iBudget

Software Quality Assurance Plan

Approvals:	L. Assayah	LA	date:	2/13/2012
	C. Leung	CL	date:	2/13/2012
	Q. Pham	QCP	date:	2/13/2012
	V. Velev	VV	date:	2/13/2012
	J. Reimels	JR	date:	2/13/2012

Revision History

Date	Author	Version	Reason
2/12/12	V. Dineva & L.Assayah	1.0	First Draft

This SQAP follows the IEEE standard 730-1989.

1. Purpose

The purpose of this plan is to define iBudget's Software Quality Assurance Plan (SQAP). It also specifies the standards and procedures used in carrying out all software quality assurance (SQA) activities. In addition it defines the tools, techniques, and methodologies to support SQA activities and SQA reporting.

The goal of the SQA program is to verify that all deliverable software and accompanying documentation meet the technical requirements. The SQA procedures defined in this document will be used to examine both software and documentation to determine compliance and ensure the quality during all phases of the project, namely the design, development, testing and validation phases of the project.

2. Referenced documents

See section 4.2

3. Management

3.1 Organization

The Quality Assurance (QA) leader is responsible for:

- Implementing the quality program.
- Reviewing and approving the iBudget's SQAP.
- Resolving and following-up on any quality issues raised by SQA.
- Developing and maintaining documents such as the Program Management Plan, Test Plans, and this SQA Plan.

All team members are responsible for:

- Reviewing and commenting on the iBudget's SQAP.
- Implementing the quality program in accordance with this SQA Plan.
- Resolving and following-up on any quality issues raised by SQA related to software design and development.
- Identifying, implementing, and evaluating the quality factors to be implemented in the software.

Once in the testing phase, cross-developer testing will take place.

3.2 Tasks

QA tasks shall include

- documentation
- review meetings
- verification (including inspections) of
    - Software Requirements Analysis Process
    - Software Design Process
    - Software Implementation and Unit Testing
    - Evaluate Configuration Management Process
- validation (mostly testing)
- identify standards and guidelines
- activities designed to improve the quality assurance process itself

These tasks are detailed in this document.

3.3 Responsibilities

The quality assurance leader is responsible for the completions of tasks in 3.2, and to ensure that the prescriptions in this document are followed, including scheduling the reviews specified. Vanya Dineva was chosen as the QA leader for the iBudget project.

While each team member is responsible for the quality of his or her work, the ultimate responsibility for the quality of the iBudget application and associated documentation produced by the iTeam rests with the Project Manager. The QA Leader shall implement the SQA procedures defined in this plan and shall monitor project staff activities and review products for compliance to the standards and procedures. The results of SQA monitoring and analysis along with SQA recommendations for corrective action shall be reported to the Project Manager. In addition, all documents and software approved by the Project Manager for release shall have been reviewed and approved by the quality assurance leader.

4. Documentation

4.1 Purpose

All documentation that supports the iBudget product or its development process shall be created and updated periodically throughout the development cycle. In addition, to ensure the adequacy of these documents, they will undergo a peer review process.

4.2 Minimum documentation requirements

The following documents will be produced.

- SQAP: software quality assurance plan (this document)
- SCMP: software configuration management plan
- SPMP: software project management plan
- SRS: software requirements specifications
- SDD: software design document
- STD: software test documentation
- User's manual
- Maintenance plan

In addition to these documents, the PHP source code will utilize PHPdoc, and will therefore be capable of auto-generating documentation.

5. Standards, practices, conventions and metrics

5.1 Purpose

This section describes the standards, practices, conventions and metrics to be used for the iBudget project. These are intended not only to ensure quality, but also to obtain quantitative metric data on the SQA process itself. This data is to be used to help evaluate the CMM level of the iTeam.

5.2 Content

Standards: the IEEE standards, with appropriate modifications, are to be used for documentation.

Practices:

1. To verify the delivery of a fully conforming, high-quality product, every individual assigned to the project will participate in quality assurance.
2. Since providing quality software that meets the wants and needs of our customers is our topmost priority, engineers should apply quality precepts while they work, rather than as an afterthought.
3. All project artifacts are to be inspected by at least 2 reviewers and are made available to the team on the team's Google project page.
4. All processes are to be reviewed at least once for improvement, and the written results forwarded by the project leader via email.

Writing Conventions: All documentation, including code comments and wiki articles should be written in a concise manner, with no grammatical errors. Where feasible, writing conventions should conform to the suggestions in Software Engineering :

Modern Approaches - 2nd Edition by Eric J. Braude and Michael E. Bernstein; ISBN: 978-0-471-69208-9.

Coding Conventions: The following coding convention should be used:

- Class names should start with an uppercase letter and multiple words should be separated with an underscore (e.g. Class_Name). All class methods should be entirely lowercased and named to clearly indicate their function, preferably including a verb. Avoid using overly long and verbose names.
- Variables should contain only lowercase letters, use underscore separators, and be reasonably named to indicate their purpose and contents. Very short, non-word variables should only be used as iterators in for() loops.
- Use only one statement per line
- Braces should always be included when writing code using if, for, while etc. blocks even if the braces could be omitted.
- Braces should always be placed on a line on their own. They should also align properly so a closing brace is always in the same column as the corresponding opening brace.

Metrics: At least three metrics will be maintained for every process, and every document.

1. time spent by individuals on subtasks.
2. quality self-assessment on a scale of one through ten, approximately in a bell-shaped distribution. Self-assessment scores will not be used for the evaluation of personnel by management. Failure to produce them may be negatively impact the evaluation of an engineer by management, however.
3. number of defects per unit (e.g., lines of code).
4. number of deviations open vs number of deviations closed

Quality goals for delivered products are as follows.

Detected within 2 months of delivery:

Requirements: no more than one minor defective detailed requirement per 100 requirements.

Design: no more than one minor defect per five diagrams;

Pseudocode: no more than two minor defects per thousand lines

Code: no more than two minor defects per KLOC (thousand lines of non-commented code)

The actual metric data from this project are to be reported as appendix X to this document.

SQA is responsible for reporting these measurements to the Project Manager on a bi-weekly basis.

6. Reviews and audits

6.1 Purpose

The purpose of the review process is to formally review the product and identify any defects as early as possible. Reviews carry this out in a scheduled and thorough manner. Audits do so on the basis of random sampling with short notice.

6.2 Minimum requirements

Refer to the SPMP for the schedule of reviews described below.

6.2.1 Software requirements review

This is a walkthrough of the requirements document in the presence of the entire team. The review will be lead by the project leader. It is expected that the requirements will not have been inspected prior to this review. This review is not intended to replace inspections of the requirements. The requirements leader (see SPMP) will be responsible for seeing to it that these inspections are carried out.

6.2.2 Preliminary design review

This is a review of alternative architectures with the entire team. The review will be led by the project leader or his/her designee. It is expected that the team will provide feedback, which will be reflected in the final design. The alternative architectures will not have been inspected prior to this review. The design leader (see SPMP) will be responsible for seeing to it that this review is carried out.

6.2.3 Critical design review

This is an inspection of the proposed architecture, in the presence of the entire team. The design leader will be responsible for seeing to it that these inspections are carried out. The architecture will have been inspected prior to this review. If possible, the architecture will be decomposed into detailed designs of its parts, and these will undergo separate critical design reviews.

6.2.4 SVVP review

Since V&V is to be conducted by an independent team, there will be no review of the SVVP Plan by QA.

6.2.5 Functional audit

Prior to delivery, the project leader shall be responsible for checking that the product being delivered satisfies the requirements in the SRS. Where exceptions are necessary, the project leader shall obtain the prior consent of his or her management to allow the delivery. He or she shall communicate these exceptions to the customer by appropriate means, including a "readme" file, and a cover letter citing this file.

6.2.6 Physical audit

Prior to each delivery, the QA leader shall be responsible for checking that the physical software and its documentation designated for delivery are indeed delivered.

6.2.7 SCMP review

The QA leader shall review the status of CM on a monthly basis in a manner independent of the procedures specified in the SCMP.

6.2.8 Post mortem review

The iTeam shall conduct post-mortems of all phases, in order to provide a log for potential future projects. These will include reviews of the project phase just completed, as well as the QA process itself. QA leader shall takes notes on a process improvement report for every phase, and for the QA process itself.

6.3 Inspections

All artifacts of the iBudget project will be inspected by all team members upon their completion and any issues found should be fixed within 10 days and resubmitted for follow-up .

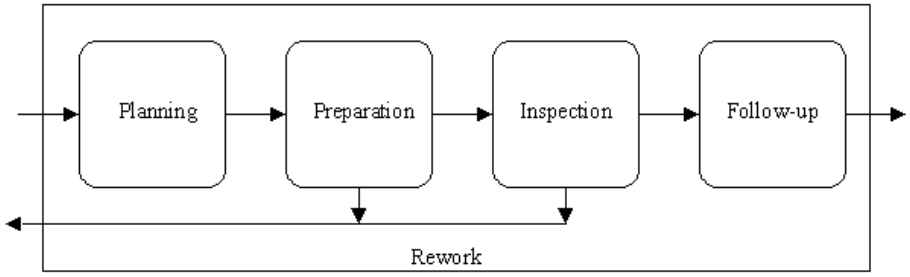

figure 1: The Inspection Process

7. Test

For the first three iterations, the QA leader will perform all of the Quality Control (QC) functions. The developers on the iTeam is responsible for testing individual methods and combinations of methods in a class ("unit testing"). The quality leader is responsible for all other testing. Refer to the Software Test Documentation for details on testing iBudget.

8. Problem reporting & corrective action

The Problem Reporting Form to be used by the iTeam in response to a software problem report generated by QA is shown in figure 2.

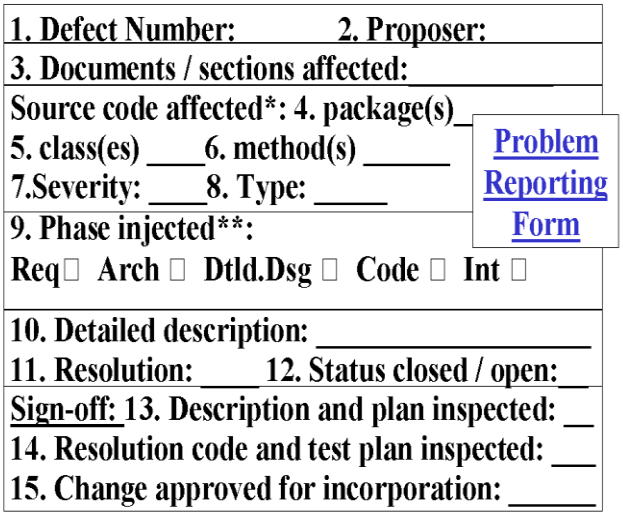

figure 2: Problem Reporting Form Example

To use this form, the members of the team should fill the describeDefect form on directory QA on the Google Code repository. The values for severity are as follows.

- Major: results in a requirement not satisfied
- Trivial: will not affect the execution of the application or its maintenance
- Minor: neither major nor trivial

The members of the team are not encouraged to create defect reports for trivial defects, but to send e-mail to the leader of the affected section who will most likely to be in a position to repair the defect or to send it to the member who is most qualified.

The values for documentation defect type are missing material, unclear, ambiguous, incomplete, redundant (within or between documents), and contradictory.

The values for code defects type are syntax, logic, data (i.e., allows a wrong variable value), insecure (allows unacceptable security breach).

When a member of the team resolves a defect, he must put the defect's status to closed. The QA leader must check the filled forms in the QA/Defects folder on a weekly basis and move the ones with a closed status to the QA/FixedDefects folder.

When defects are not fixed within a certain period of time the team leader is notified by the QA leader and has to make the appropriated decision.

The QA leader must either write or assign another member to write a weekly report about the defects (most important defects, global situation...).

9. Tools, techniques and methodologies

SQA techniques include the auditing of standards, requirements tracing, design verification, software inspections and the verification of formal methods. The SQA tools consist of mainly checklists. Checklists include the following.

- Review checklists are used at formal meetings, for document reviews and for inspections.
- Checklists will be used for verifying the quality of the following activities and documents: Preliminary Design Review, Critical Design Review, Test Readiness Review, Functional Configuration Audit, Physical Configuration Audit, SRS, SDD, SPMP and Software Development Folders.
- Separate checklists and forms are used for software audit purposes

10. Code control

The methods and facilities used to maintain, store, secure and document versions of completed code during all phases of the software life cycle are specified in the SCMP. The SQA team verifies that the code control procedures specified in the SCMP are followed.

The QA team will give out formal tests in a checklist for the implementing team to execute to check if their code respects the quality standards. The developer will have a few days to fill the checklist and give the results to the QA leader.

If the results are not conclusive, the QA leader will work with the developer to change the code that has not respected the quality standards defined before.

11. Media control

The SQA team verifies that the software media are built and configured per the SCMP and that authorized changes have been installed and tested. In addition, the SQA team verifies that the software media are duplicated using only the procedures identified in the SCMP. The SQA audit reports for media control are intended as further evidence that QA procedures have been followed.

The I iBudget project will exist solely on member hard drives and in the cloud hosted at code.google.com. Physical media will not be utilized.

12. Supplier control

For this project, there will be no suppliers to deal with. The software used is Open Source and can be downloaded on the Internet freely.

13. Records collection, maintenance & retention

The SQA records collected and archived shall include:

- task reports
- anomaly reports not handled by the regular problem reporting mechanism
- memos, including recommendations to responsible parties
- logbooks of SQA activities,
- audit reports,
- signed-off checklists from reviews and audits
- signed-off checklists from defects handling, formal tests

Besides verifying the archive procedures specified in the SCMP, SQA shall separately archive its own records at least once a week. These records are retained throughout the operation and maintenance phase.

14. Training

The QA leader will do a quick presentation of the metrics to be used, the forms to fill when finding a defect. At the beginning of each activity, the QA leader will also inform the team about the quality aspects of the activity.

At the beginning and during the development, the QA team will expose examples of formal tests that have to be done at some point of the development in order to show the developers how to execute them.

15. Risk Management

The QA leader should identify risks as early as possible. The procedures for risk management are specified in section 3.3 of the SPMP.