



Evaluation criteria and verification procedures of the ACGT platform

Project Number: FP6-2005-IST-026996

Deliverable id: D13.1

Deliverable name:

Evaluation Criteria and Verification Procedures of the ACGT Platform

Date: September 2007



Information Society
Technologies

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	ACGT
Project Full Name:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Document id:	D13.1
Document name:	Evaluation Criteria and Validation Procedures of the ACGT Platform
Document type (PU, INT, RE)	INT
Version:	0.15
Date:	September 2007
Authors: Organisation: Address:	Thierry Sengstag and WP13 Partners SIB Thierry.Sengstag@isrec.ch

Document type PU = public, INT = internal, RE = restricted

ABSTRACT:

The present document provides a list of evaluation and validation procedures for the ACGT infrastructure. It is subdivided into two major parts. The first part provides recommendations for the development of quality procedures to ensure the quality of software in each technical work package. The actual QA procedures for the ACGT software components can be found on the BSCW document server. The access to software management tools made available in the context of ACGT is also described in the present document.

The second part provides evaluation criteria from the perspective of the end-user. The latter is focused on the ability of users to utilize the ACGT infrastructure to solve practical scenarios. The scenarios are tightly bound to the data architecture currently accepted they may and will evolve with the architecture of the system.

KEYWORD LIST: Evaluation, validation, end-user scenarios, software management, software quality assurance.

MODIFICATION CONTROL			
Version	Date	Status	Author
0.1	01.02.2007	Draft	Thierry Sengstag
0.2	25.06.2007	Draft	Thierry Sengstag
0.15	04.09.2007	Draft, pre-final	Thierry Sengstag
1.0	07.09.2007	Final	Thierry Sengstag

List of Contributors

- Thierry Sengstag, SIB
- Vlad Popovici, SIB
- Norbert Graf, U Saar
- Christine Desmedt, IJB
- Francesca Buffa, UOxf
- Stelios Sfakianakis, FORTH
- Georgios Stamatakos, ICCS
- Dimitra Dionysiou, ICCS
- Robert Belleman, UvA
- Paul Melis, UvA
- Luis Martin, Alberto Anguita, UPM
- Juliusz Pukacki, PSNC
- K Marias, FORTH

Contents

1 INTRODUCTION.....	9
1 INTRODUCTION.....	9
1 INTRODUCTION.....	9
1.1 INTRODUCTION.....	9
2 OVERVIEW OF EVALUATION PROCESS.....	11
2 OVERVIEW OF EVALUATION PROCESS.....	11
2 OVERVIEW OF EVALUATION PROCESS.....	11
2.1 QA PROCEDURES FOR TECHNICAL WPs.....	11
2.1.1 <i>General Principles of Software QA</i>	11
2.1.1 <i>General Principles of Software QA</i>	11
2.1.1 <i>General Principles of Software QA</i>	11
2.1.2 <i>Actualization of QA procedures in ACGT</i>	13
2.1.2 <i>Actualization of QA procedures in ACGT</i>	13
2.1.2 <i>Actualization of QA procedures in ACGT</i>	13
2.2 END-USER EVALUATION PROCEDURES.....	14
2.2.1 <i>Overview of end-user evaluation</i>	14
2.2.1 <i>Overview of end-user evaluation</i>	14
2.2.1 <i>Overview of end-user evaluation</i>	14
2.2.2 <i>Mini-scenarios</i>	14
2.2.2 <i>Mini-scenarios</i>	14
2.2.2 <i>Mini-scenarios</i>	14
2.2.3 <i>Organization of periodic evaluations, reporting</i>	15
2.2.3 <i>Organization of periodic evaluations, reporting</i>	15
2.2.3 <i>Organization of periodic evaluations, reporting</i>	15
2.2.4 <i>Prioritized list of mini-scenarios</i>	16
2.2.4 <i>Prioritized list of mini-scenarios</i>	16
2.2.4 <i>Prioritized list of mini-scenarios</i>	16
2.1.1 Reference ACGT Architecture.....	17
2.3 SOFTWARE MANAGEMENT TOOLS.....	17
2.3.1 <i>Software repository</i>	18
2.3.1 <i>Software repository</i>	18
2.3.1 <i>Software repository</i>	18
2.3.2 <i>Software validation (automatic testing)</i>	18
2.3.2 <i>Software validation (automatic testing)</i>	18
2.3.2 <i>Software validation (automatic testing)</i>	18
2.3.3 <i>Bug reporting/tracking</i>	21
2.3.3 <i>Bug reporting/tracking</i>	21
2.3.3 <i>Bug reporting/tracking</i>	21
3 END-USER EVALUATION SCENARIOS.....	23
3 END-USER EVALUATION SCENARIOS.....	23
3 END-USER EVALUATION SCENARIOS.....	23
3.1 DEPLOYMENT OF ACGT INFRASTRUCTURE.....	24
3.1.1 <i>ACGT software installation</i>	24
3.1.1 <i>ACGT software installation</i>	24
3.1.1 <i>ACGT software installation</i>	24
3.1.2 <i>Integration of local microarray database: BASE</i>	25
3.1.2 <i>Integration of local microarray database: BASE</i>	25
3.1.2 <i>Integration of local microarray database: BASE</i>	25
3.1.3 <i>Integration of local DICOM database</i>	26

3.1.3	<i>Integration of local DICOM database</i>	26
3.1.3	<i>Integration of local DICOM database</i>	26
3.1.4	<i>Database anonymization (SQL/EHR and BASE)</i>	27
3.1.4	<i>Database anonymization (SQL/EHR and BASE)</i>	27
3.1.4	<i>Database anonymization (SQL/EHR and BASE)</i>	27
3.1.5	<i>Database anonymization (SQL/EHR and DICOM)</i>	28
3.1.5	<i>Database anonymization (SQL/EHR and DICOM)</i>	28
3.1.5	<i>Database anonymization (SQL/EHR and DICOM)</i>	28
3.1.6	<i>Public database integration, GEO</i>	28
3.1.6	<i>Public database integration, GEO</i>	28
3.1.6	<i>Public database integration, GEO</i>	28
3.1.7	<i>Public database integration, ArrayExpress</i>	28
3.1.7	<i>Public database integration, ArrayExpress</i>	28
3.1.7	<i>Public database integration, ArrayExpress</i>	28
3.1.8	<i>Bug reporting through portal and tracking</i>	28
3.1.8	<i>Bug reporting through portal and tracking</i>	28
3.1.8	<i>Bug reporting through portal and tracking</i>	28
3.2	MANAGEMENT OF VIRTUAL ORGANIZATION.....	30
3.2.1	<i>Creation of a virtual organization</i>	30
3.2.1	<i>Creation of a virtual organization</i>	30
3.2.1	<i>Creation of a virtual organization</i>	30
3.2.2	<i>Insertion/removal of institutions in virtual organization</i>	31
3.2.2	<i>Insertion/removal of institutions in virtual organization</i>	31
3.2.2	<i>Insertion/removal of institutions in virtual organization</i>	31
3.2.3	<i>Insertion/removal of users in virtual organization</i>	32
3.2.3	<i>Insertion/removal of users in virtual organization</i>	32
3.2.3	<i>Insertion/removal of users in virtual organization</i>	32
3.2.4	<i>Management of user rights (private clinical data access)</i>	32
3.2.4	<i>Management of user rights (private clinical data access)</i>	32
3.2.4	<i>Management of user rights (private clinical data access)</i>	32
3.3	TECHNOLOGY-DRIVEN SCENARIOS.....	33
3.3.1	<i>R-based analysis: Farmer scenario</i>	33
3.3.1	<i>R-based analysis: Farmer scenario</i>	33
3.3.1	<i>R-based analysis: Farmer scenario</i>	33
3.3.2	<i>PubMed mining with BEA and visualization</i>	34
3.3.2	<i>PubMed mining with BEA and visualization</i>	34
3.3.2	<i>PubMed mining with BEA and visualization</i>	34
3.4	COMPLEX QUERY (BASED ON TOP TRIAL).....	35
3.4.1	<i>Use CRF editor to create ontology-based CRFs</i>	35
3.4.1	<i>Use CRF editor to create ontology-based CRFs</i>	35
3.4.1	<i>Use CRF editor to create ontology-based CRFs</i>	35
3.4.2	<i>Use CRF editor to create clinical-data entry template</i>	36
3.4.2	<i>Use CRF editor to create clinical-data entry template</i>	36
3.4.2	<i>Use CRF editor to create clinical-data entry template</i>	36
3.4.3	<i>Anonymize local clinical database and upload in ACGT environment</i>	37
3.4.3	<i>Anonymize local clinical database and upload in ACGT environment</i>	37
3.4.3	<i>Anonymize local clinical database and upload in ACGT environment</i>	37
3.4.4	<i>Upload imaging data (DICOM data) to ACGT database</i>	38
3.4.4	<i>Upload imaging data (DICOM data) to ACGT database</i>	38
3.4.4	<i>Upload imaging data (DICOM data) to ACGT database</i>	38
3.4.5	<i>Upload biological data (IHC, FISH, RT-PCR) and images to ACGT database</i>	38
3.4.5	<i>Upload biological data (IHC, FISH, RT-PCR) and images to ACGT database</i>	38
3.4.5	<i>Upload biological data (IHC, FISH, RT-PCR) and images to ACGT database</i>	38
3.4.6	<i>Perform Oncosimulator-based analysis with data imported from ACGT and visualize results</i>	39
3.4.6	<i>Perform Oncosimulator-based analysis with data imported from ACGT and visualize results</i>	39

3.4.6	<i>Perform Oncosimulator-based analysis with data imported from ACGT and visualize results.....</i>	<i>39</i>
3.4.7	<i>Perform QC on a set of microarray data.....</i>	<i>40</i>
3.4.7	<i>Perform QC on a set of microarray data.....</i>	<i>40</i>
3.4.7	<i>Perform QC on a set of microarray data.....</i>	<i>40</i>
3.4.8	<i>Identify genes associated to survival and/or other clinical parameters.....</i>	<i>41</i>
3.4.8	<i>Identify genes associated to survival and/or other clinical parameters.....</i>	<i>41</i>
3.4.8	<i>Identify genes associated to survival and/or other clinical parameters.....</i>	<i>41</i>
3.4.9	<i>Retrieve information about a set of genes using ACGT text mining tools.....</i>	<i>42</i>
3.4.9	<i>Retrieve information about a set of genes using ACGT text mining tools.....</i>	<i>42</i>
3.4.9	<i>Retrieve information about a set of genes using ACGT text mining tools.....</i>	<i>42</i>
3.4.10	<i>Retrieve information about a set of genes using public databases.....</i>	<i>43</i>
3.4.10	<i>Retrieve information about a set of genes using public databases.....</i>	<i>43</i>
3.4.10	<i>Retrieve information about a set of genes using public databases.....</i>	<i>43</i>
3.4.11	<i>Compute Breast-Cancer Prognostic Indexes.....</i>	<i>44</i>
3.4.11	<i>Compute Breast-Cancer Prognostic Indexes.....</i>	<i>44</i>
3.4.11	<i>Compute Breast-Cancer Prognostic Indexes.....</i>	<i>44</i>
3.4.12	<i>Perform gene-set enrichment analysis.....</i>	<i>45</i>
3.4.12	<i>Perform gene-set enrichment analysis.....</i>	<i>45</i>
3.4.12	<i>Perform gene-set enrichment analysis.....</i>	<i>45</i>
3.4.13	<i>Design a workflow, store it into a workflow db, retrieve it and execute it on test data.....</i>	<i>46</i>
3.4.13	<i>Design a workflow, store it into a workflow db, retrieve it and execute it on test data.....</i>	<i>46</i>
3.4.13	<i>Design a workflow, store it into a workflow db, retrieve it and execute it on test data.....</i>	<i>46</i>
3.5	NEPHROBLASTOMA-BASED SCENARIO.....	47
3.5.1	<i>Use CRF editor to create ontology based CRFs</i>	<i>47</i>
3.5.1	<i>Use CRF editor to create ontology based CRFs</i>	<i>47</i>
3.5.1	<i>Use CRF editor to create ontology based CRFs</i>	<i>47</i>
3.5.2	<i>Use Trial Builder to maintain the Master Ontology.....</i>	<i>48</i>
3.5.2	<i>Use Trial Builder to maintain the Master Ontology.....</i>	<i>48</i>
3.5.2	<i>Use Trial Builder to maintain the Master Ontology.....</i>	<i>48</i>
3.5.3	<i>Step 1 of the antigen Scenario as described in D2.1.....</i>	<i>50</i>
3.5.3	<i>Step 1 of the antigen Scenario as described in D2.1.....</i>	<i>50</i>
3.5.3	<i>Step 1 of the antigen Scenario as described in D2.1.....</i>	<i>50</i>
3.5.4	<i>Anonymize local clinical database and upload in ACGT environment.....</i>	<i>51</i>
3.5.4	<i>Anonymize local clinical database and upload in ACGT environment.....</i>	<i>51</i>
3.5.4	<i>Anonymize local clinical database and upload in ACGT environment.....</i>	<i>51</i>
3.5.5	<i>Upload imaging data (DICOM data) to ACGT database.....</i>	<i>52</i>
3.5.5	<i>Upload imaging data (DICOM data) to ACGT database.....</i>	<i>52</i>
3.5.5	<i>Upload imaging data (DICOM data) to ACGT database.....</i>	<i>52</i>
3.5.6	<i>Use ACGT stored images and clinical data as input for the Oncosimulator....</i>	<i>52</i>
3.5.6	<i>Use ACGT stored images and clinical data as input for the Oncosimulator....</i>	<i>52</i>
3.5.6	<i>Use ACGT stored images and clinical data as input for the Oncosimulator....</i>	<i>52</i>
3.5.7	<i>Retrieve patient outcome from ACGT database and correlate it to different clinical data using other ACGT tools (e.g. R) Identify autoantibodies associated to survival and/or other clinical parameters.....</i>	<i>52</i>
3.5.7	<i>Retrieve patient outcome from ACGT database and correlate it to different clinical data using other ACGT tools (e.g. R) Identify autoantibodies associated to survival and/or other clinical parameters.....</i>	<i>52</i>
3.5.7	<i>Retrieve patient outcome from ACGT database and correlate it to different clinical data using other ACGT tools (e.g. R) Identify autoantibodies associated to survival and/or other clinical parameters.....</i>	<i>52</i>
3.6	META-ANALYSIS SCENARIO.....	53
3.6.1	<i>Store curated microarray database in ACGT database for later retrieval.....</i>	<i>53</i>
3.6.1	<i>Store curated microarray database in ACGT database for later retrieval.....</i>	<i>53</i>
3.6.1	<i>Store curated microarray database in ACGT database for later retrieval.....</i>	<i>53</i>

3.6.2	<i>Find common features (genes) between two microarray platforms.....</i>	54
3.6.2	<i>Find common features (genes) between two microarray platforms.....</i>	54
3.6.2	<i>Find common features (genes) between two microarray platforms.....</i>	54
3.6.3	<i>Integrate results from multiple microarray platforms.....</i>	55
3.6.3	<i>Integrate results from multiple microarray platforms.....</i>	55
3.6.3	<i>Integrate results from multiple microarray platforms.....</i>	55
3.7	SCENARIOS FOR FUTURE DEVELOPMENTS.....	55
4	BIBLIOGRAPHY.....	57
4	BIBLIOGRAPHY.....	57
4	BIBLIOGRAPHY.....	57
5	GLOSSARY.....	58
5	GLOSSARY.....	58
5	GLOSSARY.....	58
6		
	MINI-SCENARIOS FOR THE INITIAL ACGT DEMONSTRATOR.....	59
	PHASE I (CORE "POST-GENOMIC" ANALYSIS)	60
	PHASE II (IMAGING AND ONCOSIMULATOR)	60
	PHASE III (ONTOLOGY HANDLING, PATIENT DATA VISUALIZATION)	61
	PHASE IV (META-ANALYSIS)	61
7		
	REPORTING FORMS FOR END-USER EVALUATION.....	62
8		
	TEMPLATE FOR QUALITY ASSURANCE REPORT FOR TECHNICAL WPS.....	70
9		
	QA AND VALIDATION SCHEMES	
	IN SELECTED OPEN SOURCE PROJECTS.....	75
9.1	R.....	75
9.2	BIOCONDUCTOR.....	75
9.3	TAVERNA (MYGRID).....	76
9.4	LINUX KERNEL.....	76
9.5	MOZILLA / FIREFOX.....	76
9.6	OPENOFFICE.....	77
9.7	GNU / GCC.....	78

Executive Summary

The present document provides procedures, scenarios and guidelines related to the evaluation and validation and to the software quality assurance (QA) of the ACGT platform.

Practical mini-scenarios of relevance for end-users (clinicians, biomedical researchers,...) are proposed. Their aim is twofold: firstly to translate the end-user requirements into practical features to be implemented in the ACGT environment and secondly to provide explicit steps for the validation of that environment.

The mini-scenarios presented in this document are covering all critical aspects of the deployment and anticipated usage of the ACGT environment, from the installation of software components, to the use of the infrastructure in the framework of clinical trials, through the management of virtual organizations, the maintenance of the ACGT ontology and the integration of local and public databases.

Given the finite resources available in the project, a list of mini-scenarios to be implemented in priority is provided in 2.1.1. Those mini-scenarios are thought to form a consistent basis for a first fully integrated ACGT demonstrator.

Software management tools made available to the ACGT developer community (or soon to be) are also described. These include a software repository, a framework for software validation through automatic testing and bug reporting tool.

A template document for QA reporting by WP leaders is provided. The implementation of those procedures for individual software components is available on the ACGT BSCW server. These should be considered an integral part of the present document, although it is considered more practical to keep them as separate documents given their evolutionary nature.

The structure of the material presented in this document reflects the architecture of the ACGT environment as it is envisioned at the time of writing. Adaptations of the procedures presented here will be needed, once actual software components become available, to reflect their true functionalities and to remain of practical use. Subsequent revised versions of this document will thus be issued in the course of the project to report on those adaptations.

1 Introduction

1.1 Introduction

With the ubiquitous presence of software systems in the modern society and people's increasing reliance on them in daily life, a rigorous approach for assuring that these software systems meet user's expectations for quality and reliability became a fundamental component of software production cycle. This is especially true for the ACGT platform as that environment will be used to store and analyze real patient data, and will ultimately be used as a tool to help clinicians in their practice, providing visualization and additional information to help them refine their diagnoses.

Broadly, the quality expectations for software systems are two fold:

- the software must *do the right things*: software systems must do what they are supposed to do (end-user perspective)
- the software must *do the things right*: software systems must perform the tasks correctly (developer perspective)

These two aspects define two of the main components of the software quality assurance system (SQAS): the *validation* (does the software do the right things?) and *verification* (does the software do the things right?). Accordingly, SQAS aims at ensuring a high quality of the software product through the related validation and verification activities. These activities must be carried out by the people and the organizations responsible for developing and supporting the system in an overall engineering process that includes:

- quality planning;
- execution of selected quality assurance activities;
- measurement and analysis to demonstrate software quality to all parties involved.

Unfortunately, as the complexity and code size of the software increase, the risks of having a failure increase as well, and there is no effective general solution to the size, complexity, quality and other software engineering problems. However, by following standardized software development practices and by addressing the quality issues during the whole life cycle of the software, the likelihood of such defects and the cost incurred by them (both to users and to producers) may be greatly reduced even if not completely eliminated.

The purpose of this document is to propose a unified approach for ensuring the quality of the software products within the ACGT project, in accordance with the guidelines established in Deliverable D1.2, Section 8.2.

The implementation of this approach is adapted from various sources (ISO and IEEE standards mainly) and involves both users and developers in the process of testing of the product. Due to the high complexity of the software to be produced/integrated in the ACGT platform, this document does not attempt at covering all possible aspects of quality monitoring/ensuring for every module, but rather provide a template that should be adapted at the level of each module. As the various organizations involved in the software

development process of ACGT project have different QA policies/strategies, we reckon the need of a common approach to SQAS.

Two axes of quality evaluation are followed:

- From a top-down perspective, various categories of end-users will evaluate the ACGT platform in terms of its suitability to achieve its intended goals. The latter may be different for each category of end-user, thus the validation of the infrastructure is based on realistic scenarios for each of those categories. The scenarios list a series of anticipated results and/or goals to be achieved, which will allow measuring the performance of the platform in an objective way. This approach is covered in Section 2.1.1 of this document which details the actual end-user scenarios to be used in validation.
- From a bottom-up perspective, technical work-packages develop software components which can be *verified* independently provided a range of boundary conditions, i.e. sets of data for their interfaces. The modular organization of the ACGT environment should facilitate the establishment of such verification procedures.

The evaluation of the ACGT platform should clearly be viewed as an iterative process. Scenarios and QA procedures will evolve as new components get integrated in the environment or as some others are removed if considered useless.

Revised versions of this document will thus be issued as ACGT tools and end-user needs are refined and become available.

2 Overview of evaluation process

2.1 QA procedures for technical WPs

The goal of the QA procedures in ACGT context is to ensure that the software produced in each technical WP is **operational, interoperable** (with other ACGT components) and **compliant** with end-user specifications. The present sections reminds some general principles of software QA. The implementation of those principles in a QA procedure for individual technical WPs is given in the following sections.

General Principles of Software QA

Quality assurance activities related to software development can be categorized into software testing (including verification and validation), software configuration management, and quality control. In addition, one has to take into account the existing standards, practices and specifications that are relevant to the specifics of each module [2].

Software testing

Software testing is the most widely used risk management strategy and its goal is to verify that the functional requirements were met. However, it is of limited use as by the time the testing occurs it is too late to build quality into the product. In the case of ACGT, testing will be implemented by running the selected set of clinical scenarios on the platform.

Verification and validation will be used throughout the whole development cycle to build quality into the product. More specifically, each module has to have a clear verification strategy and associated validation scenarios. This is the responsibility of the developers.

Software configuration management

Software configuration management (SCM) is concerned with labelling, tracking, and controlling changes in the software elements of a system. It controls the evolution of a software system by managing versions of the components and their relationships. SCM comprises version control, build configuration, and change control and component identification (see [2] for details. This system should be implemented by each institution involved in software development.

Quality control

Quality control (QC) is defined as the process and methods used to monitor work and observe whether requirements are met. In the case of software, QC usually includes specification reviews, inspection of code and documents, and checks for user deliverables. In the case of ACGT, QC will be implemented by regular inspections of the code and review of the development status.

The software quality framework, as defined by ISO-9126, covers six fundamental aspects, each with its own non-overlapping sub-aspects, as summarized below [1]:

Functionality: a set of attributes related to a defined set of functions and specified properties. The functions are those that satisfy stated or implied needs. The sub-aspects include:

- suitability
- accuracy
- interoperability
- security

Reliability: those attributes pertaining to the capability of software to maintain its level of performance under stated conditions and for a stated period of time. The sub-aspects include:

- maturity
- fault tolerance
- recoverability

Usability: a set of attributes that bear of the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users. The sub-characteristics include:

- understandability
- learnability
- operability

Efficiency: those attributes that bear on the relationship between the level of performance of the software and the amount of the resources used, under stated conditions. The sub-aspects include:

- time behavior
- resource behavior

Maintainability: those attributes that bear on the effort needed to make specified modification. The sub-aspects include:

- analyzability
- changeability
- stability
- testability

Portability: a set of attributes conditioning the ability of the software to be transferred from one environment to another. The sub-aspects include:

- adaptability
- installability
- conformance
- replaceability

All these 6 aspects and the associated issues must be addressed both at the level of each module and the global level of ACGT project.

Actualization of QA procedures in ACGT

In practice, Quality Assurance procedures for the software developed in the context of ACGT are implemented at the level of the individual technical work-packages. 2.1.1 of the present document provides a template for reporting QA procedures. This template is inspired by the recommendations of the standard IEEE 730 "IEEE Standard for Quality Assurance Plans" [4, 8]. Instead of requiring a series of separate documents in principle needed for a formal implementation of the standard -which would be beyond the available manpower available in ACGT- the template attempts to cover the most important sections of those additional recommendations and standards into a single simplified QA document. A strong emphasis is made on the recommendations IEEE 830 "Software Requirements Specifications" which requires a detailed specification of the software under development, but it also incorporates aspects of user documentation (from IEEE standard 1063) which is also considered essential. The template also addresses ISO software quality aspects for comprehensiveness, although not all aspects may be relevant for all tools developed in ACGT.

The template in 2.1.1 is a generic proposal and WP leaders may propose alternative procedures if those are found more suitable given the nature of the software they develop. (For instance, the *input-process-output* structure of the module description might not be the most suitable for an event-based software, for which a description of the various responses to explicitly listed events might be more suitable.) A special attention should be provided in the description of interfaces to other ACGT software components.

Upon finalization of an ACGT software component, additional documents have to be issued (see Section 8.2 of Deliverable D1.2), those are:

- Release Note
- User Guide
- Installation and Maintenance Guide

The actual procedures written in the context of every work package for the various components of the ACGT environment are available on the document server of the project BSCW, under:

- URL: <https://bscw.ercim.org/bscw/bscw.cgi/321729>
- Path: ACGT/Workpackages/WP 13 - Evaluation & Validation/ QA Procedures

At the time of writing of the present document initial QA procedures are available for:

- Mediator

- Oncosimulator
- Workflow Designer
- Custodix Anonymization Tool (CAT)

QA procedures for the Trial Builder will be reported separately in Deliverable D2.3.

2.2 End-user evaluation procedures

Overview of end-user evaluation

End-user evaluation of the ACGT infrastructure will be conducted through a number of realistic “master-categories” scenarios covering the anticipated usage of the infrastructure, from administration of the software components to specific clinical trials. These scenarios are modularized into mini-scenarios which are isolating functionalities of the ACGT environment. The scenarios are based as much as possible on the ACGT scenarios described in Deliverable D2.1, although some requirements in that Deliverable had to be dropped, being incompatible with the initial data architecture of the system. (For instance, it is currently not possible for clinicians to identify patients at the portal level.) For each mini-scenario, the required input data are enumerated and a description of the expected results is given. The steps listed for the execution of the scenarios correspond to criteria which will help objectively rating the degree of success of the modules addressed therein.

Mini-scenarios

The aim of mini-scenarios is to test functional units of the ACGT platform, such as accessing a database from the portal, track a patient history, anonymize patient data or integrate new tools in the ACGT environment. The mini-scenarios are further subdivided into individual “atomic” tasks which form the actual units of evaluation. The success, partial failure or failure of a task is reported in a form provided for every mini-scenario. In case of a failure, an assessment of its impact on the functionality of the tested component is performed.

Every mini-scenario is described in terms of:

- Overview summarizing the functionality tested
- Required ACGT tools (dependencies on other ACGT components)
- Input data
- Expected results
- List of steps to achieve expected results
- Scenario evaluation form
- Assessment of rate of success (percent done) and assign priority to uncompleted tasks (during evaluation workshops)

Levels of success available to report the outcome of evaluation steps:

- OK: The task was accomplished as expected
- Partial: The task could be executed partially with minor loss of functionality
- Failure: The task could not be executed with an acceptable loss of functionality

Priority ranking for uncompleted tasks:

- High: describe tasks which incomplete state prevent the completion of the scenario
- Medium: describe tasks which lack non-critical functionality
- Low: describe tasks which lack some optional features

If applicable end-users will consider the following questions as basis for an overall assessment of usability of the tool under scrutiny:

- Is the general interface suitable for your purposes?
- Rate the accessibility level (easy to use, hard, too complex)
- Is on-line help sufficient?
- Is the user manual well documented?
- Do you believe that additional training is necessary to apprehend the system?
- If yes, please precise on which functionalities
- Are security mechanisms sufficient?
- Is the software free of errors that would make it possible to circumvent its security mechanisms?
- Are you satisfied with the personalization/customization features of the system?
- Is the quality of outputs/results acceptable?
- Are all parameters required by the program available?
- Are all inputs required by the program available?
- Are information processing delays acceptable: poor, fair, good
- Have you encountered any problem with the use of alphanumeric or special characters?
- To what degree is the ACGT component interoperable with your existing IT environment/equipment? (poor, acceptable, high)

Organization of periodic evaluations, reporting

Automatic testing will be used wherever applicable, using an environment developed for this purpose in ACGT and briefly described in Section 2.1.1. The advantage of automatic testing is that tests can be run frequently (e.g. daily), thus allowing a rapid discovery of problems related, for instance, to incompatibilities introduced when new versions of software components are installed.

However, for scenarios requiring user-interaction, a human evaluation has to be performed. In ACGT, this human evaluation process will be conducted in the context of workshops to be held in association with major milestones of the project, such as the delivery of demonstrators.

Various ways to organize the evaluation process exist, specifying for instance in detail the formal role of each actor involved in the process (see e.g. Chapter 6 of Reference [8]), however all share the following overall structure:

- Before the actual evaluation, documents have to be prepared that define which components have to be tested and specify the objective criteria to assess whether the evaluation is a success or not.
- During the evaluation process, the role of participants have to be defined, involving in particular a coordinator whose role is to ensure that the process is conducted in a proper way.
- After the evaluation process, the evaluation results are collected and summarized in an evaluation report.

The ACGT evaluation workshops will be organized following the same pattern. The evaluation units are formed by the individual mini-scenarios given in Section 2.1.1 (or a selection of them in the initial development phase of the project), which incorporate expected results and define for each of them semi-quantitative criteria of success. Formal reporting forms having the structure described in 2.1.1 will be used during the evaluation process to record the testing results. Those will form the backbone of the evaluation and validation reports.

During evaluation workshops, the actual testing should be performed as much as possible by representatives of the end-user community, although representatives of the developer community (ACGT technical work packages), should also be present to provide support, clarifications and minor-bug-fixing capacity.

The exact relative timing of evaluation workshops with regards to the milestones of the project should be decided by the ACGT Management.

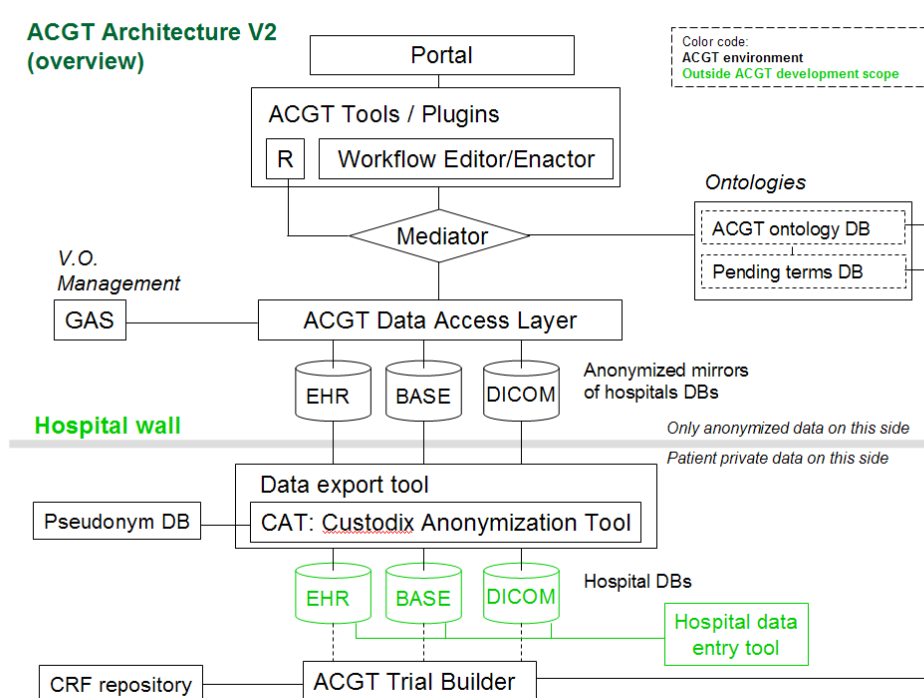
Prioritized list of mini-scenarios

2.1.1 provides a list of the mini-scenarios to be developed in priority, aiming at the first functional ACGT demonstrator. The list was reviewed by clinicians and clinical researchers and approved as providing a realistic minimal environment for clinico-genomics studies.

The selected mini-scenarios cover data management (including data access credentials), microarray-based data-analysis, Oncosimulator/imaging integration, ontology-based clinical-trial management and meta-analyses. Further details are given in the introduction of 2.1.1.

2.1.1 Reference ACGT Architecture

Anticipated scenarios for the evaluation and validation depend heavily on the data flow inside the ACGT infrastructure, which in turn depend on the architecture retained. The following figure describes the reference architecture underlying the present document. One important feature of this architecture is the isolation of the patient private information from the core of the ACGT environment described in the initial DoW; only properly anonymized data are allowed to flow outside the hospital¹. In order to facilitate data access and implementation of legacy code in the ACGT environment without rewriting interfaces, an anonymized mirror of the clinical-trial databases is maintained. It is the latter that will be accessible to clinicians and researchers, once properly authenticated.



ACGT architecture (simplified) assumed in the context of the present document.

2.3 Software management tools

The software management tools used at the level of the integrated ACGT environment include a software repository with versioning support, an automatic software testing framework (language, scripts and servers) and bug reporting tools, both for developers and end-users. Those tools are in complement to those made available by partner institutions in the context of individual technical WPs. The latter are described (if applicable) in the QA procedures implemented by technical WPs.

¹ In the present context “hospital” refers to any institution legally housing the database containing private patient information.

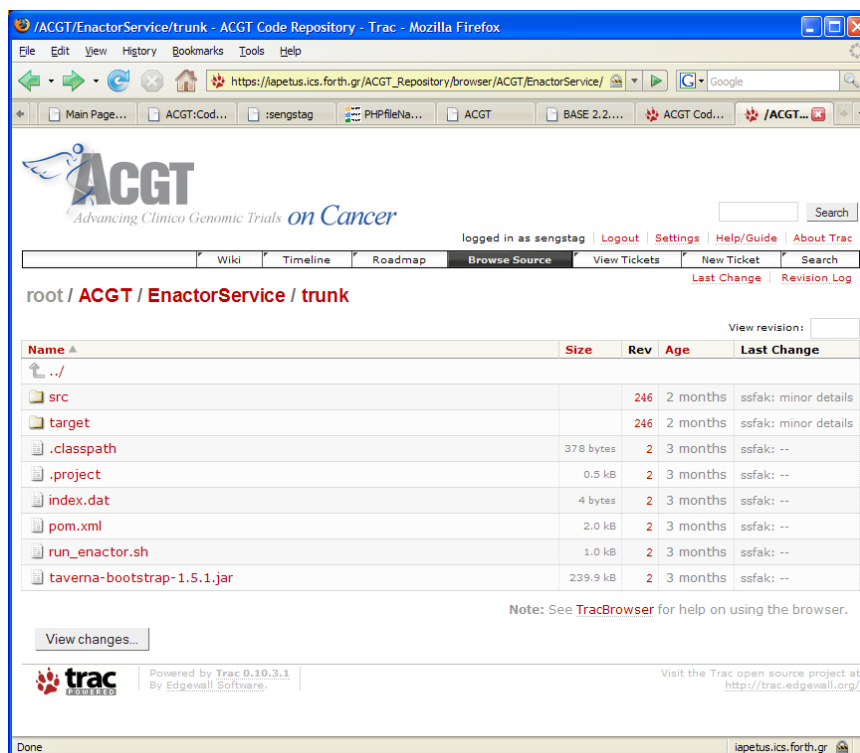
The actual status of the software management environment is available on the [ACGT wiki site](http://wiki.healthgrid.org/index.php/ACGT:Software_Management) (http://wiki.healthgrid.org/index.php/ACGT:Software_Management).

Software repository

An ACGT-wide Trac-based Subversion repository is available for sharing code between ACGT partners. It can be accessed at the URL:

https://iapetus.ics.forth.gr/ACGT_Repository/

Access to the repository is restricted to the members of the ACGT consortium and a registration procedure has been defined for gaining access. This procedure consists of choosing a username and a password by the ACGT developer, encrypting these credentials according to the Apache Web Server authentication mechanisms, and sending them by email to the maintainer of the ACGT code repository. More details are provided in the ACGT wiki.



The screenshot shows a Trac web interface for the ACGT Code Repository. The browser address bar displays https://iapetus.ics.forth.gr/ACGT_Repository/browser/ACGT/EnactorService/. The page header includes the ACGT logo and the tagline "Advancing Clinico Genomic Trials on Cancer". The user is logged in as "sengstag". The breadcrumb path is "root / ACGT / EnactorService / trunk". A table lists the files and directories in the trunk directory:

Name	Size	Rev	Age	Last Change
..				
src		246	2 months	ssfak: minor details
target		246	2 months	ssfak: minor details
.classpath	378 bytes	2	3 months	ssfak: --
.project	0.5 kB	2	3 months	ssfak: --
index.dat	4 bytes	2	3 months	ssfak: --
pom.xml	2.0 kB	2	3 months	ssfak: --
run_enactor.sh	1.0 kB	2	3 months	ssfak: --
taverna-bootstrap-1.5.1.jar	239.9 kB	2	3 months	ssfak: --

At the bottom of the page, it says "Powered by Trac 0.10.3.1 By Edgewall Software." and provides a link to the Trac open source project at <http://trac.edgewall.org/>.

Software validation (automatic testing)

In dynamic, distributed, and heterogeneous environments with multiple actors and complex use cases it is important to have a continuous validation of the different functional components. Therefore an ACGT validation and testing infrastructure is required to constantly monitor the ACGT services and report any malfunctions. This testing infrastructure for the automatic testing and validation of ACGT workflows and services will be useful both for the initial decision making process about the acceptance of a new service and for the monitoring the status of the ACGT services as a whole. In the

following list the requirements for such a testing environment and we propose a specific development platform for its implementation.

Requirements

Each ACGT service should be checked for:

- its status, i.e. if it is “alive” and responding to the clients’ requests,
- its correctness, i.e. if it delivers the correct results, working in compliance to its functional requirements,
- (possibly) its performance, e.g. if it responds in a timely fashion.

A number of tests will be developed for each service according to these criteria. These tests are stored centrally and re-evaluated in a periodic way. The results of these tests will be accessible through a user-friendly web interface and an administrator will be notified automatically if a service stops to function in accordance to its specifications.

DSL for testing ACGT services

A domain-specific language (DSL) is a programming language designed to address a specific kind of tasks. For the testing infrastructure we think that a DSL makes sense as it can provide a generic and uniform approach for the validation of services. This approach has the advantage of being independent of the language used in the implementation of the individual services, which is essential in a project integrating existing resources such as ACGT.

For this DSL we have identified the following driving forces:

- the invocation of services and workflows must be easy, terse, and compact,
- the testing of the expected behavior should be based on the philosophy of unit testing that are quite popular in many programming languages and development platforms.

The proposed DSL is heavily based on Ruby, an open source, interpreted, object-oriented, and dynamic language, which is similar in some respects to Python and Perl. Ruby also features strong meta-programming facilities that make it very attractive for building internal Domain Specific Languages.

In order to support the SOAP/WSDL Web Services interaction we have used the SOAP4R library (<http://dev.ctor.org/soap4r>) that comes as part of the Ruby distribution. Also for the unit testing functionality we are using the Test::Unit modules that provide all the assert* functionality. In addition to those components, the DSL code is using the [Libxml-Ruby](#) library, which is speeding up the XML/XPath handling.

The DSL source code is available in the ACGT Code Repository².

Overview of DSL syntax

The syntax for service validation is as follows:

² https://iapetus.ics.forth.gr/ACGT_Repository/browser/ACGT/ValidationInfr/trunk

```
require 'acgtval'  
acgt_service <name1> => <WSDL1>  
acgt_service <name2> => <WSDL2>  
test_acgt_services do  
  <commands, assertions using name1, name2>  
end
```

or, in the more compact format:

```
require 'acgtval'  
test_acgt_services <name1> => <WSDL1>, <name2> => <WSDL2> do  
  <commands, assertions using name1, name2>  
end
```

In those templates, `acgtval` is the name of a specifically written Ruby module which has to be loaded to define the ACGT validation environment.

Examples of usage of this environment in OGSA-DAI context or using a third-party service are available on the ACGT wiki site:

http://wiki.healthgrid.org/index.php/ACGT:Automated_Testing_of_Services

Execution and Monitoring

The validation tests for the ACGT services will be hosted in one or, when the need arises, more machines in the FORTH's grid node. Each test script will be stored in a separate directory with all the needed files (e.g. input data sets, WSDLs, etc). The tests will be run periodically as a UNIX Cron job. The time interval between the re-evaluations of the validation tests will be initially set to one week, a periodicity which will be adapted as experience will be gained with the system.

If the need for a finer grained or more flexible execution mechanism appears for some services, the DSL can be extended (possibly by taking advantage of the Grid technology) to suit the case at hand.

The outputs of the validation scripts will be saved in a local database with a time stamp ("validation log") and the ACGT developers will be able to monitor the status of the services in a specific portlet inside the ACGT portal. Additional mechanisms for more active notification for failures are possible, e.g. through e-mails or syndication feeds, or even by enabling the communication with the bug tracking system so that validation errors are automatically registered as a special kind of bugs to be taken care of.

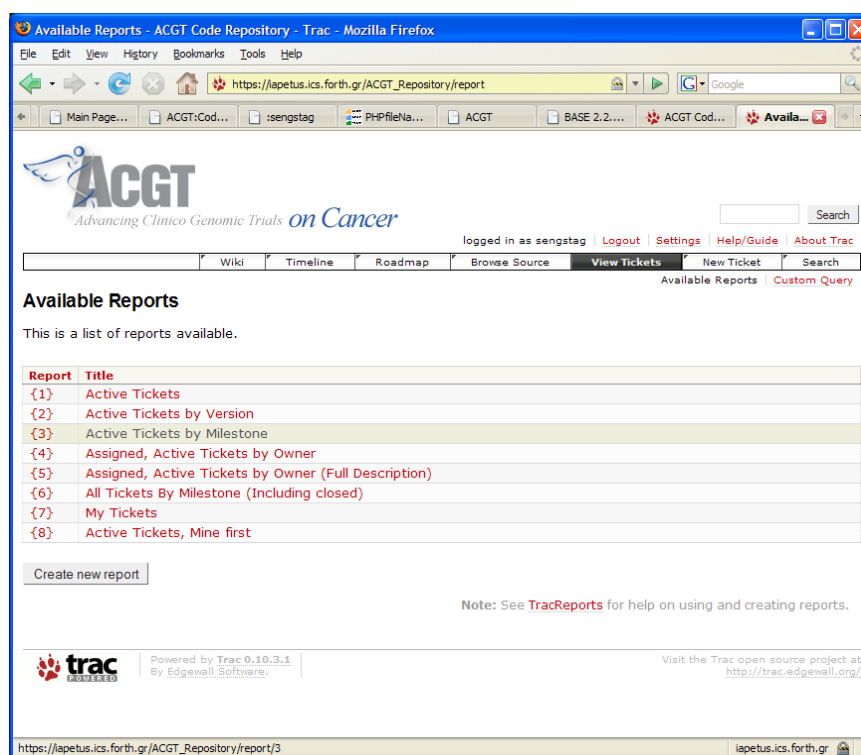
Access to the Validation Suite machines

There will be access to a test machine in the FORTH's grid node where developers can experiment and create their own validation scripts without needing to download and install additional code and libraries. This access is made through Secure Shell connections.

Details on this environment, which is still under development, will be provided in a later revision of the present document. The documentation related to the latest version of the software environment (DSL reference manual, tutorials and instructions to access the testing infrastructure and submit tests) are given on the ACGT wiki site mentioned above.

Bug reporting/tracking

Reporting/tracking of bugs and malfunctions is critical for the improvement of the platform. Currently bug-reporting features exist in the Trac front-end of the ACGT Subversion repository.



Bug reporting interface of ACGT/Trac

Trac is a powerful code-annotation tool, but its interface and functionality are essentially thought for code developers. There is thus the need for a more end-user-oriented bug-reporting tool.

At the time of writing, the final choice for such a platform was not yet done, however, based on the survey of open-source projects listed in 2.1.1, Bugzilla is a likely candidate to fulfill this role and it will be installed for testing on one of the ACGT development servers, especially as regards automatic bug notification to developers. As bug reporting in Bugzilla is not very intuitive, it is anticipated that it will ultimately be made available to end-users through a simplified portlet in the ACGT portal.

The final design and the implementation of the end-user bug-reporting tool will be detailed in a subsequent version of this document.

3 End-user evaluation scenarios

The aim of this section is to provide the evaluation criteria used to actually validate the ACGT infrastructure from the end-user perspective.

At the time of writing, the integration of most tools anticipated to be present in the final ACGT environment is still at the level of proof-of-concept, as the practical development effort having been until now conducted somewhat independently from end-users, in the context of technical work-packages.

Thus, although the mini-scenarios presented in this section will ultimately serve as basis for an explicit validation of the ACGT environment, they can also be viewed at the present stage of the project as guidelines for developers to focus their effort towards actual and immediate end-users' needs.

As mentioned above the mini-scenarios provided in this section reflect the architecture of the anticipated first integrated demonstrator. The reality of the finalized ACGT platform will almost certainly differ due to technical constraints. In particular, the details of technical implementation of those mini-scenarios³ remain to be explicitly defined in the context of each technical WP and the mini-scenarios in turn will be adjusted to match the reality of the software and infrastructure development.

The mini-scenarios presented in this section cover the most important functional goals of the infrastructure of the infrastructure. When new functionalities are added to the ACGT environment, new mini-scenarios involving them will be developed for their validation.

In practice the following fields are covered:

- Administrative scenarios linked to the setup and maintenance of the infrastructure, such as integration of databases
- Administrative scenarios linked to the management of users and institutions in the context of virtual organizations
- Technological scenarios, validating the integration of data analysis tools per-se (e.g. R) and their integration with clinical data (e.g. around the Oncosimulator)
- Clinics-oriented scenarios, validating the analysis tools as used by clinicians and biomedical researchers in realistic context
- Meta-analysis scenarios, validating the use of ACGT as clinical-research validation tool.

³ e.g. whether a menu or a clickable icon is most suitable to access a given functionality of the platform, or whether databases for meta-analysis should be accessed through the data access layer or directly accessed as web services embedded in the workflow editor.

3.1 Deployment of ACGT infrastructure

ACGT software installation

Overview

The goal of this scenario is to evaluate the adequacy of the documentation and the availability of software servers to perform a new deployment of the ACGT infrastructure.

Required ACGT tools

- None

Input data

- Installation and Maintenance Guide for the ACGT platform (collection of Installation and Maintenance guides from individual technical WPs)

Expected result

- An operational ACGT environment is up-and-running
- Installation operations appear in the log file

List of steps to achieve goal of scenario

- Follow instructions in Installation and Maintenance Guide
- Check that lower-level components of the infrastructure are functional:
 - GRID
 - Web-services access
- Check that higher-level components of the infrastructure are accessible
 - Portal
 - Virtual organization and user management interface
 - Workflow editor
 - Mediator
 - Local, v.o.-specific and public databases via data access layer
 - CRF editor
 - R engine and R-tools interface packages
 - Oncosimulator
- Check log file

Integration of local microarray database: BASE

Overview

This scenario aims at testing the procedure for the integration of a BASE microarray database in the ACGT environment. Only the mechanics of transport of data from a BASE database to the workflow editor/enactor and the modalities of access by end-user is tested. The procedure of anonymization is the object of a separate scenario.

Required ACGT tools

- BASE database with a dataset already uploaded (e.g. Farmer dataset)
- ACGT environment with data-access-layer wrapper appropriate for integration of BASE
- ACGT users belonging to the virtual organization (v.o.) which owns the database, among which users having the right to access the data in the database and users who don't, and users not belonging to the v.o.

Input data

- Administrative information to allow the mapping of BASE users with ACGT platform users

Expected result

- The integrated database is visible as a service in the workflow editor
- The data in the datasets are visible to the users having the appropriate rights (in particular users not belonging to the v.o. should not see the database in the workflow editor)
- Meta-data in the local vocabulary of the database are properly mapped onto the ACGT ontology
- Operations on the database are recorded in the log file

List of steps to achieve goal of scenario

- (Detailed steps for the integration of the database are not yet clearly defined)
- (Detailed steps for the mapping of BASE meta-data to ACGT ontology need to be defined)
- Login as users with various rights, open the workflow editor and check that the database and its contents are visible according to the users' credentials
- Browse the database to see the contents of the dataset, verify anonymity of data
- Connect the database to a workflow in the workflow editor

Integration of local DICOM database

Overview

This scenario aims at testing the procedure for the integration of a pre-existing DICOM microarray database in the ACGT environment. Anonymization of the contents of the database is treated in a separate scenario.

Required ACGT tools

- DICOM database with imaging data already uploaded
- ACGT environment with data-access-layer wrapper appropriate for integration of DICOM database
- ACGT users belonging to the virtual organization (v.o.) which owns the database, among which users having the right to access the data in the database and users who don't, and users not belonging to the v.o.

Input data

- Administrative information to allow the mapping of DICOM users with ACGT platform users

Expected result

- The integrated DICOM database is visible as a service in the workflow editor
- The data in the dataset are visible to the users having the appropriate rights (in particular users not belonging to the v.o. should not see the database in the workflow editor)
- Meta-data in the local vocabulary of the database are properly mapped onto the ACGT ontology
- Operations on the database are recorded in the log file

List of steps to achieve goal of scenario

- (Detailed steps for the integration of the database are not yet clearly defined)
- (Detailed steps for the mapping of DICOM meta-data to ACGT ontology need to be defined)
- Login as users with various rights, open the workflow editor and check that the database and its contents are visible according to the users' credentials
- Browse the database to see the contents of the dataset
- Connect the database to a workflow in the workflow editor
- Connect the database to the Oncosimulator workflow component

Database anonymization (SQL/EHR and BASE)

Overview

This scenario aims at testing the anonymization procedure, including the preservation of cross-references between the databases. As test bed a simple SQL database (EHR) containing (pseudo-)patient information for which microarrays are stored in a separate BASE database will be used. The references of records in the EHR pointing to microarray data in BASE should be preserved after anonymization. The microarray platform to be used for testing is Affymetrix, as CEL files contains potential patient identifiers (the file name is repeated in the header section of the file); this reference must be properly anonymized too.

Required ACGT tools

- Custodix anonymization tool (CAT), and data export tool

Input data

- EHR and BASE databases containing patient private information (real or simulated), i.e. "hospital database"
- Mirror EHR and BASE databases ready to accept anonymized data

Expected results

- Cross-references between database entries in the anonymized mirror are preserved (patient record in EHR pointing to microarray)

List of steps to achieve goal of scenario

- Configure the anonymization procedure:
 - List fields to be exported from the hospital databases
 - Enumerate location of potential anonymity-breaking information (e.g. file name in the body of CEL files)
 - Decide on which fields should be anonymized
 - Enumerate cross-references between databases
- Run the anonymization
- Verify that all fields are properly anonymized
- Verify that cross-references are preserved between databases

Database anonymization (SQL/EHR and DICOM)

Overview

This scenario targets the anonymization of a DICOM database. It is defined by analogy with Scenario 2.1.1.

Public database integration, GEO

Overview

This scenario aims at testing the procedure for the integration of the GEO (Gene Expression Omnibus, <http://www.ncbi.nlm.nih.gov/geo/>) public repository in the ACGT environment. Access to GEO should be a default service of the ACGT infrastructure, it should thus be visible “out of the box”.

Required ACGT tools

- ACGT environment with data-access-layer wrapper appropriate for integration of GEO database
- ACGT user

Input data

- None

Expected result

- The datasets available in the GEO database can be retrieved and used in a workflow

List of steps to achieve goal of scenario

- Login as a user and check that GEO is visible as a service in the workflow editor
- Use a known GEO accession number to retrieve the associated data (accession number for the Farmer-scenario dataset: GSE1561)
- Use the data in a workflow
- Check that accesses to GEO are recorded in the log file

Public database integration, ArrayExpress

Overview

This scenario aims at testing the procedure for the integration of the ArrayExpress repository (<http://www.ebi.ac.uk/arrayexpress/>) in the ACGT environment. The steps for this scenario are identical to those for GEO.

Bug reporting through portal and tracking

Overview

The aim of this scenario is to test the bug reporting tool available in the ACGT portal.

Required ACGT tools

- ACGT portal with bug-reporting portlet
- ACGT user

Input data

- None

Expected result

- A bug is reported to the ACGT developers using the portal
- The developers are properly notified of the bug report

List of steps to achieve goal of scenario

- Login as user, the bug reporting portlet should be accessible
- The user submits a bug
- Check that the bug is properly reported to the developers (e.g. through a mailing list)
- Check that the bug report is recorded in the log file

3.2 Management of virtual organization

Creation of a virtual organization

Overview

The goal of this scenario is to evaluate the capability of the ACGT infrastructure to manage multi-centric virtual organizations, namely to associate users from multiple institutions to a user group with identical rights on the data associated to a clinical trial.

Required ACGT tools

- ACGT portal user/virtual organization management interface (GAS)

Input data

- List of institutions to be grouped in a single virtual organization (including contact information)

Expected result

- A virtual organization exists on a perennial manner in the ACGT environment
- Relevant ACGT anonymized databases are associated to the v.o.
- All operations appear in the log file

List of steps to achieve goal of scenario

- Login into ACGT under administrative mode
- Create virtual organization and add individual institutions
- Log out and log in again to verify that the virtual organization is still registered
- Check log file

Insertion/removal of institutions in virtual organization

Overview

The goal of this scenario is to evaluate the capability of the ACGT infrastructure to adapt the structure of a virtual organization to reflect changes in the institutional membership in a clinical trial.

Required ACGT tools

- ACGT portal user/virtual organization management interface
- Registered virtual organization

Input data

- List of institutions to be added/removed

Expected result

- A virtual organization with updated structure exists on a perennial manner in the ACGT environment
- Members of institution removed from the v.o. should not have access to the databases anymore.
- All operations appear in the log file

List of steps to achieve goal of scenario

- Login into ACGT under administrative mode
- Add and remove institutions as needed
- Log out and log in again to verify that the virtual organization has the appropriate structure
- Check log file

Insertion/removal of users in virtual organization

Overview

The goal of this scenario is to evaluate the capability of the ACGT infrastructure to adapt the membership of users in institutions composing virtual organizations.

Required ACGT tools

- ACGT portal user/virtual organization management interface
- Virtual organization with several registered institutions

Input data

- List of users to be added/removed to individual institutions (including multiple affiliations).

Expected result

- The virtual organization membership reflects the new status
- All operations appear in the log file

List of steps to achieve goal of scenario

- Login into ACGT under administrative mode
- Add/remove users to/from user database
- Associate users to the various institutions
- Log out and log in again to verify that the virtual organization has the appropriate structure
- Check log file

Management of user rights (private clinical data access)

Overview

In the context of the architecture retained, private clinical data are not accessible from the ACGT data mining environment.

3.3 Technology-driven scenarios

R-based analysis: Farmer scenario

Overview

This scenario is a real-life minimal “proof-of-concept” example of research conducted with clinical data. It was chosen as one of the reference cases for the initial phase of ACGT platform development. The scenario tests the ability to use R and a database to conduct a statistical analysis. The database used in the present scenario does not require anonymization.

Required ACGT tools

- Microarray database connected to the ACGT environment
- Interface for the workflow editor with “R-template” workflow
- Web-service accessible R server

Input data

- CEL files from the Farmer scenario
- Clinical data
- Set of R commands to be executed in the R workflow

Expected results

- Reproduce the figures and results of the scenario description available on the ACGT/BSCW server (<https://bscw.ercim.org/bscw/bscw.cgi/147172>)

List of steps to achieve goal of scenario

- Connect to the ACGT portal
- Open the workflow editor with R-workflow template
- Fill in the commands to be executed
- Execute the workflow
- Compare the outcome of the workflow with the expected result

PubMed mining with BEA and visualization

Overview

The aim of this scenario is to test the proper integration of Biovista's text-mining tool, BEA, into the ACGT analysis environment.

Required ACGT tools

- ACGT infrastructure with anonymous user
- BEA available as a service in the workflow editor
- A PDF-document visualization portlet

Input data

- A list of gene-identifiers (provided as a file or as output from an analysis workflow)

Expected results

- A list of PubMed IDs most relevant to the list of genes is returned as output
- A portlet lets the user download and visualize the contents of PubMed documents.

List of steps to achieve goal of scenario

- The user connects to the ACGT environment
- The workflow editor is opened and BEA is visible as service
- The user connects the source of gene identifiers (file or workflow) to the BEA web service to request matching PubMed IDs
- The returned PubMed list is visualized in a viewer
- A click of one element of the PubMed list displays the contents of the document

3.4 Complex query (based on TOP trial)

Use CRF editor to create ontology-based CRFs

Overview

The goal of this scenario is to evaluate the capability of the ACGT infrastructure to create ontology based CRFs. The trial builder as a tool has to be integrated into the ACGT platform. All CRFs that will be created have to be stored in a CRF repository. The Master Ontology and other ontologies are needed as described in D2.2 (User requirements for an ontology based clinical data management system and for the Trial Builder). As a result an ontology based trial database will be automatically created and stored in the ACGT platform. The system can be tested with the CRF that are used in the TOP trial.

Required ACGT tools

- Virtual organization with account having write access
- Master Ontology for breast cancer
- Clinical Trial Ontology
- Interface with the Trial Builder

Input data

- Items needed for a CRF
- Thesaurus for the item - controlled vocabularies available from the Enterprise Vocabulary Services (EVS)⁴ providing a semantic integration of the many diverse medical terminologies

Expected result

- The item will be mapped to the ontology
- The trial database will be automatically created as an ontology based database

List of steps to achieve goal of scenario

- Connect to ACGT platform (URL...)
- Login into ACGT under test virtual organization with user account
- Login into the Trial Builder
- Upload files to database (Master Ontology (MO), Enterprise Vocabulary Services (EVS))
- Use the CRF Creator of the Trial builder
- Create a new item on the CRF, that is connected to the Ontology
- Set metadata to the item on the CRF

⁴ http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/vocabulary

Use CRF editor to create clinical-data entry template

Overview

This scenario aims at creating an user interface for data entry, based on the description of a clinical trial. The data entry template should allow editing existing clinical data. It should be possible to prepare data entry templates independently from an actual clinical database and to link them at a later stage. *Note: In the ACGT architecture retained for the present document, data entry is performed using existing hospital tools (e.g. Oracle clinical in one of ACGT partner institutions). This scenario should be implemented only in case a research center wishes to have his own data entry model, for instance to cover needs not addressed by commercial tools.*

Required ACGT tools

- Virtual organization with an account having write access
- CRF database with form input as described in Scenario 2.1.1 for an existing clinical database
- Tool to prepare graphical interface to database

Input data

- Existing database

Expected results

- An user-interface to access fields in a clinical database linked to ACGT is available

List of steps to achieve goal of scenario

- Connect to the ACGT infrastructure
- Open data entry form preparation tool
- Link data elements from the CRF to data-input widgets
- Save the design of the data entry form
- Associate the data entry form to the actual clinical database
- Open clinical records
- Change some fields and verify that the changes are permanently recorded to the database and logged in log file

Anonymize local clinical database and upload in ACGT environment

Overview

The goal of this scenario is to evaluate the capability of the pseudonymization/anonymization tool in the ACGT infrastructure to upload and store a real-life clinical database in the ACGT environment. As a database the TOP database can be used. The tool should help a chairman to select all data in a database that should be anonymized/pseudonymized, then this will be done and the resulting new database should be uploaded to the ACGT platform. The anonymization tool should be available via the ACGT platform.

Required ACGT tools

- Virtual organization with account having write access
- ACGT mirror database ready to accept data
- Interface for uploading data (e.g. through portal)
- Pseudonymization/anonymization tool

Input data

- Clinical data files on local computer
- Information of the database is given in D5.1

Expected result

- Personal data will be anonymized
- Clinical database is stored in the ACGT environment and is anonymized
- Information is visible only by authorized people

List of steps to achieve goal of scenario

- Access anonymization/pseudonymization tool
- Select fields in the database for anonymization/pseudonymization
- Upload files to database after anonymization of personal data
- Set access rights to data (for other members of the same virtual organization)

Upload imaging data (DICOM data) to ACGT database

The same scenario as 2.1.1. The only difference is the data type. Anonymization is also needed. Again the anonymization tool should be available via the ACGT platform, the end-user can select all entries that should be anonymized or pseudonymized. After anonymization the DICOM files should be uploaded to the ACGT platform. If this is working fine, the whole process should be automatized, in the way that the end-user does not need to define the fields for anonymization. The tool should do the anonymization process without interacting by the end-user and the data should be uploaded directly to the ACGT platform. The only duty of the end-user is the selection of the fields that should be uploaded after anonymization.

Upload biological data (IHC, FISH, RT-PCR) and images to ACGT database

The same scenario as 2.1.1. The only difference is the data type. Numeric data are usually stored into Excel files. The images are stored in an SQL database. Anonymization is also needed.

Perform Oncosimulator-based analysis with data imported from ACGT and visualize results

Overview

Imaging studies and clinical data that are stored according to 2.1.1, 2.1.1 and 2.1.1 should be available for input into the Oncosimulator. (To correlate imaging data with clinical data these data have to be pseudonymized and not anonymized.)

ACGT tools

- ACGT environment up-and-running
- DICOM and clinical database populated with images and data relevant for the Oncosimulator
- User account with access rights to databases
- Oncosimulator interface connected to ACGT

Input data

- Scenario for Oncosimulator simulation

Expected results

- Images and data are read from the ACGT databases and fed into the Oncosimulator
- An Oncosimulator simulation is successfully run

List of steps to achieve goal of scenario

- Launch Oncosimulator GUI
- Validate access to ACGT
- Browse images stored in DICOM database and load required ones in the Oncosimulator
- Run the simulation
- Visualize results
- Access to images is recorded in log file

Perform QC on a set of microarray data

Overview

This scenario aims at using workflows designed for the assessment of the quality of microarrays used in a clinical trial. Workflows are needed for a number of technologies, in particular Affymetrix, Agilent and Illumina.

Required ACGT tools

- ACGT environment up-and-running
- BASE database integrated in the system and populated with microarray data relevant for a clinical trial
- Clinical database containing patient-specific data and technical information about the microarrays (e.g. date of hybridization, potential batch identifiers (such as centers and technicians running the hybridization), amplification protocol used, etc.)
- Predefined workflows for microarray quality assessment
- ACGT user account with access right to the BASE dataset

Input data

- None (all data are contained in preexisting databases)

Expected results

- A series of plots and summary numbers relevant to assess the quality of the microarray data:
 - at the array level
 - at the batch level
 - at the whole-trial level
- A normalized gene-expression matrix

List of steps to achieve goal of scenario

- Log into the ACGT environment
- Open the workflow editor
- Check that the database for the trial is visible as data source
- Browse the workflow repository to select the QA workflow for the relevant microarray technology
- Connect the trial database to the workflow and run the evaluation
- Visualize the results, identify dubious arrays and store this information

Identify genes associated to survival and/or other clinical parameters

Overview

This scenario aims at identifying the genes having the strongest explanatory power for an observed phenotype. For instance, in the case of the TOP trial, we are interested in genes correlated with pathological complete response (binary variable) to the epirubicin therapy.

Required ACGT tools

- ACGT environment up-and-running
- Anonymized trial databases connected to the Data Access Layer (EHR and BASE)
- Workflow dedicated to survival analysis

Input data

- Survival data loaded from the trial clinical database (EHR)
- Other clinical parameter associated to samples (e.g. ER status)
- Normalized gene-expression matrix (output of a microarray normalization procedure, either retrieved from a permanent storage or computed “on the fly” in a microarray normalization workflow).

Expected results

- Ranking of array features (genes) according to the significance for the parameter(s) under scrutiny (e.g. Cox model for categories of patients or linear model).
- Produce plots (e.g. Kaplan-Meier) for the most discriminating array features / genes.
- Output ranked list of array features / genes in a format usable as input for another workflow component

List of steps to achieve goal of scenario

- Login into ACGT environment, open the workflow editor
- Build a workflow to connect BASE to the appropriate normalization workflow or select a workflow designed to retrieve intermediately stored data
- Use the previously defined workflow and a workflow that retrieves data from the clinical database as input for the survival analysis workflow
- Execute the workflow, and visualize the results (survival curves and ranked list of genes)
- Select a workflow designed to exe

Retrieve information about a set of genes using ACGT text mining tools

Overview

This scenario aims at retrieving information about a set of genes identified in an analysis.

Required ACGT tools

- Access to ACGT portal (anonymous access?)
- ACGT text-mining tool wrapper workflow

Input data

- A list of gene identifiers (might be the output of a workflow)

Expected results

- ACGT text-mining tool returns the information retrieved from its internal database for browsing by the user
- The outcome of the query is visualized by the user
- The outcome of the query is ready to be used as input for another workflow component.

List of steps to achieve goal of scenario

- Connect to the ACGT platform (portal)
- A text-mining-tool submission workflow is opened and identifiers are submitted as query to the text-mining tool
- (Alternative: the output of another scenario's workflow is pipelined is used.)
- Results are visualized within the portal

Retrieve information about a set of genes using public databases

Overview

The goal of the scenario is to provide information from public biological databases to the end-user. Two major sources of genomic information are the NCBI and the EMBL (Ensembl) who both provide web-service-based interfaces to their databases. Predefined wrappers to those services should be made available for the most demanded genomic information, i.e. PubMed, OMIM, EntrezGene, etc...

Required ACGT tools

- Access to ACGT portal
- Public-database query workflows

Input data

- Identifiers about which queries have to be performed (keywords)

Expected results

- Retrieved information is presented in an appropriate visualization tool
- Retrieved information is made available in XML format for further processing

List of steps to achieve goal of scenario

- Connect to the ACGT portal
- Open public-database-query submission workflow and query terms in the literature, gene, and OMIM database
- Browse the output in a window
- Retrieve results in XML format as output of the workflow

Compute Breast-Cancer Prognostic Indexes

Overview

This scenario aims at assessing the risk factor associated to the clinical data of a patient, using established prognostic indexes. The patient data are retrieved from the patient database and fed into a predefined workflow which is computing all prognostic indicators compatible with available data. Clinical prognostic indicators: St-Gallen, NIH, NPI. Genomic prognostic indicators: GGI, 70-gene signature, 76-gene signature. (Interface with external tools such as Adjuvant Online and Recurrence Score, should be included too.)

Required ACGT tools

- Virtual organization with account having read access
- Clinical database with patient data (clinical and/or genomic)
- Workflow with prognostic calculation algorithms

Input data

- None (see database above)

Expected results

- Prognostic indexes for the patient(s)

List of steps to achieve goal of scenario

- Login to the ACGT platform
- Open patient database and select patients for which prognostic indexes have to be computed
- Execute workflow
- Visualize results graphically and/or export results in XML format

Perform gene-set enrichment analysis

Overview

This scenario aims at identifying gene sets (e.g. MSigDB pathways, GO terms, user defined gene sets) for which there is a significant enrichment in differentially expressed genes in a microarray experiment.

Required ACGT tools

- Virtual organization with account having read access
- Database of clinical patient information
- Genomic (microarray) database connected to the ACGT platform

Input data

- Data available in databases
- User defined gene set(s), e.g. TOP2A, CHD1 in TOP trial

Expected results

- List of gene sets for which there is an enrichment of differential expression

List of steps to achieve goal of scenario

- Perform a test on genomic information to rank genes in order of significance
- Use ranking to assess enriched pathways (with and/or without significance weighting)
- Visualize the output (list of genes)
- Export list in XML format (e.g. to be fed into text mining tool on a later stage)

Design a workflow, store it into a workflow db, retrieve it and execute it on test data

Overview

This scenario aims at testing the ability of ACGT to store a workflow in the workflow database for later reuse. The workflow must be annotated, with fields describing the inputs and outputs (possibly automatically) and with an ontology-based annotation of its purpose.

Required ACGT tools

- Virtual organization with account having write access
- Workflow database
- Test clinical database
- Ontology for annotation of workflows

Input data

- A description of a procedure to be implemented as a workflow (e.g. calculation of a breast cancer prognostic index), using clinical data available in the test database

Expected results

- A functional workflow with input data taken from the clinical database and some output fields is produced and stored in the workflow database
- A query using keywords used to annotate the workflow is able to find and display it

List of steps to achieve goal of scenario

- The user connects to the ACGT environment
- The workflow editor is opened and the workflow is prepared (with connection to various fields available from the clinical database)
- The workflow is annotated
- The workflow is stored in the workflow database
- The user logs out
- The user logs in
- The workflow database is queried to retrieve the workflow
- The workflow is executed on the test clinical database (or an individual patient)

3.5 Nephroblastoma-based scenario

Use CRF editor to create ontology based CRFs

Overview

The goal of this scenario is to evaluate the capability of the ACGT infrastructure to create ontology based CRFs. The trial builder as a tool has to be integrated into the ACGT platform. All CRFs that will be created have to be stored in a CRF repository. The Master Ontology and other ontologies are needed as described in D2.2 (User requirements for an ontology based clinical data management system and for the Trial Builder). As a result an ontology based trial database will be automatically created and stored in the ACGT platform. The system can be tested with the CRFs that are used in the nephroblastoma trial.

Required ACGT tools

- Master Ontology for nephroblastoma
- Clinical Trial Ontology
- Interface with the Trial Builder

Input data

- Items needed for a CRF
- Thesaurus for the item - controlled vocabularies available from the [Enterprise Vocabulary Services \(EVS\)](#) providing a semantic integration of the many diverse medical terminologies

Expected result

- The items will be mapped to the ontology
- The trial database will be automatically created as an ontology based database

List of steps to achieve goal of scenario

- Start the Trial Builder
- Check availability of ontologies/controlled vocabularies (Master Ontology (MO), Enterprise Vocabulary Services (EVS))
- Use the CRF Creator of the Trial builder
- Create a new item on the CRF, that is connected to the Ontology
- Set metadata to the item on the CRF

Use Trial Builder to maintain the Master Ontology

Overview

The goal of this scenario is to evaluate the capability of the ACGT infrastructure to maintain and extend the ACGT Master Ontology (MO) by using the Trial Builder. Below is a summary of the functionality of the Trial Builder, a more extensive description is given in ACGT Deliverable D2.2.

With the trial builder, the clinician will have the ability to enter every question/item on a CRF. This can be done in two different ways:

1. The trial items are input into a search field which is querying the MO and the associated Thesaurus ([Enterprise Vocabulary Services \(EVS\)](#)).
2. A clinical view of the MO is presented to the clinician in a graphical way (tree with branches) allowing the clinician can to select a single item (or a whole branch) of the clinical view for inclusion in the CRF.

Two possibilities may occur in case a clinician wants to add a new item to a CRF in the above described way.

1. The item (or a synonym of the item found in an implemented Thesaurus) is found in the MO:

No further steps are necessary. By copying the item (or even a whole branch) to the CRF the link to the MO is automatically created.

2. The item or a synonym of the item is not found in the MO:

In this case the clinician is able to add this item to a selected branch of the MO, as a candidate for the extension of the ontology. A notification of the change will be automatically sent by e-mail to the curators for evaluation.

Required ACGT tools

- Servers for the ACGT Master Ontology for nephroblastoma and EVS
- Interface with the Trial Builder

Input data

- Items needed for a CRF
- Thesaurus for the items

Expected result

- The items will be mapped to the ontology
- The corresponding ontological items will be stored in the database
- The MO will be extended by new items

List of steps to achieve goal of scenario

- Start the Trial Builder
- Verify availability of ontologies / controlled vocabularies (MO and EVS)

- Use the CRF Creator of the Trial builder
- Create a new item on the CRF
- Search for the item in the MO
- Add item to MO if item is not found
- Verify that the modification of the MO is notified to a curator
- Connect to the MO database as a curator and take action regarding the proposed change in the MO. (Action can be acceptance, modification-and-acceptation or rejection of the change. In the latter case an alternative term must be proposed by the curator to avoid inconsistency (i.e. orphan entries) in the CRF linking to it.)

Step 1 of the antigen Scenario as described in D2.1

Overview

The goal of this scenario is to evaluate the capability of the ACGT infrastructure to store SEREX data in a secure way and to search public databases to describe the autoantigens that were found. The result of this description should be stored in a database in the ACGT platform and the output should also be visualized. The data of the SEREX analysis can be provided as a simple datasheet.

Required ACGT tools

- Virtual organization with account having write access
- Database (local or ACGT) ready to accept data
- Interface for uploading data (e.g. through portal)

Input data

- SEREX data files on local computer (In the final version of the scenario, the location of explicit test files will be provided, e.g. on the BSCW server.)
- Information of public databases for querying
- Workflow description of the scenario

Expected result

- SEREX data are stored in the database
- SEREX data are correctly associated with data of public databases
- Information is visible only by authorized people

List of steps to achieve goal of scenario

- Connect to ACGT platform (URL...)
- Login into ACGT under test virtual organization with user account
- Access SEREX data upload tool
- Upload files to database
(detailed list of actions is given in D2.1 of the antigen scenario)
- Associate search results of public databases and SEREX data
- Set access rights to data (for other members of the same virtual organization)

Anonymize local clinical database and upload in ACGT environment

Overview

The goal of this scenario is to evaluate the capability of the pseudonymization/anonymization tool in the ACGT infrastructure to upload and store a clinical database in the ACGT environment. As a database the SIOP 2001/GPOH database can be used. The tool should help a chairman to select all data in a database that should be anonymized/pseudonymized, then this will be done and the resulting new database should be uploaded to the ACGT platform. The anonymization tool should be available via the ACGT platform.

Required ACGT tools

- Virtual organization with account having write access
- Mirror databases ready to accept data
- Data export tool for exporting data
- Pseudonymization/anonymization tool

Input data

- Clinical data files on local computer. (SIOP 2001/GPOH trial)
- Information of the database is given in D5.1

Expected result

- Personal data will be anonymized
- Clinical database is stored in the ACGT environment and is anonymized
- Information is visible only by authorized people

List of steps to achieve goal of scenario

- Connect to ACGT platform (URL...)
- Login into ACGT under test virtual organization with user account
- Access anonymization/pseudonymization tool
- Select fields in the database for anonymization/pseudonymization
- Upload files to database after anonymization of personal data
- Set access rights to data (for other members of the same virtual organization)

Upload imaging data (DICOM data) to ACGT database

The same scenario as 6.5.4. The only difference is the data type. Anonymization is also needed. A collection of DICOM imaging studies is available for testing this scenario. Again the anonymization tool should be available via the ACGT platform, the end user can select all entries that should be anonymized or pseudonymized. After anonymization the DICOM files should be uploaded to the ACGT platform. If this is working fine, the whole process should be automatized, in the way that the end user does not need to define the fields for anonymization. The tool should do the anonymization process without interacting by the end user and the data should be uploaded directly to the ACGT platform. The only duty of the end user is the selection of the fields that should be uploaded after anonymization. This scenario is similar to 2.1.1, it is a validation with a second database.

Use ACGT stored images and clinical data as input for the Oncosimulator

Imaging studies and clinical data that are stored according to 6.5.4 and 6.5.5 should be available for input into the Oncosimulator. To correlate imaging data with clinical data these data have ultimately to be pseudonymized and not anonymized, even though with the present data architecture only anonymized data are available. The input into the Oncosimulator has to be described by WP8. This scenario is similar to 2.1.1.

Retrieve patient outcome from ACGT database and correlate it to different clinical data using other ACGT tools (e.g. R) Identify autoantibodies associated to survival and/or other clinical parameters

The clinical data of the SIOP nephroblastoma trial will be uploaded to ACGT platform after anonymization. These data can be used for statistical analysis using R. For the end user it should be easy to select different parameters that he wants to correlate with outcome, or to perform in an easy way descriptive statistical analysis regarding data from the database he can select. The output should be also visualized.

The clinical data as well as the data from the SEREX experiments will be uploaded to the ACGT platform and can be used using R as a statistical tool, that is provided by the ACGT platform. The analysis will be done automatically in a way that for every autoantibody correlations will be done regarding clinical data and outcome. The results will be listed and visualized for the end user. The end user will only select the clinical data for which he wants to get these output information. The scenario should do this analysis with every single autoantibody and a combination of different autoantibodies. In the output only those results should be presented that show a significant correlation. This analysis might create specific signatures of different autoantibodies regarding histology or outcome of patients.

This scenario is similar to 2.1.1, with the gene-expression matrix replaced by the genomic markers of the SIOP trial.

3.6 Meta-analysis scenario

Store curated microarray database in ACGT database for later retrieval

Overview

This scenario aims at testing the integration of data from public data sets in the context of ACGT-based clinical trial, for instance to validate the trial results on an independent dataset. For the time being the only large scale public dataset repositories are for microarray data, with Gene Expression Omnibus and ArrayExpress centralizing the vast majority of them. As no high-level and curated annotation of the datasets in those repositories exists, it is difficult to automate the queries to those databases and a significant curation effort has to be provided to transform the data retrieved from those repositories into workable information.

The data manipulation tools and ontologies used in ACGT environment renders possible the storage of public datasets (once the curation effort will have been conducted) in an ontology-based way which will allow reusing the information in those public datasets in other ACGT-based clinical trials.

The present scenario assumes that the datasets were curated manually to be then used to populate an ACGT-connected database (e.g. BASE). Upcoming tools such as R-based "GEOquery" might allow the preprocessing and curation of public datasets directly through the ACGT portal (via the R environment).

Required ACGT tools

- ACGT environment with database ready to receive data
- BASE database ready to accept raw microarray files and associated information
- User account with write access on the BASE database

Input data

- Individual microarray files downloaded from a public microarray repository
- Tab-delimited file describing the samples
- Annotation file describing the microarray

Expected results

- The raw files of the public dataset are visible to the members of the V.O. (or possibly to the entire community) through the Mediator

List of steps to achieve goal of scenario

- Files have to be uploaded to the BASE database using its specific interface.

Find common features (genes) between two microarray platforms

Overview

This scenario is testing the invocation of a workflow of critical importance in the context of the meta-analysis of post-genomic datasets.

Integrating results from multiple microarray platforms require the identification of which features measure the expression of the same genes. Several strategies exist for that purpose, the most frequently adopted being the following: 1. Each sequence on the individual platform is associated to a gene identifier (Entrez GeneID or HUGO symbol), 2. For each dataset measured with its platform the variance associated to each array feature is computed, 3. Among the features associated to the same gene on a platform (and for each individual dataset) the one with the largest variance is retained as the one measuring the gene expression and 4. the intersection of the lists of unique symbols for each platform is taken.

The sequence to gene-identifier mapping can be done in multiple ways, the two most common being either to use the annotation from the chip manufacturer, or to use a new mapping based on a new alignment (home made or from microarray analysis packages such as Bioconductor) of the array sequences to genome databases (e.g. GenBank). In the present scenario the manufacturer annotation will be used. (Alternative more complex workflows calling external sequence alignment web services, e.g. at EBI or NCBI, may also be designed.) Two datasets measured each with its own platform will be used in the present scenario.

Required ACGT tools

- ACGT environment with database containing datasets (public or trial-specific) from two different microarray platforms.
- Workflow for identifying common array features

Input data

- Normalized expression matrix from the two datasets, following the microarray convention with array features on the rows and samples in the columns
- Microarray annotation (or chip identifier) for the two platforms

Expected results

- Mapping between unique gene identifiers and unique array features for each platform

List of steps to achieve goal of scenario

- Log into the ACGT environment
- Open the workflow editor
- Connect the input of the array mapping workflow to the databases to be used
- Run the workflow and save the array mapping

Integrate results from multiple microarray platforms

As for scenario 2.1.1, the present scenario is based on the invocation of workflows that may need to be chained. At the time of writing the exact scenario has not yet been finalized. However it should cover microarray normalization, combination of data from multiple platforms, including some dataset from the same platform but with potential batch effect (e.g. hybridization in different laboratories).

The final workflow used for this scenarios should incorporate as much as possible the recommendations of the MAQC consortium.

3.7 Scenarios for future developments

Some features of the ACGT infrastructure are, at the time of writing of this document, not yet clearly defined, either because they address needs of lower priority (e.g. software component issuing forms for clinical data collection) or, more importantly, because the technologies they have to address are not yet mature or, finally, because they depend on technical choices which have remain to be frozen in the design of the ACGT infrastructure.

Beyond natural extensions to the scenarios described in the previous sections that will occur as the ACGT environment is developed, the fields to be developed in priority after the publication of the first prototypes:

Administrative scenarios

The initial developments of the ACGT aim at data sharing and data analysis. However, considering that administration and science are deeply intermingled in clinical trials (e.g. the hospital admission date can be considered as a variable in the analysis of a trial), there is a need to provide an integrated trial management environment.

In addition, the following technical/administrative tasks require validation:

- GRID configuration
- External web-service integration

Data persistence, intermediate results storage

In order to avoid repeating some operations, it should be possible to store intermediate analysis results to retrieve them later. For instance in microarray context, data mining is performed on the basis of the curated microarray results, which involves normalization, filtering, and some manual gene/sample selection based on QC. Those operations should not be repeated each time one works with a dataset, thus the normalized and curated matrix should be storable and reusable. It should be possible for multiple users inside a same virtual organization to be able to access these stored intermediate results.

To ensure reproducibility of results, analysis scripting should be possible.

Extension of existing scenarios, higher analysis integration level

Scenarios providing a higher level of integration in the analysis should demonstrate the added-value brought by using the ACGT environment, for instance:

- The output of Oncosimulator simulations could be correlated with gene-expression and/or clinical variables not yet taken into account in the model.

Integration of non R-based analysis tools from ACGT partners

A number of tools developed by ACGT partners and for which there is a need in ACGT context have not been described in the present document. These are the basis of two additional technology-driven scenarios to be developed, namely:

- Association rules analysis environment, developed at FORTH
- Association rules analysis environment (Prep/Engine), developed at U. Malaga

Integration of additional databases

In the scenarios described in the previous sections, the focus was put on the integration of a BASE and a DICOM database and of two widely used public databases containing public trial datasets. Wrapper for other databases actually used in hospitals (e.g. Oracle) should be developed. Besides, other sources of data should be considered as well, such as the NCBI or EMBL databases on genomic/protein annotation.

Additional high-throughput data sources

Applications of high-throughput techniques such as biomarker discovery through mass-spectroscopy-based proteomics or genotyping by shotgun sequencing (e.g. 454 or Solexa sequencing) are not yet part of routine clinical trials. However such techniques will certainly make their way in the clinics in the future and scenarios coping with this kind of data should be developed.

Extension of the base ACGT toolbox

The base ACGT toolbox should provide a number of widely-used services. These include for instance:

- Microarray normalization
- Microarray quality control
- Prognostic indexes for breast cancer
- Prognostic indexes for Nephroblastoma
- Survival analysis module
- Support for other high-throughput technologies than gene-expression microarrays (RT-PCR, sequence-based genotyping, proteomics, metabolomics, etc.)

4 Bibliography

- [1] Tian, Jeff: *Software Quality Engineering*, John Wiley & Sons, 2005
- [2] Lewis, William: *Software Testing and Continuous Quality Improvement*, CRC Press, 2005
- [3] Duggan, Evan and Reichgelt, Han: *Measuring Information Systems Delivery Quality*, IDEA Group Publishing, 2005
- [4] IEEE series of standards: IEEE 830 (Software requirements specification), IEEE 829 (Software test documentation), IEEE 1063 (Software user documentation)
- [5] ISO 9001, ISO-9126 (ISO 2001)
- [6] Wikipedia (<http://www.wikipedia.org/>)
- [7] JM Juran, AB Godfrey, *Juran's Quality Handbook*, 5th ed, Mc-Graw-Hill, 1998
- [8] MEC Schmidt, *Implementing the IEEE Software Engineering Standards*, SAMS Publisher, 2000
- [9] D Hoyle, *ISO 9000 Quality Systems Handbook*, 4th ed, Butterworth-Heinemann, 2001
- [10] J Schlickman, *ISO 9001:2000 Quality Management System Design*, Artech House, 2003

5 Glossary

Module	A software component in the broad sense (e.g. program, library) developed in the context of a work package. A work package may produce several modules, each of which may be the object of a separate report.
Module attributes	Properties of the module from the perspective of the ISO framework. For instance the “security” attribute description may express the need for an encrypted data-transfer mechanism and describe how it is (or will be) implemented in the module.
Refactoring	Action to modify a code (usually to clean and simplify it) without changing its external behaviour.
Regression testing	Any type of software testing which seeks to uncover regression bugs. Regression bugs occur when a software functionality that previously worked as desired stops working or works differently. Test suites made of test cases comparing the actual outcome of the execution of the program with its expected outcome are used to uncover regression bugs. A common strategy is to run the test suite regularly.
Unit testing	Procedure used to verify that individual units of code are working properly. Units are the smallest testable parts of an application (can be a full program in procedural programming, usually a class in object-oriented programming). Modules are made of units.

6

Mini-scenarios for the initial ACGT demonstrator

Overview

This appendix lists the mini-scenarios selected to orient the software developments towards the first integrated demonstrator. These mini-scenarios have been categorized in “Phases” reflecting the priority with which they should be implemented, in particular:

- Phase I addresses the need for an environment in which microarray data can be uploaded, retrieved and analyzed, taking into account anonymization issues.
- Phase II addresses the integration of imaging data and their use in the OncoSimulator.
- Phase III addresses the implementation of the clinical record form editor, its integration with ontology-related tools and proposes to test the procedure for the addition of new terms in the ontology. Patient-data visualization/editing is also addressed in this phase.
- Phase IV is concerned with the ability to conduct meta-analyses in the ACGT environment.

Some developments can be conducted in parallel among the various phases, but the effort should go into in priority into Phases of lower order number (i.e. into Phase I, then Phase II, etc...).

The detailed description of the individual mini-scenarios selected for the initial demonstrator can be found in the main body of the present document.

Phase I (core "post-genomic" analysis)

- Setup a virtual organization [Mini-scenarios 2.1.1, 2.1.1, 2.1.1]:
- Integrate BASE in the ACGT infrastructure [2.1.1] (connection with the mediator)
- Populate BASE with microarray data (e.g. from the Farmer scenario)
- Associate clinical data to microarrays in BASE (file in tabular format loaded in BASE or simple SQL database)
- Ensure availability of services in the workflow editor [2.1.1]
 - Workflow editor embedded in a portlet of the ACGT portal
- Create R-based workflows to [2.1.1, 2.1.1]:
 - A) Retrieve microarray (and clinical) data from BASE
 - B) Normalize the data in a R script
 - C) Extract N most differentially expressed genes from the gene expression matrix (R script, $N \sim 20$)
 - D) Gets the gene symbols for the N top genes (R script)
 - E) Get most relevant PubMed identifiers associated to the N genes from BioVista's text mining tool [2.1.1]
- Open a window to display the PDFs the articles once clicked on [2.1.1]

Extension of Phase I:

- Simulate a "hospital" BASE (and associated clinical information database) with patient private information
- Use the Custodix Anonymization Tool (CAT) to populate the mediator-connected BASE database [2.1.1]

Phase II (imaging and Oncosimulator)

- Integrate a DICOM database in ACGT infrastructure [2.1.1]
- Integrate the OncoSimulator in ACGT infrastructure [2.1.1]
- Use Oncosimulator image-analysis modules to define initial conditions for an Oncosimulator simulation
- Perform the OncoSimulator simulation on the ACGT grid

Phase III (ontology handling, patient data visualization)

- Use CRF editor to prepare a CRF; use both ACGT-ontology-certified terms and new terms candidate to be submitted to the ontology
- Visualize all data available for a “pseudo-patient” (i.e. all information related to each other by a patient identifier before anonymization)

Phase IV (meta-analysis)

- Plug the GEO repository in the ACGT infrastructure
- Plug a second database in the infrastructure
- Integrate a MAQC-like "annotation standardization tool" for microarrays (as a workflow) and use it to select common features from two array platforms picked from two different databases (including BASE as integrated in Phase I)
- Design a meta-analytic workflow to use the expression matrices associated to the selected features in multiple experiments

7

Reporting forms for end-user evaluation

Overview

The present appendix provides a template for a mini-scenario evaluation form to be used by evaluators (end-users) in validation workshops. The form is structured as follows:

- Description of who conducted the evaluation and when
- Explicit list of steps to test in the scenarios, based on the mini-scenario description found in the body of the present deliverable. (The individual steps cover the following aspects: First an inventory of the tools and data needed for the scenario is given. Then the actual steps to use the tools under evaluation are listed. Finally a list of items is provided to check that the outcome of the scenario corresponds to expectations.)
- Overall assessment of completeness/adequacy of tools under evaluation to achieve the goal of the mini-scenario.
- List of tasks that could not be completed is made explicit at the end of the reporting form. Each uncompleted task receive a priority, from the end-user perspective, to orient the future developments.

The codes to assess success level of individual tasks, the priority ranking scheme for uncompleted tasks, as well as a series of questions helping the evaluator to complete the evaluation are provided in Section .

Following the template forms for selected mini-scenarios are provided.

Scenario evaluation form (template)

Name of evaluator(s): _____

Evaluation start date: _____ end date: _____

Task	Success level	Date/visum
Prerequisite tools and data		
Scenario steps		
Expected results		

Rate of success and uncompleted-task priority assignment

Percent done: %

Comments on rating:

Uncompleted task	Assigned Priority	Date/Visum

Scenario evaluation form for mini-scenario 2.1.1

Name of evaluator(s): _____

Evaluation start date: _____ end date: _____

Task	Success level	Date/visum
Required ACGT tools available		
Input data available		
Scenario steps		
Connect to ACGT platform (URL...)		
Login into ACGT with user account		
Login into Trial Builder with user account		
Upload files to database		
Create a new item on the CRF, that will be automatically linked to the MO		
Set metadata to the item		
Expected results		
Ontology based database is automatically created and stored in ACGT platform when finishing the creation of the CRF		
Item is correctly associated with Ontology		
Metadata are stored to the item		

Rate of success and uncompleted-task priority assignment

Percent done: %

Comments on rating:

Uncompleted task	Assigned Priority	Date/Visum

Scenario evaluation form for mini-scenario 2.1.1

Name of evaluator(s): _____

Evaluation start date: _____ end date: _____

Task	Success level	Date/visum
Required ACGT tools available		
Input data available		
Scenario steps		
Connect to ACGT platform (URL...)		
Login into ACGT with user account		
Access anonymisation/pseudonymisation tool		
Select fields for anonymisation / pseudonymisation		
Upload files to database after anonymisation / pseudonymisation		
Set data access rights to database		
Expected results		
anonymisation/pseudonymisation tool is user friendly		
Fields for anonymisation / pseudonymisation can be selected		
Database is anonymised / pseudonymised		
Database is uploaded in ACGT environment		
Information is visible only by authorized people		

Rate of success and uncompleted-task priority assignment

Percent done: %

Comments on rating:

Uncompleted task	Assigned Priority	Date/Visum

Scenario evaluation form for scenario 2.1.1

Name of evaluator(s): _____

Evaluation start date: _____ end date: _____

Task	Success level	Date/visum
Required ACGT tools available		
Input data available		
Scenario steps		
Connect to ACGT platform (URL...)		
Login into ACGT with user account		
Login into Trial Builder with user account		
Upload files to database		
Create a new item on the CRF, that will be automatically linked to the MO		
Set metadata to the item		
Expected results		
Ontology based database is automatically created and stored in ACGT platform when finishing the creation of the CRF		
Item is correctly associated with Ontology		
Metadata are stored to the item		

Rate of success and uncompleted-task priority assignment

Percent done: %

Comments on rating:

Uncompleted task	Assigned Priority	Date/Visum

Scenario evaluation form for mini-scenario 2.1.1

Name of evaluator(s): _____

Evaluation start date: _____ end date: _____

Task	Success level	Date/visum
Required ACGT tools available		
Input data available		
Scenario steps		
Connect to ACGT platform (URL...)		
Login into ACGT with user account		
Connect to the Trial Builder		
Verify that Ontologies are accessible		
Create a new item on the CRF		
Search MO for the item		
Extend MO if not found		
Expected results		
Ontology based database is automatically created and stored in ACGT platform when finishing the creation of the CRF		
Item is correctly associated with Ontology		
MO is extended		

Rate of success and uncompleted-task priority assignment

Percent done: %

Comments on rating:

Uncompleted task	Assigned Priority	Date/Visum

Scenario evaluation form for mini-scenario 2.1.1

Name of evaluator(s): _____

Evaluation start date: _____ end date: _____

Task	Success level	Date/visum
Required ACGT tools available		
Input data available		
Scenario steps		
Connect to ACGT platform (URL...)		
Login into ACGT with user account		
Upload files to database		
Associate results of the search of public databases with SEREX data		
Set data access rights		
Expected results		
SEREX data are stored in the database		
SEREX data and information coming from public databases are correctly associated		
Information is visible only by authorized people		

Rate of success and uncompleted-task priority assignment

Percent done: %

Comments on rating:

Uncompleted task	Assigned Priority	Date/Visum

Scenario evaluation form for mini-scenario 2.1.1

Name of evaluator(s): _____

Evaluation start date: _____ end date: _____

Task	Success level	Date/visum
Required ACGT tools available		
Input data available		
Scenario steps		
Connect to ACGT platform (URL...)		
Login into ACGT with user account		
Access anonymisation/pseudonymisation tool		
Select fields for anonymisation / pseudonymisation		
Upload files to database after anonymisation / pseudonymisation		
Set data access rights to database		
Expected results		
anonymisation/pseudonymisation tool is user friendly		
Fields for anonymisation / pseudonymisation can be selected		
Database is anonymised / pseudonymised		
Database is uploaded in ACGT environment		
Information is visible only by authorized people		

Rate of success and uncompleted-task priority assignment

Percent done: %

Comments on rating:

Uncompleted task	Assigned Priority	Date/Visum

8

**Template for Quality Assurance
Report for Technical WPs**



Software QA description

Work package:

Module:

Author(s):

Date:

ABSTRACT:

KEYWORD LIST:

This document must be filled by every technical work package evaluation and validation delegate and sent to WP13 for consolidating into a single report. Some of the sections have to be filled in only once, while others pertain to the evolution of the software during its development cycle and must be updated regularly. If it is natural to split the software developed in the context of a work package into several separate modules, multiple evaluation and validation documents can be filled.

MODIFICATION CONTROL			
Version	Date	Status	Author(s)

Module identification

- o Purpose and scope
- o Reporting period
- o Definitions, acronyms, abbreviations
- o Dependencies
- o References

Module requirements description

- o Functional requirements
 - Functional requirement 1
 - Introduction
 - Inputs
 - Processing
 - Outputs
 - Functional requirement 2
 - Introduction
 - Inputs
 - Processing
 - Outputs
 - ...
- o External interface requirements
 - User interfaces
 - Hardware interfaces
 - Software interfaces
 - Communication interfaces
- o Design constraints and Performance Requirements
(This section addresses constraints on the module, such as execution time, error propagation, memory requirements, etc...)
- o Module attributes
(This section addresses issues related to the items in the six fundamental aspects not addressed elsewhere in the document. The subsection headers below are examples.)
 - Security
 - Portability
 - Maintainability
 - ...
- o Other requirements

Quality assurance plan

- o Software configuration and management: tools used (e.g. CVS, Bitkeeper, Bugzilla, make, ant,...), other specifics
- o Evaluation criteria for the 5 following ISO criteria: functionality, reliability, efficiency, maintainability, portability [The 6th, related to usability, is addressed in Section 3.4.]
- o Software testing (for each module)
 - Verification (during development cycle)
 - Verification plan and design (describe scope, tasks and approach to component verification)
 - Verification log (verification actions and incident reporting)
 - Incident tracking (previous issues status (open/closed), new issues)
 - Verification summary (including an evaluation of the 5 criteria above: functionality, reliability,...)
 - Validation (when freezing a version of the module)
 - Validation plan and design
 - Validation scenarios
 - Validation log
 - Incident tracking (previous issues status (open/closed), new issues)
 - Validation summary
Include an evaluation of the 5 criteria from Section 4.2: functionality, reliability,...
(report also the percentage of requirements satisfied and still missing).
- o User documentation (a “short-yet-complete” documentation including I/O specification, methods, error messages, ...)

9

QA and validation schemes in selected open source projects

9.1 R

Source of information:	website
Website:	http://www.r-project.org/
Software mgmt tools:	Subversion
Automatic testing:	regression testing of R packages on multiple OSes
Testing tools:	make + diff on a set of test scripts
Testing frequency:	daily
Testing overview:	http://cran.r-project.org/src/contrib/checkSummary.html
Bug reporting:	JitterBug (http://bugs.r-project.org/), emails, mailing lists
Other notes:	The packages RUnit provides unit testing functionalities for developments at the package level.

9.2 BioConductor

Source of information:	website, contact with Hervé Pages
Website:	http://www.bioconductor.org/
Software mgmt tools:	Subversion (https://hedgehog.fhcrc.org/bioconductor/)
Automatic testing:	regression testing on installation, and package-specific test cases, testing on multiple OSes and stable/devel versions
Testing tools:	Home-built Python-based scripts (handles dependencies and is able to make parallel builds)
Testing frequency:	hourly for core packages, daily for experimental packages
Testing overview:	http://www.bioconductor.org/checkResults/checkResults.html
Bug reporting:	email to author, mailing list
Developers' wiki:	http://wiki.fhcrc.org/bioc/DeveloperPage

9.3 Taverna (myGrid)

Source of information: website, contact with Carole Goble
Websites: <http://www.mygrid.org.uk/>
<http://taverna.sourceforge.net/>
Software mgmt tools: CVS
Automatic testing: no automatic testing; selected user cases are used for manual validation
Bug reporting: Sourceforge bug tracking, Taverna mailing lists

9.4 Linux kernel

Source of information: web
Websites: <http://www.kernel.org/>
Software mgmt tools: Git / Cogito (<http://git.or.cz/>)
Automatic testing: none official, but several independent projects – see below
Testing tools: open projects (see below) and dbench, tbench, kernbench, reaim and fsx.
Testing frequency: depends on the project
Testing overview: regression tests for official branch (maintainers of other branches use a similar testing) and several open-source projects for testing the kernel:
- <http://ltp.sourceforge.net/tooltable.php> Linux Test Project
- <http://crackerjack.sourceforge.net/cgi-bin/moin.cgi> linux regression tools
- <http://linuxquality.sunsite.dk/articles/testsuites/> test suites
- <http://kernel-perf.sourceforge.net/> for performance testing
Another layer of testing is brought by the community-based tests: every new kernel is released as “test” version waiting for the users to pick up the bugs.
Bug reporting: Bugzilla, <http://bugzilla.kernel.org/>
Developers comm.: mailing lists, <http://vger.kernel.org/>, no wiki but annual conference (Linux Kernel Developers Summit), and forums <http://kerneltrap.org/forum>

9.5 Mozilla / Firefox

Source of information: web
Websites: <http://www.mozilla.org>

Software mgmt tools: CVS

Automatic testing: several ways (see below)

Testing tools: as there is large number of projects, each with its own needs, the number of test tools is huge. A full list is available at http://wiki.mozilla.org/MozillaQualityAssurance:Test_Suite_and_Tool_Inventory

Testing frequency: nightly builds and tests

Testing overview: a project is responsible for QA of the Mozilla foundation tools: <http://quality.mozilla.org/> Mozilla relies on the testing efforts of community members. Several websites organize the results of the tests: - <http://litmus.mozilla.org/> - “integrated testcase management and QA tool that is designed to improve workflow, visibility, and turnaround time in the Mozilla QA process” which assembles the test reports. There is a “Basic Functionality Test” and a “Full Functionality Test”. Testcases are web-based: http://litmus.mozilla.org/show_test.cgi

Bug reporting: bugzilla

Developers’ wiki: <http://developer.mozilla.org/> and <http://developer.mozilla.org/en/docs/QA>

Communication tools: web-based community

9.6 OpenOffice

Source of information: web

Websites: <http://www.openoffice.org/>, <http://qa.openoffice.org/>

Software mgmt tools: Subversion (previously CVS)

Automatic testing: yes

Testing tools: qatesttool (ftp://ftp.oodev.org/pub/qa/qatesttool_ooo201*); QA-track

Testing frequency: per release and per build (full tests take 1 week)

Testing overview: automatic and manual testing; various layers of testing: global tests and incremental tests; manual tests based on test-cases. See <http://wiki.services.openoffice.org/wiki/OAAutomation> and <http://qa.openoffice.org/ooQAReloaded/ooQA-ManualTesting.html>

Bug reporting: bugzilla, <http://www.openoffice.org/issues/query.cgi>

Developers’ wiki: http://wiki.services.openoffice.org/wiki/Main_Page

Additional website: <http://qa.openoffice.org/>

9.7 GNU / GCC

Source of information: web

Websites: <http://gcc.gnu.org>

Software mgmt tools: Subversion

Automatic testing: None

Testing tools: regression tests after build (make check-gcc) uses DejaGnu (<http://www.gnu.org/software/dejagnu/>)

Testing frequency: every build

Testing overview: two levels of regression tests (“serious” and “all”); each user building the compiler suite is encouraged to submit the results of make check-gcc

Bug reporting: bugzilla, <http://gcc.gnu.org/bugzilla/>, mailing list: gcc-bugs@gcc.gnu.org <http://gcc.gnu.org/bugs.html>

Developers’ wiki: <http://gcc.gnu.org/wiki/>

Additional web site: <http://www.gnu.org/software/devel.html>