker (ISO)

This list will be updated on a regular basis.

Quality Assurance Manager (QAM)

The QAM's task is to ensure that the final product will be properly tested and that it will conform to agreed standards of operation, quality, safety, security, etc.

This position is in charge of activities related to Quality Assurance and will safeguard the adherence to Quality Management procedures as described herein. The QAM is considered to be the responsible contact partner in case of questions related to Quality management issues arising in daily project work and from Public Administration Partners. In addition, the QAM will produce quality guidelines, outline quality criteria and product descriptions for each deliverable.

The QAM will cooperate closely with the Software Development Committee, LOPAs and the PCC. The QAM will provide feedback to LOFAs and SDC and will advise the Project Manager and the Project Co-ordination Committee on the projects quality aspects throughout the project.

Quality Assurance Manager is Kamila Dec (EMA).

3 QUALITY ASSURANCE PROCEDURES FOR WRITTEN DELIVERABLES

This chapter is considered to describe the quality assurance procedures for written Access-eGov deliverables. It outlines a workflow with standard rules and procedures related to document production that all the partners apply throughout the project.

The document quality procedure is applicable:

- by all partners,
- for all report deliverable to the European Commission, and
- for documents exchanged between project partners.

All documents produced within Access-eGov with the intention of being published, have to follow these guidelines.

All procedures described below are conducted by Access-eGov project partners, unless explicitly mentioned otherwise. These internal reviews are in no way related to review conducted by the EC.

3.1 INTRODUCTION

Internal reviews are usually conducted on a mutual basis inside the project consortium to ensure that a deliverable complies with the quality requirements that all project members previously agreed upon. Before internal dissemination among the project partners and before external publication or submission to the EC, all Access-eGov deliverables have to undergo an internal review procedure as depicted in interrelated Figures 1 and 2 at the end of this section (page 14 and 15).

Public deliverables may be accessible via the Project website or even published in the media, in contrast to internal deliverables (which are for the Project and EC use only). The review procedures are similar in all of these cases and differ only in their respective follow-up action after approval by the WPL.

3.2 DOCUMENT STATUS

The current status of a document appears on the document front page. Deliverables in the Access-eGov project can belong to three different types of document classes:

Draft	No final approval yet. The project partners involved in creating the deliverable have completed work on the document, but it is still open to major comments.
Final	The deliverable has been completed for distribution among the project partners. The deliverable can now be sent to other project partners for a review of the Final version. Comments upon the Final version are mostly expected to imply minor changes.
Final/Approved	This implies that no further corrections will have to be added to the final version; publicly available documents (media releases, etc.) can now be

	published upon consent from the PM. Deliverables to the EC will then be submitted by the PM. The status "Approved" appears in the document sign-off section on page 3. It will have to be assigned by the WPL (see Chapter 3.5).

3.3 PRECONDITIONS

A precondition for performing an internal review is the deliverable's completeness, i.e. a completed intermediate step of a deliverable's generation process. The TL will have to decide when the deliverable is complete and ready for a first peer review among project partners involved in the creation of the document (see Figure 1 on page 14).

Usually, internal quality assurance procedures in the very sense of this chapter will most likely be performed on completed deliverables seeking final approval by the WPL and the PM.

3.4 QUALITY ASSURANCE ACTIVITIES

The internal review activities are intended to commence only after the first draft version has been completed (and all major parts from other involved project partners have been added). All procedures described in the following refer only to the internal processes of the Access-eGov project.

After completing a deliverable in form and content, it has to be reviewed internally by other project partners. Each deliverable is thereby evaluated against correctness criteria that all the responsible project members agreed upon beforehand. The criteria for written deliverables are to a certain level subject to the project member's individual judgement.

In most cases, comments via reliable and traceable electronic correspondence (email) should suffice. The author(s) shall then update the deliverable according to these review comments until a new version of the deliverable is ready to be issued to the project partners. The QAM (who will also conduct Auditing as outlined in Chapter 6) will be notified about the beginning of internal review procedures (e.g. QAM will receive copies of relevant e-mails).

Quality criteria as applied to written deliverables in Access-eGov are as follows:

Format	the correct format of the document, the presentation, the identification, the front pages, the summary, the glossary, the annexes, production rules according to the document templates and additions in this PQP;
Internal coherence	coherence of the content itself regarding all the information necessary for clear comprehension;
External coherence	coherence and compliance of the document with other relating documents to prevent contradiction; If the deliverable has to be incorporated into another document, the resulting document's coherence also requires checking.

Generally, the internal Quality Assurance Process consists of the following steps that should begin at the latest four weeks before reaching the deadline for submission. Upon accord with the QAM, the WPL may adjust the timeline for the review to the actual situation.

- **Peer Review**

  The peer review will be conducted by all the authors themselves on Task level. After completing the deliverable, the TL decides about conducting a first draft review (called "peer review" here): the TL will electronically forward copies of the deliverable to all involved project partners and ask for comments and suggestions. This step usually will not require standardized communication or procedures and will be conducted informally. It is supposed to last up to one week and should start not later than 3 weeks before the deadline (see Figure 1 on page 14).

  According to the nature of the comments, the TL decides about re-work or assigns the document status "Final". The TL will then have to fill in a new entry (entitled "Reviewed") in the document sign-off section on page 3 of the written deliverable. It can now be forwarded to the WPL for a review of the Final version.

- **Final Internal Review**

  After completion of a Final document version, the WPL will conduct an internal review by requesting comments from the PM and all other project partners in Access-eGov that are invited by the WPL and by the PM for this review. The Final version will be forwarded electronically to all the project partners and is accessible for commenting. Based on the nature of the feedback, the WPL will decide whether the deliverable needs to undergo further re-work or whether it can be approved.

  This last decision implies adding a new entry (entitled "Approved") to the sign-off section on page 3. Final Internal Review will have to be conducted at the latest two weeks before the deadline and should last up to one week (see Figure 2 on page 15).

After conducting these quality assurance procedures on Task/WP and Project level, the deliverable has Final status and two entries in its Document Sign-Off section: "Reviewed" (added by the TL after Peer Review) and "Approved" (added by the WPL after Final Internal Review).

Submission

After approval of the Final version, the deliverable will be forwarded to the PM for submission to EC. It is here, where the PM has the last chance to trigger further treatment, in case he refuses submission. The deliverable will, in that particular case, be directed to the responsible WPL to undergo another rework cycle.

Otherwise, the deliverable will be submitted to the EC on time and with the expected level of quality (i.e. "submitted"). Before, the PM has to add an entry in the document's sign-off section (entitled "Submitted") as a formal recognition of the document's completeness.

Follow-Up

Submitted deliverables will have to be stored in an adequate way on the internal section of the Project website.

Public deliverables will have to be made accessible on the publicly available part of the Access-eGov website.

3.5 DOCUMENT SIGN-OFF

The different review stages of a written deliverable will have to be signed off with an electronic entry in the sign-off section on page 3 in each of the deliverables.

The document sign-off section should include the following tabular fields (as can be seen on page 3 of the PQP):

- Nature of Sign-off (Reviewed/Approved/Submitted)
- Name
- Role (e.g. TL, WPL, PM)
- Participant short name
- Date

After submission to the EC, a document should have at least three entries in its sign-off section (page 3):

- one by the TL (entitled “Reviewed”) and filled in after successful peer review on Task Level,
- one by the responsible WPL (entitled “Approved”) and finally
- one sign-off entry by the PM (entitled “Submitted”) that has to be filled in just before submission to the EC.

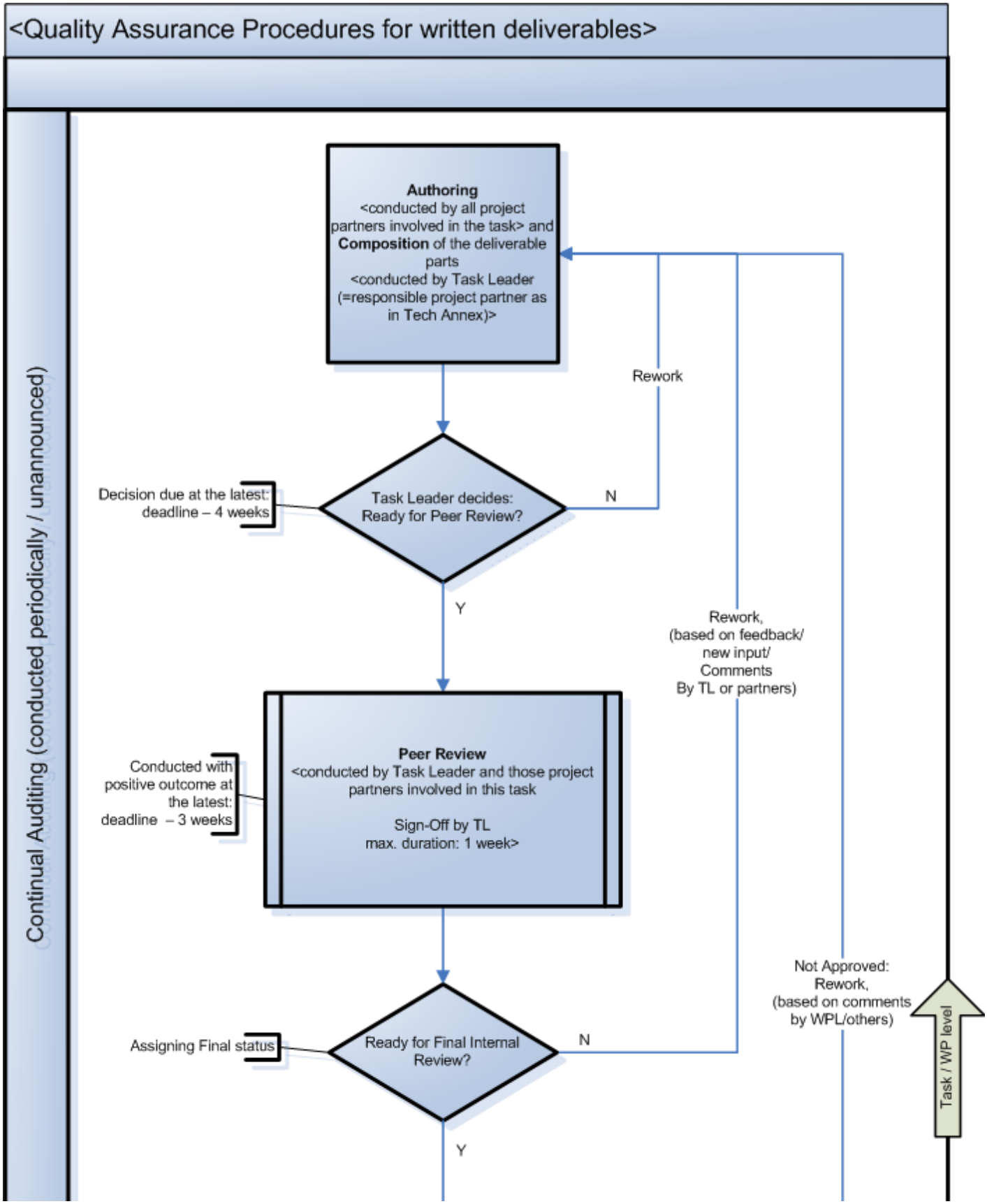

Figure 1 Quality Assurance Process for written deliverables (Task/WP level)

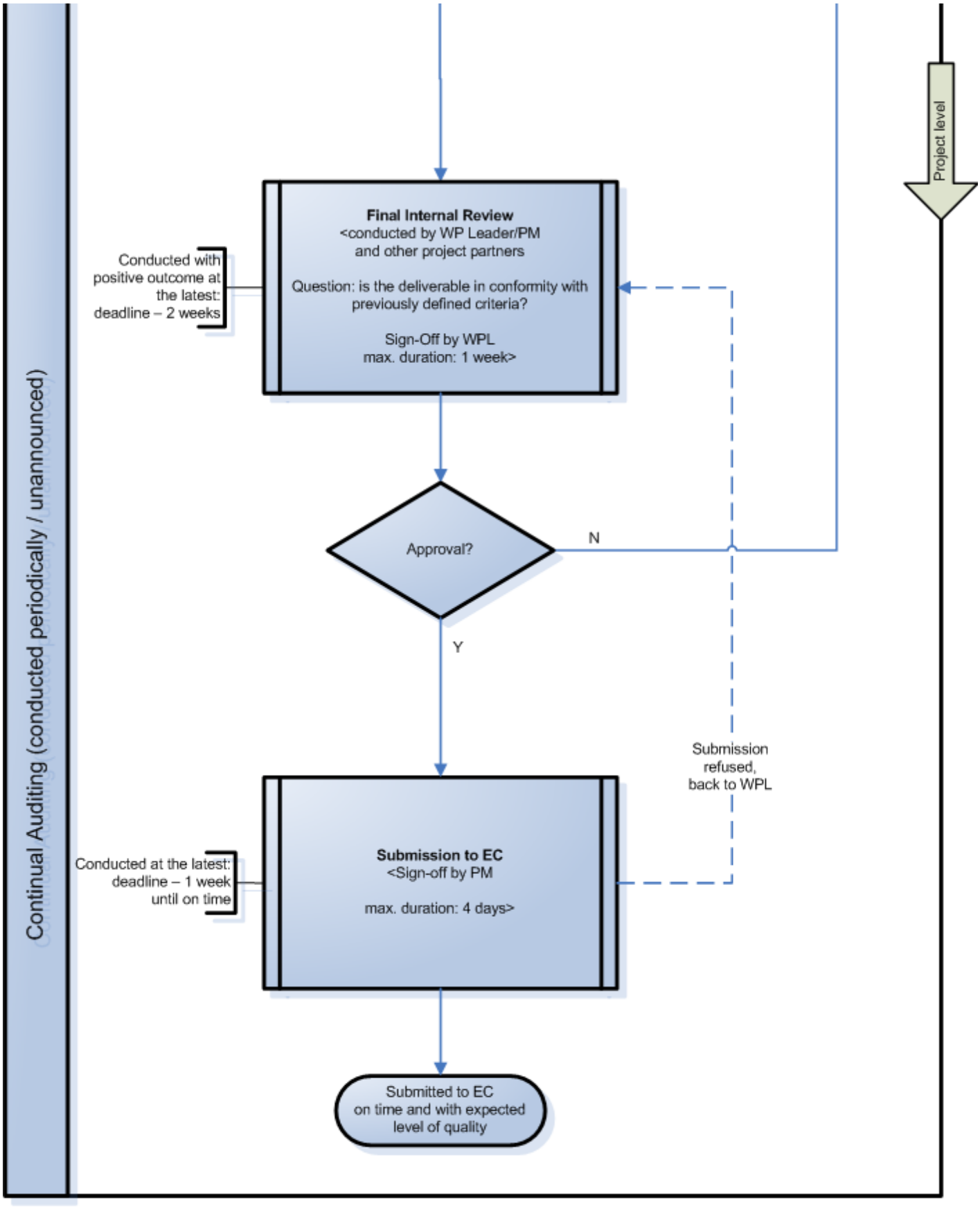

Figure 2 Quality Assurance Process for written deliverables (Project level)

4 FOR SOFTWARE DELIVERABLES

4.1 SOFTWARE QUALITY STRATEGY OUTLINE

4.1.1 Introduction

This chapter gives an overall description of testing activities. It is a guide for the Test Manager (see Roles and Responsibilities in 4.2.2.2) who finally decides how the test team will be organized, what tools and techniques to use, which types of tests to performance. He/she will document his/her decisions in the Test Plan (see template in Annex 8.1).

The method and principles regarding the testing process in the Access-eGov undertaking is governed by the Test Procedure described in this document (see 4.2.2). The procedure specifies actions to be taken, division of responsibilities for their performance as well as the deliverables. Moreover, the procedure includes document templates and specifications regarding the content of documents used in the testing process.

The Test Procedure is based on the following Standards: BS7925-1 and BS7925-2 (*British Standards*) as well as ANSI/IEEE 829/1983, pursuant to which the testing process is documented and carried out.

4.1.1.1 Software testing objective

Three objectives of testing the software deliverables can be distinguished:

- Assessing the level of risk related to the deployment of a deliverable here and now

The testing process provides information on the number and the significance of known yet not removed incidents.

- Detecting incidents (critical for the user and at the earliest stage possible)

The purpose of the tests is triggering faults in order to detect (and remove) incidents. Testing does not enhance the quality of the deliverable, but it is an assessment method with quality increasing through localising and removing the found incidents.

- Delivering data regarding the quality of the software development process (organization's goal)

During the project, the testing process allows for assessing the status of the project and deliverables, identification of increased risk points (elements of low quality requiring greater labour outlays) and thus - the implementation of remedies. Following the completion of the project, the testing process allows for improvement of procedures related to software development, thanks to the data delivered for analysis of incident reports statistics and their sources.

4.1.1.2 Basic terms

Incident – repeatable situation in which an element of the system does not work according to approved documentation or generally accepted standards.

Severity – a value indicating the seriousness of an incident, how it impacts the deliverable quality and what the consequences are:

- High – if the system crashes or is unusable (application will not function or system fails);
- Medium – if a workaround is available;

- Low – cosmetic problem with no impact on the functionality or usability of the process, but still it is not according to requirements/ design specifications.

Priority – a value indicating how important correction of the incident is and when it should be corrected:

- High – stop further testing, must be fixed as soon as possible;
- Medium – stop some tests, other tests can proceed; system is usable but incident must be fixed prior to next level of test;
- Low – possible to proceed, incident can be in principle left as it is - if necessary due to time or costs.

Regression testing – tests aimed at verifying whether corrections made did not change the functionality of the unmodified part of the system.

Re-tests – tests verifying if incidents reported in previous iterations of testing have been corrected.

Validation – control of the conformity of the deliverable – outcome of each stage of the software production process – with user requirements and needs.

Verification – control of the conformity of the deliverable – outcome of each stage of the software production process – with the assumptions/terms and conditions set at the beginning of the stage.

4.1.2 Testing principles

4.1.2.1 Tests organisation

The organization of test is a key to its effectiveness; therefore it has two special features:

- separation of the testing team from the team engaged in design, production and the selection of components – such a solution ensures impartiality and objectivism in the design and carrying out of test cases,
- documentation of the entire testing process, from its plan to incident reports.

Tests are carried out in many cycles comprising:

- original tests – performed on the basis of initial test cases.
- re-tests aim of which is to check whether incidents reported in previous iterations of testing have been corrected,
- regression testing - aiming at verifying whether corrections made did not change the functionality of the unmodified part of the system (these tests are optional, the Test Manager should decide whether to perform them or not).

The frequency of re-tests and regression testing depends mostly on the quality of deliverables delivered for testing.

The structure of the testing team, the list of tasks, responsibilities and the test schedule are specified in the Test Plan described below.

4.1.2.2 Documentation

Test process documentation in Access-eGov consists of:

- Test Plan,
- Specification of test cases,
- Incident Reports,

- Quality Metric (playing the role of the Test Report).

The Test Plan (see template in Annex 8.1) covers the specification of the scope, methodology, resources and test schedule, functionalities to be tested, actions to be performed as well as persons responsible for each action. This is the fundamental and the first document which specifies the way of organising the test process both in terms of methodology:

- selection of components to be tested,
- selection of types of tests,
- selection of tools and methods,

and in terms of organisation:

- the structure of the testing team,
- list of actions to be performed and the areas of responsibility,
- schedule of works.

The template of the Test Plan is included as an Annex to the document herein.

Specification of test cases constitutes a set of test cases, each containing a definition of:

- input data,
- actions – the method of performing a given test case and the manner of verifying/reading the results,
- expected outcome.

Test cases are built mainly based on Equivalence Partitioning and on the Boundary Value Analysis. Sequences are arranged, which correspond to the sequence of performing the tests, thanks to which performing one test case can serve at the same time for establishing the context for a subsequent test case.

In the specification of test cases, besides the input data, the expected outcome are of paramount importance, however their assessment being the most difficult task.

The sources of expected outcome most commonly used during the test process include:

- User requirements and specifications of such requirements – used e.g., in functionality, performance and security tests.
- Existing systems, dedicated applications developed by an independent team of programmers (parallel to the system developed) – used e.g. in cases of testing computing algorithms.
- Standards – standards related to i.a., designing of the user interface are used.

Requirements related to the construction of the test case have been described in detail in the appendix to this document.

Incident Reports describe nonconformities revealed in the course of performing test cases. A correctly formulated Incident Report:

- is as concise as possible – describes only the facts and details which are necessary for presenting and describing the incident, an Incident Report should contain a specification of the detailed sequence of steps that trigger an issue,
- describes one incident only,

- is simple and general – describes only the actions, which lead to generating an incident (disregarding other, immaterial actions),
- enables reproduction of the incident – the incident should be described in such a manner, so as to make possible repeating the incident through performing the described (and only those) actions.

Requirements related to developing the Incident Report are described in detail in Appendix to this document.

Incident Reports need to be stored in repositories. The form of the repository depends on the tools used – these can be paper documents (e.g. recorded in an Excel spreadsheet) or dedicated software (e.g. Mantis, Bugzilla). The repository should be built in such a way, so as to facilitate access to Incident Reports for all parties concerned (the reporting person, the programmer, Test Manager). It must allow for tracing the progress of work over the Incident Reports and filling up a Quality Metric (through an analysis of Incident Reports).

The Quality Metric is a document which sums up the outcome of tests. The Quality Metric is prepared at the end of a given testing cycle. In Access-eGov the Quality Metric for software deliverables plays the role of the Test Report.

The template of the Quality Metric constitutes an Appendix to the document herein.

4.1.2.3 Test tools and techniques

The techniques used in the testing process can be divided into the following groups:

- dynamic testing,
- static testing.

Dynamic testing is the process of evaluating a system or component based upon its behaviour during execution (applies exclusively to software deliverables). In practice, it often means searching for undesired side effects of the system's operation, e.g. memory leakage or loss of data.

Dynamic testing is based on the following techniques:

- Black-box testing (also referred to as functional testing)

The most often used of these techniques include: Equivalence Partitioning and Boundary Value Analysis. These methods are complementary and are used most often.

The equivalence class consists of a set of test cases that test the same or reveal the same incident. The Equivalence Partitioning method consists of limiting the unlimited set of test cases to a smaller set, at the same time ensuring the same (or close) effectiveness. This is a methodical process consisting of grouping similar input data, similar output data and similar operations of the program and the selection of one/a few representative(s) from each group. Each group may constitute a separate equivalence class.

The Boundary Value Analysis method also begins with selection of the equivalence classes. However, in this method, boundary conditions are specified for each equivalence class, testing the permitted value close to the boundary value, the boundary value and the forbidden value located as close to the boundary as possible.

- White-box testing (also referred to as structural testing or glass-box testing)

The degree of use of these methods decreases in the course of system development. They are the most useful at the stage of creating and testing components (unit testing).

For the purpose of this undertaking, instead of the term "dynamic testing", the term "testing" will be used. The method and the principles of carrying out these tests are governed by the Test Procedure described herein (see 4.2.2).

Static testing covers all types of analyses, both of the source code (in case of software components) and documentation. It is conducted by a way of inspections and reviews. For the purpose of this undertaking, instead of the term "static testing", the term "review" will be used. The method and the principles of carrying out code reviews are governed by the Code Review Procedure described herein (see 4.2.3).

4.1.3 Types of tests

Software testing is done in several stages, depending on the current stage of software development. These stages are shown in Fig. 3.

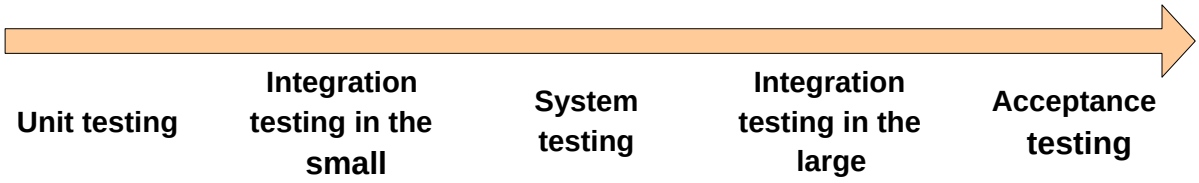

Figure 3 Stages of the testing process

This chapter discusses each of the stages presented above.

4.1.3.1 Unit testing

Unit testing (also referred to as module testing or basic testing) is the first type of tests performed in the testing process. At this stage, the unit is tested in separation from all the other units, disregarding its relations with other system modules. In case of software units, these tests are in most cases performed by programmers (e.g. using white-box testing method – with the help of debugging programs).

4.1.3.2 Integration testing in the small

The integration testing in the small covers tests of communication of the system units with other units of the same system.

For the purpose of further definitions, let us assume the following structure of the system:

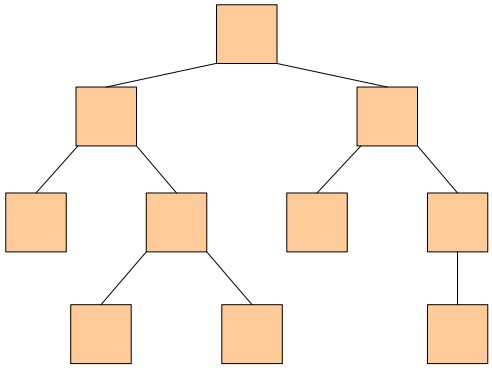

Figure 4 System integration – an example

In case of integration testing in the small, the following strategies are adopted:

- Bottom-up integration

The bottom-up integration diagram looks like as follows:

1. H + E,
2. H + E + I,
3. H + E + I + B,
4. etc.

- Top-Down integration

The top-down integration diagram looks like as follows:

1. A + B,
2. A + B + C,
3. A + B + C + D,
4. etc.

4.1.3.3 System testing

System testing is the first opportunity for checking the system as a whole – communication between the units and the operation of the system as a whole.

Functional system testing

The objective of functional system testing is to verify whether the system achieves the objectives and performs all the defined functionalities.

Functional system testing is based on:

- system requirements

Each functional requirement is covered by one or more test case(s).

- business processes

Test cases are based on the projected user profiles and the system use cases. They are the result of answers to the following questions:

- Who is the system's user?
- What is expected from the system?
- Which elements of the system are the most critical for the user?

Non-Functional system testing

The non-functional system testing of the system covers the attributes of the system's quality, which include, among others: performance, security and usability.

Performance testing

These tests are particularly important in dispersed systems processing significant amounts of data and in which a considerable obstacle in achieving system performance that would satisfy the user is the time required for data transmission as well as the database access time. All these factors contribute to the ease of work with the system.

Security testing

The security testing relates to all the aspects connected with the system's security, which is:

- user authentication process,
- administration of users and user rights to perform certain operations in the system,

- data transmission security,
- data storage security.

Usability testing

Test cases are built in such a manner, so as to give answers to the following questions:

- Does the user interface comply with the standards?
- Is the data input sequence logical?
- Are the default values reasonable?
- Are the system messages clear, transparent and understandable?

4.1.3.4 Integration testing in the large

Integration testing in the large covers tests of communication of the system with other computer systems.

4.1.3.5 Acceptance testing

Acceptance testing constitutes the basis for delivery of the system acceptance. Its main objective is providing the user with evidence that the system correctly performs the defined functions.

Acceptance testing is not in the scope of this Project Quality Plan.

4.2 SOFTWARE EVALUATION

4.2.1 Approval Procedure for software deliverables

4.2.1.1 Purpose, Scope and Objectives

The purpose of the Approval Procedure for software deliverables is to ensure that given deliverable meets quality measures defined in the Project Quality Plan (see table in 4.3). When the software deliverable is approved, it can be used as a stable reference for building the rest of the system.

The Approval Procedure for the deliverable should start as soon as possible – mostly (D6.1 is the only one exception where development and acceptance activities constitute separate tasks) it means when the deliverable development starts. Waiting for its submission for acceptance the Test Manager (see Roles and Responsibilities in 4.2.2.2) should organize the test process and prepare everything what is needed to execute tests.

4.2.1.2 Roles and Responsibilities

Mostly the Task Leader is responsible for the deliverable acceptance. There is only one exception – preliminary version of D6.1 which is performed by TUK is tested by EMA – see details in table in 4.3.

Therefore when the term “Task Leader” is used (in the context of the Approval Procedure) it should be understood as a Task Leader responsible for the task which contains in its scope deliverable acceptance.

4.2.1.3 Activities

NO.	Activity	Responsibility
1	**Fill up the Quality Metric for the deliverable.** Task Leader checks in the Project Quality Plan (see table in 4.3) which evaluation method to use for the given deliverable. There are two possibilities: test or code review (see table in 4.3). The method and principles regarding the testing process in the Access-eGov undertaking is governed by the Test Procedure described in 4.2.2. The Code Review Procedure is described in 4.2.3. Task Leader prepares the Quality Metric for the deliverable (using the Quality Metric template - see Annex 8.4) and fills up its section 1. Deliverable and 2. Plan. **Duration:** max 0.25 day.	**Task Leader**
2	**Nominate the person responsible for the deliverable evaluation.** If the software deliverable will be tested – Task Leader nominates the Test Manager from his institution. If the software deliverable will be reviewed – Task Leader nominates (from his institution) the person responsible for the code review. **Duration:** max 0.25 day.	**Task Leader**
3	**Follow up the adequate procedure.** Test Manager follows up the Test Procedure described in 4.2.2. The person responsible for the code review follows up the Code Review Procedure described in the 4.2.3. As a result of these procedures – Quality Metric is completed and the deliverable is approved or directed to rework. **Duration:** Test Procedure – max. 12 days (see 4.2.2). Code Review Procedure – max 4 days (see 4.2.3).	**Test Manager or the person nominated for the code review**

4.2.1.4 Outcomes

1. Start-up of the adequate evaluation procedure (Test Procedure or Code Review Procedure).
2. Quality Metric for the deliverable initially filled.

4.2.2 Software Test Procedure

4.2.2.1 Purpose, Scope and Objectives

The procedure specifies all the actions to be performed, in order to perform properly one cycle of testing, starting with its planning, through appointing a testing team, preparing the specification of test cases, preparing for performance of tests, through to preparing the Test Report (Quality Metric).

Testing is a method of assessing deliverable (software) quality and it enables verification of fulfilment of the quality measures for each deliverable as specified in this Project Quality Plan (see table in 4.3).

The Test Procedure pertains to software deliverables and it governs the actions related to dynamic testing (the process of evaluating a system or a component based on its behaviour in operation).

4.2.2.2 Roles and Responsibilities

A Role defines a set of related responsibilities of an individual or individuals. One person can perform one or more roles in the testing process.

NO.	Role	Responsibility
1	Test Manager	• Test Planning • Organizing and controlling of the test process (people and tools) • Preparing the Quality Metric for the deliverable (see template in annex 8.4) Test Manager is a person who was not involved in the creation of the tested deliverable.
2	Test Designer	• Defining test cases Test Designer is a person who was not involved in the creation of the tested deliverable.
3	Tester	• Preparing tools and environments • Preparing test data • Executing test cases • Reporting test results (in the form of Incident Reports – see annex 8.3) Tester is a person who was not involved in the creation of the tested deliverable.

As written in 4.2.1 Task Leader nominates the Test Manager. In turn, Test Manager nominates Test Designers and Testers (from his institution).

In particular Test Manager, Test Designer and Tester can be one person.

4.2.2.3 Activities

NO.	Activity	Responsibility
1	**Prepare an overall vision of testing and describe it in detail in the Test Plan.** Test Manager prepares the Test Plan using the Test Plan template (see annex 8.1). Test Plan must correspond to the Project Quality Plan and Quality Metric for the deliverable (initially filled by the Task Leader, see Approval Procedure for software deliverables in 4.2.1).	**Test Manager**

NO.	Activity	Responsibility
	While re-testing and regression testing – if applicable (see definitions in 4.1.1.2), existing Test Plan (defined earlier, in the previous testing cycle) should be verified and adapted to new version of tested deliverables. **Duration:** max 1 day.	
2	**Appoint the test team.** Test Manager appoints the test team (nominates Test Designers and Testers from his institution). Each member of the test team should know the domain of tested deliverables and tools used in the test process (documents templates, standards, procedures, software tools – e.g. repositories, etc.). **Duration:** max 0.5 day.	**Test Manager**
3	**Define test cases.** Test Designer writes test cases and indicates for each of them its predecessors and consequents, if applicable. Test cases must correspond to the Test Plan (which indicates deliverables and features to be tested, testing approach, techniques and tools, etc.). While re-testing and regression testing – if applicable (see definitions in 4.1.1.2), existing test cases (defined earlier, in the previous testing cycle) should be verified and adapted to new version of tested deliverables. Specification of each test case must comply with the specification of test case content (see annex 8.2). **Duration:** max 4 days.	**Test Designer**
4	**Verify test coverage.** Test Designer checks if defined set of test cases guarantee achievement of test coverage defined in the section 8.3 (functional coverage – estimated as proportion of tested functions, use cases, etc. or requirements' coverage – estimated as proportion of tested requirements). If not – the set of test cases should be completed. **Duration:** max 0.25 day.	**Test Designer**
5	**Prepare tools.** Tester prepares, in particular – installs and configures, tools (e.g. Incident Reports repository, etc.; test automation tools – if applicable) needed to support the test. If test-running tools are used, Tester writes scripts for defined test cases which execution will be automated. **Duration:** max 1 day (or 2.5 days if scripts need to be prepared).	**Tester**
6	**Prepare environments.** Tester prepares (basing on the Test Plan) test environments (hardware, system and communication software, etc.).	**Tester**

NO.	Activity	Responsibility
	Duration: max 1 day.	
7	**Install and configure deliverable to be tested.** Tester installs and configures deliverables to be tested. **Duration:** max 0.5 day.	**Tester**
8	**Prepare test data.** Tester prepares test data needed for test cases (especially for load testing, performance testing, etc. when much data of specified characteristic is needed). **Duration:** max 1 day.	**Tester**
9	**Execute test cases.** Tester performs (manually or using test automation tools) defined test cases. For each test case notes actual outcome and compares it with the expected outcome. If the outcome differs from the expected one, Tester tries to collect complete information about the incident found and fills up an Incident Report (see annex 8.3). **Duration:** max 4 days.	**Tester**
10	**Analyze test results (in particular – Incident Reports). Fill up the Quality Metric for the tested deliverable.** Test Manager fills up the Quality Metric (see template in annex 8.4) for the tested deliverable. Basing on the outcome of tests he approves the deliverable or assigns it status "Rework is needed". **Duration:** max 0.5 day.	**Test Manager**

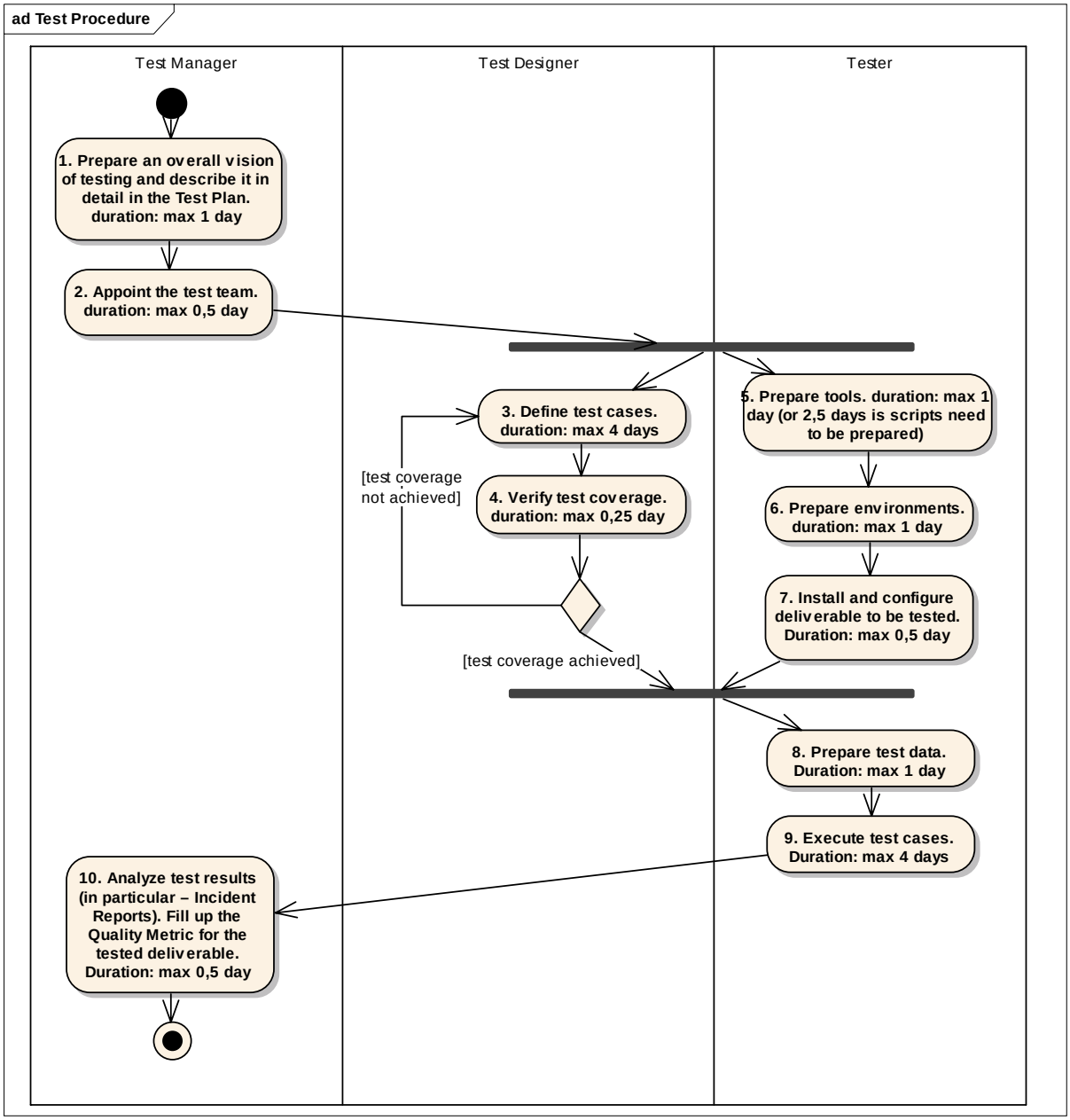

Figure 5 Test Procedure

4.2.2.4 Outcomes

- Test Plan (see template in Annex 8.1)
- Test cases (see Annex 8.2)
- Incident Reports (see Annex 8.3)
- Quality Metric for the tested deliverable (see template in Annex 8.4)

4.2.2.5 Test Procedure checklist

1. Test Plan exists and it is based on the Test Plan template (see template in Annex 8.1).
2. Test Plan corresponds to the Project Quality Plan for tested deliverables.
3. None of the test team members was involved in the creation of the tested deliverables.
4. Each member of the test team knows his role and his tasks in the test process.

5. Each member of the test team knows the domain of tested deliverables.
6. Each member of the test team knows tools used to support the test process (documents templates, standards, procedures, software tools, etc.).
7. Test cases are defined and grouped into test sequences.
8. Test cases specifications correspond to the Test Plan (testing approach, techniques, tools, etc.).
9. Specification of each test case complies with the specification of the test case content (see Annex 8.2).
10. Test coverage is estimated and known.
11. Defined set of test cases guarantee achievement of test coverage.
12. All tools needed to support the test process are identified and listed in the Test Plan.
13. All tools needed to support the test process are prepared before executing the test.
14. Test environments are identified and specified in the Test Plan.
15. Test environments are prepared before executing the test.
16. All items and features to be tested are identified and specified in the Test Plan.
17. All deliverables to be tested are installed and configured before executing the test.
18. Test data needed for test cases is specified.
19. Test data is prepared before executing the test.
20. Errors found are documented using Incident Report.
21. Each Incident Report complies with the specification of the Incident Report content (see Annex 8.3).
22. Quality Metric for the software deliverable is prepared and it is based on the Quality Metric template (see template in Annex 8.4).

4.2.3 Code Review Procedure

The purpose of the Code Review is analyzing the source code in order to detect all nonconformities with defined guidelines for developers.

Code Review should be done by a person (one at least) who was not involved in the creation of the reviewed software deliverable. This person is nominated by the Task Leader responsible for the deliverable acceptance (see 4.2.1).

Code Review Procedure composes of two activities:

1. Review the source code and compare it with Developers' technical guidelines. (duration: max 3.5 days)
2. Document the outcome of the code review in the Quality Metric for the deliverable (see template in Annex 8.4). Write in all detected nonconformities. Approve the software deliverable or direct it to rework. (duration: max 0.5 day)

4.3 SOFTWARE QUALITY ASSURANCE PRACTICE IN ACCESS-EGOV

The table below presents all software deliverables and for each of them:

- participant responsible for the deliverable performance or change,
- participant responsible for the deliverable acceptance (mostly the same as above),
- the name of the task which contains in its scope deliverable performance or change,
- the name of the task which contains in its scope deliverable acceptance (mostly the same as above),

- evaluation methods,
- quality measures,
- deadline for deliverable acceptance (when the deliverable must be accepted, at last).

DELIVERABLE				ACCEPTANCE				
Identifier	**Name**	**Task which contains in its scope deliverable performance or change**	**Task Leader**	**Evaluation methods**	**Quality measure**	**Task which contains in its scope deliverable acceptance**	**Responsible**	**Deadline for acceptance**
D4.2, preliminary version	Components for 'management of egovernment service mark-up'	T4.3, Implementation of the mark-up platform components and their internal testing	TUK	Black box testing at the stage of: ▪ Functional testing (using simulated real-life scenarios) ▪ stress testing, ▪ and other types of tests (e.g. security, usability, performance) covering types of defined requirements)	100% test coverage achieved AND all test cases executed AND no open Incident Reports of high severity	T4.3, Implementation of the mark-up platform components and their internal testing	TUK	M20
D4.2, final version	Components for 'management of egovernment service mark-up'	T4.4, Implementation of requirements on mark-up platform components arising from the trial 1 evaluation	UR	Black-box testing (re-tests of the preliminary version)	100% test coverage achieved AND all test cases executed AND no open Incident Reports of high severity.	T4.4, Implementation of requirements on mark-up platform components arising from the trial 1 evaluation	UR	M27
D5.2, preliminary version	Components for 'Personal Assistant'	T5.2, Implementation of the personal assistant platform components and their internal testing	EMA	Black-box testing at the stage of: ▪ Functional testing, ▪ and other types of tests (security, usability, performance, etc.) covering types of requirements (if applicable)	100% test coverage achieved AND all test cases executed AND no open Incident Reports of high severity.	T5.2, Implementation of the personal assistant platform components and their internal testing	EMA	M20
D5.2, final version	Components for 'Personal Assistant'	T5.4, Implementation of requirements on personal assistant	EMA	Black-box testing (re-tests of the preliminary version)	100% test coverage achieved AND all test cases executed AND no open	T5.4, Implementation of requirements on personal assistant	EMA	M27

DELIVERABLE				ACCEPTANCE				
Identifier	**Name**	**Task which contains in its scope deliverable performance or change**	**Task Leader**	**Evaluation methods**	**Quality measure**	**Task which contains in its scope deliverable acceptance**	**Responsible**	**Deadline for acceptance**
		platform components arising from the trial 1 evaluation			Incident Reports of high severity.	platform components arising from the trial 1 evaluation		
D6.1, preliminary version	Access-eGov platform	T6.1, Integration of the basic components	TUK	Code Review	Conformity with Developers' technical guidelines.	T6.1, Integration of the basic components	TUK	M30
				Black-box testing at the stage of: ▪ Functional testing, ▪ and other types of tests (security, usability, performance, etc.) covering types of requirements (if applicable)	100% test coverage achieved AND all test cases executed AND no open Incident Reports of high severity.	T6.2, Internal testing of integrated components and implementation of requirements arising from the evaluation of the trial 2	EMA	M36
D6.1, final version	Access-eGov platform	T6.2, Internal testing of integrated components and implementation of requirements arising from the evaluation of the trial 2	TUK	Black-box testing (re-tests of the preliminary version)	100% test coverage achieved AND all test cases executed AND no open Incident Reports of high severity.	T6.2, Internal testing of integrated components and implementation of requirements arising from the evaluation of the trial 2	EMA	M36

5 CHANGE MANAGEMENT

Nowadays, flexibility in IT projects is a must and changes are often considered beneficial. Changes can be manifold; however the scope of change management in IT projects is limited to those types of change which directly affect the quality of the project's output (e.g. requirement changes, configuration changes; but not e.g. cuts in the project's budget). Since external factors can have a major impact (e.g. when organizational changes lead to new requirements) change management must be 'reactive', in which project management is responding to changes in the macro-environment, as well as proactive, in which project management is initiating the change in order to achieve a desired goal (that is, the source of the change is internal).

In order to enable this kind of flexibility, the Access-eGov project team plans for changes and follows through a process of constant communication and negotiation. This process is supported by a clear-cut strategy (see Fig. 6 below, adopted from [Schwalbe, 2004]) and a number of artefacts such as specialized software (as provided in an integrated development environment) and simple reporting templates (see templates below).

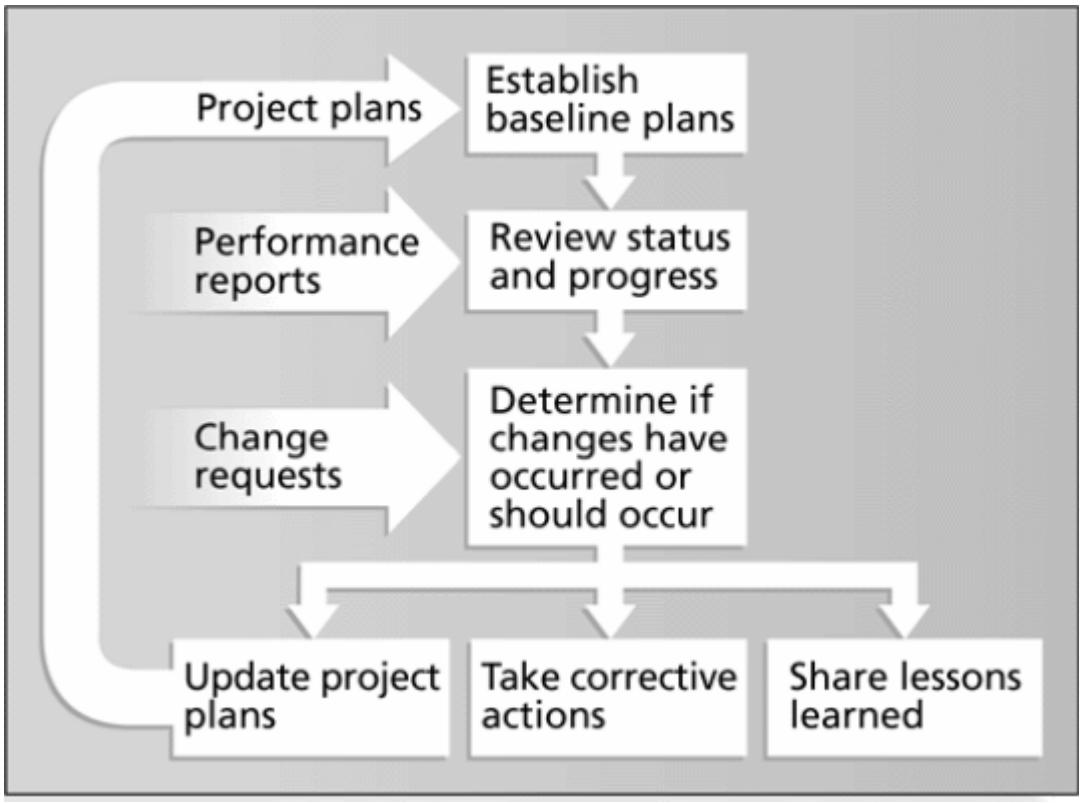

Figure 6 Integrated Change Control Process

([Schwalbe, 2004] p. 123)

Change management within an information technology project involves identifying, evaluating, and managing changes throughout the project life cycle. Therefore, the three main tasks of change control within Access-eGov are:

1. Influence the factors that create changes to ensure they are beneficial
2. Determine that a change has occurred

3. Manage actual changes when and as they occur

Change management related to an information technology project can have an environment-oriented and/or a product-oriented focus. Since both of these do require different approaches and participation of different actors, the change management procedures implementing the above tasks within Access-eGov are discussed separately in the following subsections.

5.1 ENVIRONMENT-ORIENTED CHANGE MANAGEMENT

The (future) users of an IT system and their frame of action (the organization they work for, the laws they have to obey etc.) we call the environment of the system to be developed. Since environmental factors can have a major impact on development (user requirements) and use (e.g. acceptance) environment-oriented change management is especially important for information system projects. This kind of change management is closely related to the social science primarily dealing with the human aspect of change. However, in practice, effective change management is multi-disciplinary, touching all aspects of the organization, mostly triggered by implementation of new procedures and/or technologies. Change management aligns people, processes and IT during the transition from the current to the desired state. This requires a new vision that takes the organization in a direction it would otherwise not take. This vision usually implies a variety of assumptions regarding the change or stability of organizational and environmental conditions.

Within Access-eGov as an IT project, environment-oriented change management mainly has to focus on these assumptions regarding the change or stability of organizational and environmental conditions such as e.g. change of procedures, business processes, or training requirements. The three main tasks of change control are to be implemented as follows:

Ad 1) Influence the factors that create changes to ensure they are beneficial
As with any business transformation, success is based on a careful balance between the amount of change and the rate at which the change is introduced to the organization. For establishing a baseline plan, the initial requirement analysis should already cover the aspects of environmental change (e.g. scenarios for new procedures, change of organizational responsibility, foreseen changes in laws etc.). However, any change might encounter resistance, therefore a comprehensive change management strategy is needed including:

1. Creating consensus in the top team to the change vision, goals and strategies.
2. Cascading the change vision, goals and strategies down to leaders at every level.
3. Creating strategic alignment of all current and future projects that ensures a clear line of sight between the project and the change vision and goals.

It is mainly the responsibility of the project's user partners to ensure this kind of change management related to the project's visions of future IT usage, and the PCC must follow up whether this kind of change management is implemented successfully and decide on how to deal with major problems and obstacles in this area.

Ad 2) Determine that a change has occurred
Access-eGov is a three year project, and as the project moves on changes and/or additional assumptions will occur (e.g. due to a political reform a user partner faces a change of responsibility of certain administrative services). The status of these changes and assumptions has to be documented since they are usually critical success factors for the use of the new IT system. It is mainly the responsibility of the project's user partners and in particular the liaison officers (LOPA) to ensure that any change related to the project's visions of future IT usage is detected, reported and put on the agenda for scrutinizing the impact of these changes.

Ad 3) Manage actual changes when and as they occur

In any case, change reports and requests must be filed using a standardized form. The template for reporting/requesting environment-oriented changes is included in Appendix 8.6. The procedure to be followed in Access-eGov is as follows:

1. Any member of the project team may fill out the details of the form to report on an environmental change that already happened, will happen or is requested to happen.
2. The filled form will be passed on electronically to the concerned liaison officer(s) (LOPA) and to the leader of WP 2 (requirement analysis) in order to evaluate the impact and (in case of request) the feasibility of the change.
3. (a) If the change is minor, the concerned liaison officer(s) and WP 2 leader coordinate with the WPL and/or TL who filed the request or whose WP/Task is affected in order to discuss the details and to find and agree on a solution how to accommodate the environmental change (request).
   (b) If the change is major (i.e. affecting the overall project goals), the concerned liaison officer(s) and/or the WP 2 leader raise the issue to the PCC for discussion and decision. The concerned liaison officer(s) and/or the WP 2 leader coordinate with the WPL and/or TL who filed the request or whose WP/Task is affected in order to document the details and to suggest options (i.e. possible solutions) how to accommodate the environmental change (request).
4. For each form submitted the concerned liaison officer(s) and the WP 2 leader state how the environmental change (request) will be dealt with within the Access-eGov project (in case of (a) referring to the consensus reached, in case of (b) referring to the PCC decision). The submitted form and the solution statement are published in the internal project web.
5. If the solution is not implemented by the time of step 4 is finished, the WP 2 leader is responsible for following up the issue until it is finalized or a new case is filed covering the same topic.

5.2 PRODUCT-ORIENTED CHANGE MANAGEMENT

Within the core of an IT project, change management refers to approved deliverables, i.e. implementations. The main purpose is to establish and maintain the integrity of the products of the systems development project throughout the project's software life cycle, thus avoiding any confusion during development and ensuring reach of project goals. The most developed approach in IT project management is configuration management which

- ensures that the products and their descriptions are correct and complete
- concentrates on the management of technology by identifying and controlling the functional and physical design characteristics of products

According to the Software Capability Maturity Model (SW-CMM) from Carnegie Mellon University's Software Engineering Institute (SEI), software configuration (change) management involves identifying the configuration of the software (i.e., selected software work 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 life cycle. Configuration management specialists

identify and document configuration requirements, control changes, record and report changes, and audit the products to verify conformance to requirements.

6 AUDITING

6.1 PURPOSE, SCOPE AND OBJECTIVES

According to [Kenzner, 1994] a quality audit is an independent evaluation performed by quality personnel that ensures that the project is conforming to the previously agreed project quality requirements and is following the established quality procedures and policies. The Access-eGov quality audit is a continuous process which will guarantee that quality assurance procedures are used in the proper manner. The description of these quality procedures is part of this document.

Quality auditing for the Access-eGov project will be carried out by the QAM. If conflict of interest should arise (e.g. a WP is lead by project partner organization EMAX and QAM is an EMAX employee also), a member of SDC will be appointed by the PM for quality auditing. This will ensure that Acces-eGov quality audit is an independent evaluation process.

The objective of a quality audit is to identify "lessons learned" that can improve performance of this project or of other projects within the performing organization [PMI, 1996].

The main point for the Access-eGov project is to ensure that the quality assurance procedures are performed on time and are used properly (e.g. checking the sign-off table etc.). In case some non-compliance is found, the challenge is to generate some piece of knowledge ('identify lessons') which can be (re-)used to avoid the same kind of problem in the future. The checklist closing this chapter is intended to help generate a piece of reusable knowledge that will be stored in the internal section of the Access-eGov website.

6.2 ROLES AND RESPONSIBILITIES

NO.	Role	Responsibility
1	QAM	• performing audit activities
2	PM	• appointing person (no QAM) in case of possible conflict of interest • plan the follow-up if needed

6.3 PRECONDITIONS

- An internal quality assurance procedure has started

6.4 ACTIVITIES

Audit

Since the Access-eGov audit is a monitoring process, the responsibility of the QAM is to track activities which have to be executed according to the existing quality assurance procedures. He/she is an active participant in this process. If the deliverable is not produced by the defined activity deadline, he/she will contact the involved partners. If the deadline can

not be reached, the new deadline will be negotiated between the QAM and the appropriate WPL.

If some non-compliance can threaten the deliverable submission, the QAM will contact the PM (e.g. by e-mail). The PM will plan the follow-up.

Documentation

All non-compliances related to one deliverable are collected and described in details in the document (for template see Appendix 8.5 by the QAM. The questions in the auditing checklist (see the section below) should help with this description.

If every step of the quality assurance procedure related to one deliverable runs properly, the QAM will also fill in the same template (with the formal information about the deliverable).

The document will be produced for each deliverable and should be stored in the internal section of the project's web site by 2 weeks after the deliverable was finished (linked to the Task 1.4).

6.5 AUDITING CHECKLIST

The following questions are taken from [Trevor, 1996].

For any non-compliance, ask:

1. Why did this happen?
2. What were the consequences?
3. Could the situation have been anticipated?
4. Were there early signals that went unrecognised at the time?
5. When was the situation first identified?
6. Who should have reacted?
7. Was it due to unclear responsibility or authority?
8. Was it due to poor estimating?
9. Was it due to poor planning?
10. Has there been a failure of any partners to fulfil their obligations?
11. Has there been a communication failure?
12. Where have our communication processes failed?
13. Can we correct these failures now?
14. Has the control system failed?
15. Is the event a direct result of current organisational policies?
16. What have we learned from the non-compliance(s)?
17. What changes can we implement now to prevent a recurrence?
18. Are policy changes necessary?
19. How can we check the learning points (the lesson) are communicated now?

7 CONCLUSION

This Project Quality Plan outlined all internal quality assurance procedures that are applicable within Access-eGov.

They comprise:

- **Quality assurance for written deliverables**
  This chapter describes a four-week process of internal reviews conducted on Task level (called Peer Reviews) and on Project level (called Final Internal Review) until the deliverable's deadline has been reached. These internal review processes accompany the entire life-cycle of any written document from creation up to submission to EC. The completed Quality Assurance stages have to be signed off by Access-eGov project partners inside the respective document's sign-off table.

  These procedures are relevant to any project partner involved in creating written deliverables.

- **Quality assurance for software and related deliverables**
  In contrast to Quality Assurance for written deliverables, software and related products will not only have to be reviewed, but also tested. The relevant quality procedures are transparent for Access-eGov partners and constitute a monolithic Yes/No-process on Project level. Detailed testing procedures, based on international standards and conventions, are also outlined in this chapter. They are addressed to the actual Testing personnel that has to be appointed on Task level only.

- **Change Management**
  This chapter deals with managing changes in the IT and administrative environment of the Public Administration partners in Access-eGov. For that purpose, changes to the organizational or procedural environment need to be either reported (when they occurred) or requested (if necessary) using a standardized form.

- **Auditing**
  In order to control that daily quality assurance procedures conform to the guidelines as set out in this Project Quality Plan, continual Auditing will be carried out by independent staff. The Project's Quality Assurance manager is charged with conducting this continual internal evaluation process. Non-compliances are recorded and detailed follow-up strategies will be worked out. An Audit report will be generated for each deliverable.

The Appendix provides a set of necessary form templates to facilitate conducting all Quality Assurance processes in daily work. They are considered ready-to-use document templates and can instantly be used out of the Project Quality Plan without major changes.

This document is the first deliverable created under the terms of these Quality Assurance procedures. All future deliverables produced within Access-eGov will have to follow these guidelines.

8 APPENDIX

Usage Instruction

The form document templates presented in this Appendix are ready for instant use and intended for internal project work only. Thus they only compose of a light header section.

In order to generate a new document based on one of these templates, the author is supposed to insert the template (as it can be found in the Appendix) into an empty word processor document. The new document shall be given the same name and title as used in this Appendix.

8.1 TEST PLAN TEMPLATE

The Test Plan covers the specification of the scope, methodology, resources and test schedule, functionalities to be tested, actions to be performed as well as persons responsible for each action.

The Test Plan template is based on ANSI/ IEEE 829-1998 standard for software test documentation and can be found on the following three pages:

Test Plan

1 INTRODUCTION

[Summarize the purpose of the Test Plan. Identify the scope of the Test Plan (in relation to the Project Quality Plan). Include references to other project plans, e.g. Project Quality Plan or documents relevant to this test process, e.g. standards. Summarize the software items and features to be tested.]

2 TEST ITEMS

[Identify the test items you intend to test, including their version/ revision. Include references to existing Incident Reports and test item documentation, e.g. requirements specifications, design specifications, users guides, installation procedures, etc. Identify any critical steps required before testing can begin as well, such as how to obtain the required item.

Indicate items that will be excluded from testing.]

3 FEATURES TO BE TESTED

[Identify all software features and combinations of features (from the users point of view) that will be tested. Identify the test cases associated with each feature or combination of features.]

4 FEATURES NOT TO BE TESTED

[Identify all software features and combinations of features that will be excluded from testing. Indicate the reasons (why they will not be tested), e.g. not to be included in this release of software; low risk – has been used before and is considered stable, etc.]

5 APPROACH

[Describe the overall approach to testing, describe overall rules and processes. Identify groups of features and specify the chosen test approach that will ensure adequate testing for each group. Specify techniques (e.g. techniques to be used to trace requirements, testing techniques) and tools (e.g. repositories, test automation tools) to be used. Define significant constraints such as resource availability or deadlines. Specify regression testing rules (e.g.

how much will be done – if any, when will be done). Specify special requirements for the testing, if applicable.]

6 ITEM PASS/ FAIL CRITERIA

[Re-write quality measures for the deliverable from the Quality Metric for the deliverable – they should be the same as listed in the Project Quality Plan.]

7 SUPSENSION CRITERIA AND RESUMPTION REQUIREMENTS

[Specify the criteria used to suspend all testing or parts of testing activity described in this Test Plan (e.g. if the number or type of incidents reaches a point where there is no sense in continue the test). Specify testing activities that must be repeated when testing is resumed (e.g. install and configure software deliverables to be tested, execute some test cases, etc.).]

8 TEST DELIVERABLES

[Identify the test documents, e.g.:

1. *Test Plan*
2. *Specification of test cases*
3. *Incident Reports*
4. *Test Report*

Identify test data and test tools prepared to use in the test process.]

9 ENVIRONMENTAL NEEDS

[Specify environmental needs including:

1. *hardware,*
2. *external devices (printers, scanners, modems, UPS, etc.),*
3. *system software and drivers,*
4. *communication requirements/ hardware/ software (network topology, devices such as bridges/ routers, etc.),*
5. *required level of security,*
6. *test tools,*
7. *any other needs (e.g. office space, phones, etc., if applicable).]*

10 TESTING TASKS

[Identify all testing tasks including test planning, test specification and test execution. Identify all inter-tasks dependencies and required skill levels. You can use the Test Procedure (described in the Project Quality Plan) as a guide - testing tasks are listed therein.

Notice – this section could be omit if the schedule (see the section 13) ensures this information.]

11 RESPONSIBILITIES

[Assign a responsible person for each test task identified (in other words - assign a person for each role, roles in the test process are described in the Project Quality Plan – see the Test Procedure therein).

Notice – this section could be omit if the schedule (see the section 13) ensures this information.]

12 SCHEDULE

[Specify the schedule for each testing task and test milestones including dependencies between tasks defined in the Testing Tasks section. Estimate the time required for each task (see the Test Procedure in the Project Quality Plan – each testing activity has assigned a max duration there). Specify period of use for each testing resource.

Test schedule should be based on the project schedule.]

13 APPROVALS

[Specify the names and titles of all persons who must approve this plan.]

8.2 SPECIFICATION OF TEST CASES

The specification of test case's contents is based on ANSI/ IEEE 829-1998 standard for software test documentation.

Each test case must include at least:

1. **Unique identifier**

Permanent identifier used to identify every test case.

2. **The purpose of the test case**

Reference to requirement/ requirements or feature/ set of features to be tested by this test case.

3. **Name/ title**

Short test case title.

4. **Preconditions**

Description of system state prior to the execution of test case.

5. **Input specification**

Specification of all inputs required to execute the test case. It could be definition of data (values, ranges, sets), files or relationships (e.g. timing).

6. **Action**

Detailed information on how to perform the test case.

7. **Expected result (outcome)**

Specification of expected outcome, e.g. outputs (displayed or changed GUI objects, sent messages or signals, printouts, sounds, movements), state transitions, data changes, long-time results (e.g. system still works correctly after a week), quality attributes (time, size, etc.), side-effects.

Defined test cases should be grouped into test sequences.

8.3 INCIDENT REPORTS

The specification of Incident Report's contents is based on ANSI/ IEEE 82901998 standard for software documentation.

Each Incident Report must include at least:

1. **Unique identifier**

Permanent identifier used to identify every Incident Report.

2. **Test case identifier and revision**

Test case that will provide the information to repeat the incident. If the incident was not discovered as a result of any test case, provide information such as: input specification, action to perform, expected and actual results, anomalies.

3. **Software under test identifier and revision**

Identifier/ name/ title of software deliverable that was tested.

4. **Test environment identifier and revision**

Specification of the test environment (it should correspond to the Test Plan).

5. **Name of the Tester**

Tester reporting the incident – the author of the Incident Report.

6. **Expected and actual results**

If the incident was discovered as a result of the test case, expected results can be omit (test case specification ensures this information). Otherwise specify expected and actual results.

7. **Severity**

Severity indicates the importance to users and potential impact to the system.

- High – if the system crashes or is unusable (application will not function or system fails)
- Medium – if a workaround is available
- Low – cosmetic problem which does not impact the functionality or usability of the process but is not according to requirements/ design specifications

8. **Priority**

Priority indicates the urgency to fix the incident (the order in which the incidents are to be addressed).

- High – stops further testing, must be fixed as soon as possible
- Medium – stops some tests, other tests can proceed; system is usable but incident must be fixed prior to next level of test
- Low – possible to proceed, incident can be left in if necessary due to time or costs

8.4 QUALITY METRICS TEMPLATE

The Quality Metrics template can be found on the following text page.

FP6-2004-27020

Access-eGov

WP1: Project management

Task T1.4: Project evaluation

Quality Metrics

1. Deliverable

1.1. Identifier	*[Deliverable identifier, e.g. "D4.2" for Components for 'management of egovernment service mark-up']*
1.2. Name	*[Deliverable name, e.g. "Components for 'management of egovernment service mark-up'"]*
1.3. Workpackage	*[Workpackage identifier, e.g. "WP4" for Components for 'management of egovernment service mark-up']*
1.4. Workpackage Leader	*[Short name of the workpackage Leader, e.g. "UR" for Components for 'management of egovernment service mark-up']*
1.5. Delivery date (planned)	*[Planned delivery date – month in which the deliverable is available, e.g. "M20" for the preliminary version of Components for 'management of egovernment service mark-up']*
1.6. Version	*[Deliverable version]*

2. Plan

2.1. Evaluation methods	*[Planned evaluation methods for the deliverable listed in 4.3]*
2.2. Quality measures	*[Planned quality measure for the deliverable listed in 4.3]*

3. Test *[Fill up if the evaluation method = test]*

3.1. Evaluation methods	*[Applied evaluation methods for the deliverable]*	
3.2. Achieved test coverage	*[Percentage of defined requirements or functions covered by test cases]*	
3.3. Number of test cases	**3.3.1. Defined**	*[Give number]*
	3.3.2. Executed	*[Give number]*
3.4. Number of open Incident Reports	**3.4.1. High severity**	*[Give number]*
	3.4.2. Medium severity	*[Give number]*
	3.4.3. Low severity	*[Give number]*

4. Code review *[Fill up if the evaluation method = code review]*

4.1. Detected nonconformities	☐ **None** *[check if the source code conforms with the Developers' technical guidelines]* ☐ **Following nonconformities where detected:** *[check otherwise]* *[List all nonconformities]*

5. Acceptance

5.1. Responsible	*[A Leader of the task which contains in its scope internal testing. All testing responsibilities are listed in 4.3].*	
5.2. Decision	☐ **Approved** *[for test - check if 3.2. = 100% and 3.3.1. = 3.3.2. and 3.4.1. = 0* *for code review - check if 4.1. = None]*	date……
	☐ **Rework is needed** *[check otherwise]*	date……

8.5 AUDITING TEMPLATE

The Audit Report template can be found below:

FP6-2004-27020

Access-eGov

WP1: Project management

Task T1.4: Project evaluation

Audit Report

Deliverable	Identifier	*[Deliverable identifier, e.g. "D1.3"]*
	Name	*[Deliverable name, e.g. "Project Quality Plan"]*
	Workpackage	*[Workpackage identifier, e.g. "WP1"]*
	Lead participant	*[Short name of the Participant responsible for the deliverable]*
	Delivery date (planned)	*[Planned delivery date – month in which the deliverable is available.]*
	Date of submission for acceptance	*[Actual date of submission the deliverable]*
Audit summary	Performed by	*[full name of auditor and short name of partner]*
	Date	*[audit activities finished DD-MM-YYYY]*
	Involved quality procedures	*[Quality procedures for the deliverable - review and/or test]*
	Number of observed non-compliances	*[observed number – 0 or some positive number]*
	The PM was contacted due to some issue	*[N/A or NO or YES and brief specification of this issue - e. g."2 weeks delay of peer review"]*

Description of observed non-compliances:		
[brief description]		
NO.	**Questions:**	**Answers:**
1	Why did this happen?	*[N/A or brief answer]*
2	What were the consequences?	
3	Could the situation have been anticipated?	
4	Were there early signals that were unrecognised at the time?	
5	When was the situation first identified?	
6	Who should have reacted?	
7	Was it due to unclear responsibility or authority?	
8	Was it due to poor estimating?	
9	Was it due to poor planning?	
10	Has there been a failure of any partners to fulfil their obligations?	
11	Has there been a communication failure?	
12	Where have our communication processes failed?	
13	Can we correct these failures now?	
14	Has the control system failed?	
15	Is the event a direct result of current organisational policies?	
16	What have we learned from the non-compliance(s)?	
17	What changes can we implement now to prevent a recurrence?	
18	Are policy changes necessary?	
19	How can we check the learning points (the lesson) are communicated now?	

8.6 REPORT AND REQUEST FORM FOR ENVIRONMENT-ORIENTED CHANGE

Purpose: The filled form should serve as a foundation of a deeper discussion with the people facing the change and/or affected by the change.

Time: Section A can be completed by an individual in 15 minutes.

Background: As the Access-eGov project moves on changes and/or additional assumptions will occur (e.g. due to a political reform a user partner faces a change of responsibility of certain administrative services). Environment-oriented change management mainly has to focus on the assumptions regarding the change or stability of organizational and environmental conditions such as e.g. change of procedures, business processes, or training requirements. The status of these changes and assumptions has to be documented since they are usually critical success factors for the use of the new IT system. It is mainly the responsibility of the project's user partners and in particular the liaison officers (LOPA) to ensure that any change related to the project's visions of future IT usage is detected, reported and put on the agenda for scrutinizing the impact of these changes. [See Project Quality Plan for more details.]

Usage Instructions: Simply enter short answers for section A to start the change management procedure as described in the Project Quality Plan. Sections B & C will be filled out by concerned liaison officer(s) and the WP 2 leader.

The Report & Request form template used in Access-eGov Change Management can be found on the following three pages:

Access-eGov

WP1: Project management

Task T1.4: Project evaluation

Report & Request Form for Environment-oriented Change

Reported / requested by: [*name*]

Project partner:

Project role:

Date reported / requested:

Reference #: [*will be assigned by WP 2 leader*]

A. Description of Change

Type of change: *[* Check one or more boxes *]*

- ❑ Requirement (expected IT functionality)
- ❑ Scope of application
- ❑ Business process (within given procedure)
- ❑ Organizational procedure / responsibility (e.g. change of service)
- ❑ Training
- ❑ Legal & political circumstances
- ❑ Other: …

[*Describe the reported / requested change in a few* sentences]

Change motivation / driver / initiator:

Change object / topic:

Impact / risk (expected):

Optional: action recommended / other comments:

B. Assessment

[*Concerned liaison officer(s) (LOPA) and to the WP 2 leader in order to evaluate the impact and (in case of request) the feasibility of the change.*]

Assessment by liaison officer (name: _____ / date: _____):

Assessment by WP 2 leader (name: _____ / date: _____):

- ❑ WP leader(s) to be informed: _____
- ❑ Task leader(s) to be informed: _____
- ❑ In case of major change: the issue should be raised to the PCC for discussion & decision

C. Action / Solution

[*The concerned liaison officer(s) and the WP 2 leader state how the environmental change (request) is dealt with:*]

Action/solution reported by liaison officer (name: _____ / date: _____):

Action/solution reported by WP 2 leader (name: _____ / date: _____):

8.7 REFERENCES

[Kenzner, 1994] Kerzner H., Project Management, Van Nostrand Reinhold, 1994, p. 1054.

[PMI, 1996] PMI standard committee, A guide to the project management body of knowledge, 1996, p. 88.

[Schwalbe, 2004] Schwalbe K., Information Technology Project Management, Thompson Course Technology, 2004

[Trevor, 1996] Trevor L. Y., The Handbook of Project Management, Biddles Ltd, 1996, p. 191.