



www.gen2phen.org

D1.1. Specification of Procedures for Quality Testing of Software

WP1 – SCIENTIFIC COORDINATION

**V2.1
Final Draft**

Lead beneficiary: ULEIC
Date: October 2008
Nature: Report
Dissemination level: PU



 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

TABLE OF CONTENTS

DOCUMENT INFORMATION	3
DOCUMENT HISTORY	3
DEFINITIONS	4
1. INTRODUCTION	5
2. GENERAL RECOMMENDATIONS	6
3. FORMAL GUIDELINES	9
4. QUALITY ASSURANCE REPORT	12
ANNEXES	13
ANNEX I: GEN2PHEN QUALITY ASSURANCE REPORT TEMPLATE	13

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

Document Information

Grant Agreement Number	HEALTH-F4-2007-200754	Acronym	GEN2PHEN
Full title	Genotype-To-Phenotype Databases: A Holistic Solution		
Project URL	http://www.gen2phen.org		
EU Project officer	Frederick Marcus (Frederick.Marcus@ec.europa.eu)		


Deliverable	Number	1.1	Title	Specification of Procedures for Quality Testing of Software
Work package	Number	1	Title	Scientific Coordination

Delivery date	Contractual	Month 9	Actual	Month 10
Status	Final draft		Final <input checked="" type="checkbox"/>	
Nature	Report <input checked="" type="checkbox"/> Prototype <input type="checkbox"/> Other <input type="checkbox"/>			
Dissemination Level	Public <input checked="" type="checkbox"/> Confidential <input type="checkbox"/>			

Authors (Partner)	A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)			
Responsible Author	A.J. Brookes		Email	ajb97@leicester.ac.uk
	Partner	ULEIC	Phone	+44 (0)116 2523401

Document History

Name	Date	Version	Description
A.J. Brookes (ULEIC) and C. Diaz (FIMIM)	18/08/08	V0	Document creation
A.J. Brookes (ULEIC)	28/08/08	V1	Document refinement
Carlos Diaz (FIMIM)	02/09/08	V1	Document review
David Atlan (PHENO)	18/09/08	V1	Document review
Paul Flicek (EBI)	05/10/08	V1	Document review
A.J. Brookes (ULEIC)	08/10/08	V2	Final draft
Raymond Dalgleish (ULEIC)	13/10/08	V2.1	Consortium review

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

Definitions

- Partners of the GEN2PHEN Consortium are referred to herein according to the following codes:

ULEIC – University of Leicester (UK) – Coordinator

EMBL – European Molecular Biology Laboratory (Germany) – Beneficiary

FIMIM – Fundació IMIM (Spain) – Beneficiary

LUMC – Leiden University Medical Center (Netherlands) – Beneficiary

INSERM – Institut National de la Santé et de la Recherche Médicale (France) – Beneficiary

KI – Karolinska Institutet (Sweden) – Beneficiary

FORTH – Foundation for Research and Technology Hellas (Greece) – Beneficiary

CEA – Commissariat à l’Energie Atomique (France) – Beneficiary

EMC – Erasmus Universitair Medisch Centrum Rotterdam (Netherlands) – Beneficiary

UH.FGC – Helsingin Yliopisto (Finland) – Beneficiary

UAVR – Universidade de Aveiro (Portugal) – Beneficiary

UWC – University of the Western Cape (South Africa) – Beneficiary

CSIR – Council of Scientific and Industrial Research (India) – Beneficiary

SIB – Swiss Institute of Bioinformatics (Switzerland) – Beneficiary

UNIMAN – The University of Manchester (UK) – Beneficiary


BIOBASE – BioBase GmbH. (Germany) – Beneficiary

deCODE – Islensk Erfoagreining EH (Iceland) – Beneficiary

PHENO – Phenosystems S.A. (Belgium) – Beneficiary

BCP – Biocomputing Platforms Ltd. Oy (Finland) – Beneficiary

- Grant Agreement:** The agreement signed between the beneficiaries and the European Commission for the undertaking of the GEN2PHEN project (HEALTH-200754).
- Project:** The sum of all activities carried out in the framework of the Grant Agreement by the Consortium.
- Work plan:** Schedule of tasks, deliverables, efforts, dates and responsibilities corresponding to the work to be carried out for the GEN2PHEN project, as specified in Annex I to the Grant Agreement.
- Consortium:** The GEN2PHEN Consortium, conformed by the above-mentioned legal entities.
- Consortium agreement:** Agreement concluded amongst GEN2PHEN participants for the implementation of the Grant Agreement. Such an agreement shall not affect the parties’ obligations to the Community and/or to one another arising from the Grant Agreement.

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

1. INTRODUCTION

This Deliverable is motivated by the need to ensure that all informatics products and services created by GEN2PHEN are of the highest possible quality and reliability.

To this end, the following text provides our Consortium members with:

- i) an explanation of software development principles,
- ii) guidance on putting good software development principles into practice, and
- iii) a means for specifying the good software development principles that were employed in the production of each piece of software emerging from GEN2PHEN.

To avoid any confusion, this document is intended to provide strong and helpful guidance, rather than any form of dictate. Also, it concerns the processes of development that lead to the release of a new major version of some software, a tool, or a database, whether starting from scratch or from an earlier major versioned release of that product. The document is not intended to cover the development of small software development exercises for items that are not being issued under the banner of GEN2PHEN, nor for the development of large integrated systems such as the Ensembl or Uniprot platforms. Further still, this document is not designed to guide activities such as the development of data standards or the creation of ontologies.


The specific purpose of this document is twofold:

A: To provide guidelines on how to undertake software and database development. The use of best practice in software development not only leads to a better final product, but also saves considerable time and money. For example, it has been estimated that defects identified at the initial design stage require 50-fold less effort to remedy than if they are discovered later at testing phase of product development.

B: To provide a template for a Quality Assurance Report (QAR) that should accompany each project deliverable and software product released by GEN2PHEN to the public. This statement will concisely summarise the range of design, construction and testing procedures employed during software development.

It should also be noted that we are confident that GEN2PHEN teams are highly competent and experienced software developers. Hence, this document is expected to do nothing more or less than re-encourage good practice, and provide a consistent means for summarising the software development procedures employed upon release of software tools or code to the community.

All ideas and principles included in this document are taken from 2 books [Rakitin, Steven R. Software verification and validation: A practitioner's guide. 1997 Artech House; and Horch, John W. Practical guide to software quality management. 1996 Artech House], and these are merged with operational experience of various GEN2PHEN Partners.

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

2. GENERAL RECOMMENDATIONS

2.1. Development environment

- Try to achieve an appropriate balance between allowing programmers to select and maintain the development environment that they feel most comfortable with (operating system, text editor/IDE etc), versus the extent to which the project needs to impose a particular system for smooth interaction with the rest of the project.

2.2. Revision control

- Teams are advised to make it mandatory that programmers use a revision control system, such as Subversion or Git. This is particularly critical since aspects of each Partner's code depository will be made public via transparent links within the project's Knowledge Center website.

- It is advisable to develop significant new functionality on a branch, and merge it back into the main trunk (or HEAD in CVS terminology) only after it has been fully tested and proven to be functional.

- Tag milestones, such as major releases, with a consistent name so that they can easily be tracked. Similarly, link branch and schema names to avoid any confusion about which software version applies to which database schema and vice versa.

- Assign a particular person or small group with the task of reviewing construction work, especially contributions being made by non-core contributors.


- Code commits to the revision control system should only be allowed from individual user accounts (as opposed to shared accounts), and the inclusion of meaningful commit messages should be strongly encouraged. Code commits should also happen frequently and regularly, and there should be automated email notification of these commits to other members of the team.

2.3. Coding standards

- Wherever possible, use a widespread language for implementation (e.g. C, Perl, Java or Python), and minimise the number of different languages that are used within a project to aid maintainability and interactions.

- It is preferable to employ open technologies, which do not require payment of licenses for distribution (e.g. MySQL, versus 4D).

- Encourage modularity and object oriented techniques for non-trivial code development, balanced against the importance of creating code that does its job well.

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

- Build base classes which encourage easy use of a standard which can be extended to provide additional functionality.
- Code syntax rules and formal style guides are generally unnecessary, but esoteric language-specific constructs which are difficult for the casual reader to interpret should be avoided. Variables, methods and modules should be sensibly named according to their purpose.
- Decide on a naming convention and stick to it religiously. Ideally, use pre-existing conventions that correspond to the language of the development. For example, Ensembl uses consistent module and method names so that users can guess the right one to use even if they've never employed that specific module before. The same principle applies for separating words, either using the Perl convention of mixed case for module names (e.g. GeneAdaptor) or underscores for method names (e.g. fetch_all_by_biotype()).

2.4. Documentation


- Using 'Status labels' is a good way to label and distinguish methods that are stable versus those that are likely to change. For example, Ensembl uses a simple two-tier system; "stable" and "At risk". Regular reviews should be undertaken to reconsider the status of methods, with the aim of promoting "at risk" methods to "stable" where possible.
- Stress the value of including brief but meaningful comments through the source code, and always at the start of each function or method to define its purpose. Language-specific guidelines for commenting should be followed (e.g. pdoc for Perl or JavaDoc for Java) to enable automated creation of documentation from the source code. This includes the use of standard POD/Javadoc "fields" for labeling parameters, return types, exceptions etc.
- Keep good and high-level documentation, in a central location. Update it regularly and review occasionally to make sure it's still current and relevant. Produce a CHANGELOG file, in the standard format, so that users can easily tell what has changed between releases.

2.5. Code reviews

- Hold frequent code reviews, ranging from two people developing a piece of software together to a group of people reviewing a particularly problematic piece of code, or studying a particularly elegant solution (see further notes below).


2.6. Testing

- Module testing is vital, and this should be professionally managed (see further notes below). All developers should use a purpose built 'test suite' as a matter of routine.

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

2.7. Extreme Programming

- This very distinct software development approach may have a role in some projects, and it implies a manner of working which supersedes or extends many of the guidance points in this Deliverable. It is based heavily upon principles of strong team work, paired coding, simple design steps, obsessive testing, and dynamic response to client demands.

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

3. FORMAL GUIDELINES

Software development involves three major activities; ‘Design’, ‘Construction’, and ‘Testing’. These activities are not usually undertaken as a single cycle of sequential steps. Instead, software development usually progresses by overlapping sub-cycles of development. These sub-cycles typically include minor in-process alterations to the overall project design as ideas are tried and adapted (e.g., replacing code modules and algorithms with more elegant, generic, and faster solutions). With Extreme Programming, the steps are so incremental and intertwined that they become indistinguishable. There may also be a need for quite substantial functionality alterations and enhancements to the overall project design, innovated during the product’s development (typically expressed as sub-versions of the software).

Despite this real world complexity and non-linearity of the overall software development process, it remains convenient to discuss the three major conceptually different work components (Design, Construction, and Testing). Each of these is therefore considered in more detail below, wherein the sub-components of these activities are identified and explained. An understanding of these sub-components will be needed in order to complete a Quality Assurance Report (QAR) for each piece of software that is developed in the GEN2PHEN project.


3.1. DESIGN

Configuration Identification: Standard development work should start with a clear concept, based upon thorough and careful planning, all of which is expressed in some written form that we refer to as a Configuration Identification. At the very least, the Configuration Identification should state the purpose of the software product in terms of ‘**Requirement Specifications**’. An exception would be ‘Rapid Application Development’ (RAD) projects, wherein the system is being designed and built simultaneously. RAD projects should still start with a reasonable design concept, framework, and set of goals (e.g., a general project objective and an initial data model), but these will be far less precisely stated and not in the form of a recognisable Configuration Identification.

Quality Metrics: When a Configuration Identification is produced, each Requirement Specification should be matched with one or more ‘**Quality Metrics**’ that comprise a quantitative indication of the level of functionality intended to be achieved.

Configuration Control: Ideally, a system should be put in place whereby changes to the Configuration Identification can be properly evaluated and justified, and this is called Configuration Control process.

Configuration Auditing: Related to Configuration Control, a system should be put in place whereby the history and status of the project can be recorded.

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

3.2. CONSTRUCTION

To ensure that the software or database product construction process proceeds optimally, frequent **Reviews** of the work should be conducted. Typically, these might involve such activities as:

- **Informal Peer Reviews:** These would entail presentation of the state of progress of components of the work to others in the development team so that difficulties and possible solutions can be discussed, and so that everyone knows the state of play of the project as a whole

- **Walk-Through Reviews:** These are more structured presentations and discussions, often involving people beyond the core development team. Typically slides or other documentation will be prepared in advance, demonstrations will be given, and notes kept on the substance and the outcome of the whole discussion.


- **Inspection Reviews:** These are highly structured reviews wherein specific tasks are assigned to different individuals in the Review Meeting, and a pre-designed procedure is followed to examine the project's progress and to record the outcome and the conclusions of the meeting. Experts from other projects will often be invited to join these reviews to ensure objectivity and thoroughness.

3.3. TESTING

Testing occurs at various levels, from the checking of lines of code and algorithms by the coder (**'Debugging'**), through to **'Module/Unit Testing'** of software units, and right up to **'Integration Testing'** of systems formed from multiple different modules. Debugging and Module Testing are referred to as **'White Box'** testing because the testers are typically very familiar with the internal workings of the product being tested. Integration Testing may be 'white box' in nature, or **'black box'** if it is focussed more on functionality and/or being conducted by people other than the original coders of the software.

- **Issue Tracking** is important to control. As testing proceeds and problem issues are identified, it is necessary to record those issues, and their resolution, in a system such as Trac. The system should be linked directly to the code depository. Using tools such as Trac ensures that all the project participants are kept fully aware of the nature and status of all known problems, and provides an effective means for keeping track of what problems have and have not been remedied.


- **Module and Integration Testing** are best performed in a formal or at least semi-formal manner. This should involve the creation and documentation of components that will assist in the testing process; such as a **'Test Plan'**, a **'Test Procedure'**, one or more **'Test Scripts'**, **'Test Cases'**, and **'Test Data'**.

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

- **A Requirements Traceability Matrix** should be drawn up for thorough black box tests when working from a Configuration Identification. This is a document that specifies all the tests undertaken and relates each test item to one or more Quality Metrics themselves matched to the original Requirement Specifications.

- **Alpha Testing** should be conducted following successful Module and Integration Testing. This involves full functionality assessment achieved by letting ‘trial users’ work with and report back on how well the product performed. These trial users are typically colleagues or associates of the development team that work in the same arena as the intended eventual real-world users of the software. Their findings should be documented, and will typically result in some redesign of the Configuration Identification followed by a further cycle of construction and testing to implement those changes.


- **Beta-Testing** is the final stage of product development. This involves putting the software into the real-world for use by selected members of the real user community. In the case of GEN2PHEN this will typically be accomplished by work with Consortium Partners not involved in the development work, and by tapping into outreach and community partnership activities such as the Project Pilots and the Knowledge Center.

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

4. QUALITY ASSURANCE REPORT

It is proposed that, as standard practice, a ‘Quality Assurance Report’ (QAR) should accompany each and every versioned release of a software, service, or database emerging from and accredited to GEN2PHEN. The QAR need not be long and burdensome to create, but it should be sufficiently detailed to elucidate what quality control and software development procedures were used to generate the product being released.

The QAR should be based upon the template that is provided for it (see Annex I), and this itself has been based upon the notes provided in section 3 of this Deliverable. Those notes, and this deliverable as a whole, should therefore provide a useful guide when completing a QAR.

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

ANNEXES

Annex I: GEN2PHEN Quality Assurance Report Template


This Quality Assurance Report (QAR) is designed to accompany a Software Unit, a Database Component, or an Informatics Service produced by the GEN2PHEN Consortium (www.gen2phen.org) as part of this project’s mission to ‘help unify human and model organism genetic variation databases towards increasingly holistic views into Genotype-To-Phenotype data, and to link this system into other biomedical knowledge sources via genome browser functionality’.

The QAR provides users with a concise but meaningful indication of the quality control and software development procedures that were used to generate the particular software product stated below:

Software Unit, Database Component, or Informatics Service covered by this QAR (name of product, and version details):

Purpose of the Software Unit, Database Component, or Informatics Service (brief details of the product’s purpose and relationship/dependency to other products):

Team(s) that produced the Software Unit, Database Component, or Informatics Service: (contact details for each team, and role played in the product’s development):

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

Release date for the Software Unit, Database Component, or Informatics Service:

DESIGN ACTIVITIES

Configuration Identification:

State how the design concept was conceived, and to what degree the project design was formalised. Was a Rapid Application Development approach employed.

Requirement Specifications:


If available, list each of the project's Requirement Specifications, and wherever possible express this in terms of a quantitative Quality Metric.

Configuration Control:

Except in the case of RAD projects, describe the procedures used for Configuration Control (i.e., for managing changes to the Configuration Identification).

Configuration Auditing:

Except in the case of RAD projects, describe the procedures used for Configuration Auditing (i.e., for recording changes to the Configuration Identification)

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final


CONSTRUCTION ACTIVITIES

Revision Control System:

State the coding language(s) used for the project's implementation, and whether or not a revision control system employed to track and archive the software development work.

Reviews During Product Construction:

Indicate the type and frequency of reviews that were used to optimise the construction work, for example Informal Peer Reviews, Walk-Through Reviews, Inspection Reviews, or Other procedures.

 HEALTH-200754	D1.1 – Specification of Procedures for Quality Testing of Software		
	WP1: Scientific Coordination		Security: PU
	Author(s): A.J. Brookes (ULEIC), G. Proctor (EBI), C. Diaz (FIMIM)		Version: v2.1 – Final

TESTING ACTIVITIES

Module and Integration Testing:

Describe the white box and black box testing procedures that were used to assess the final product, stating who performed these tests. Give further details if these were formal and documented tests, specifying whether or not they employed a Test Plan, a Test Procedure, one or more Test Scripts, Test Cases, Test Data, or Other means of facilitation. Were any weaknesses or defects discovered that were not resolved before product release.

Issue Tracking:

Describe what form of issue tracking was employed in the project.

Requirements Traceability Matrix:

Was a Requirements Traceability Matrix utilised, and if so give details of the test items and their correlation to the original Quality Metrics and Requirement Specifications. Specify any tests that the product was unable to pass or areas of identified weakness in the product.

Alpha Testing:

State whether Alpha Testing was performed, and if it was then give details of the nature and extent of that testing, who performed it, and how it was documented. Specify any features of the product that raised concern in that testing if they are still to be resolved.