sqp\_notes

4.1

1. Comment on the division of defect handling: prevention, reduction and containment. Describe techniques in each of the three areas.

Defect prevention: try to solve the problem at its root, educate people using right methodology and tools effectively to detect and avoid those errors, train people with product specific knowledge for knowing its business value, train people with software development knowledge for gaining the benefits of development techniques, train people specific knowledge of development process for cooperating well in the collaborative environment.

Defect reduction: try to remove the defects after detecting it. The techniques of detecting defects includes static and dynamic verification, static analysis, such as code inspections done by people or tools.

Defect containment: try to enable the software capable to work after defects occurred, and reduce the impact, such as extra functionalities or mechanism to restore the normal functionalities from failure.

2. Give examples about each of the four areas in which education can be used for defect prevention.

Knowing the detail of business product could help improve the assets value of the product, provides new features and improve existing features.

Knowing product Architecture and relevant techniques used in the development, could help developer prolong the product life while under the changes.

Knowing tools and methodology in the development could increase the productivity of developers, reduce any repeatable work may cause human error.

Knowing development process could help developer understand responsibility of each position and task in the development cycle, know theirs role in order to control their work better.

3. Comment on other defect prevention techniques.

Good programming practices: know what the code block must perform; brief and appropriate comments for what code does; keep code simple and increase maintainability of code with design pattern.

4.2 Notes on Quality Assurance and Control

4.3

1. Write one action devoted to “prevent” errors deriving from the lack of expertise of team members.

It depends on what area of expertise is lacked. For a specific area of product, a short course and training, could be provided by some experts. However, for development techniques and process knowledge, it would be better provide a longer training before the development, since they are fundamental. For methodology and tools, some senior developer can give some appropriate instructions before or during the development.

2. Write one action devoted to “detect” faults related to performance in your product.

Daily build the project and run tools to measure the certain performance relevant fields, such as memory usage, CPU profile, response time of some important service etc. Period code and design inspection in case some faults not detected by the automation process.

3. Can you think of an instance in which an initially detected defect is then ruled out as not a defect upon closer inspection?

An initially user action would cause the system saving some privacy data in the unprotected storage unexpected, during the inspection of the relevant function, a configuration option was found to turn off this saving feature, which was initially created for testing purpose.

4. Give an example of how would you implement one system test for your product.

System testing is to detect defects both within the integrated software units and also within the system as a whole.

1. Referring to the specific user requirement documents, list the changed or new features would be added into the system.

2. Design the system requirement test cases, which is to evaluate the result of system features.

3. Put the system requirement test cases into the test environment, prioritize the cases

4. Design the functional requirement test cases according to design documents of system, conversation between developers if some crucial functions or services being affected.

5. Add new cases or change the previous test cases if some functions behavior being removed or changed.

6. Put the functional requirement test cases into the test environment, prioritize the cases.

7. Prepare certain amount of test data, in order to cover all the cases.

8. Run the system requirement test cases, log the system data

9. Run the functional requirement test cases, log the system running data and functions outputs

10. Generate reports for both tests' results

4.4

1. What is the purpose of the IEEE 730-2002 standard? (SQAP--Software quality assurance plan)

It is to provide uniform, minimum acceptable requirements for preparation and content of SQAP, detail how to prepare SQAP including any type of information would be needed. It lists the name(s) of the software items covered by the SQAP and the intended use of the software. It states the portion of the software life cycle covered by the SQAP for each software item specified.

Short: The purpose of the standard is to describe what a Software Quality Assurance Plan is, and the type of document that is required to describe such plan.

2. Choose one of the documents suggested by IEEE 730-2002 and describe its content

Software Design Description (SDD): A description in blocks of the intended implementation to comply with the software requirements, describe the components and subcomponents of the software design, including database and internal interface, should be expanded in details afterwards for facility the implementation. Its content contains the followings:

Identification the Date of issue and status, scope of software, issuing organization, responsibility or copyright information, references and context. Identify the design stakeholders for the design subject. Identify the design concerns of each identified design stakeholder. Address each identified design concern. Organized into design views: contain design elements, and different design concerns (functionality, reliability, performance and maintainability). Design viewpoint: functional views of system (such as functions, input/output), and also should select design languages for viewpoint specification. Design overlays: to represent additional information which only relevant to one design viewpoint. Design rationale: lists the reasons of designing the system and how to make those design decisions, includes any design issues and options being considered.

Software Requirement Specification (description): a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform with respect to functional and non-functional requirements. A set of use cases may be included to help clarify the specifications.

Verification and Validation plan: describe the processes that will be implemented to verify and validate a specific software application. This plan is used to strive to ensure that quality is built into the software and the software satisfies user requirements. In other words, it ensures that the project team is building the right project and building that product correctly. Tasks such as code analysis, inspections and testing and test result reporting are generally found within the detailed document.

Or:

The verification plan will define the documents that will be written to verify that the software implemented works as expected. This will include a test protocol, test results, test analysis and report, justifying or explaining the action to be taken on any found failures. The validation plan defines how the implemented software will be tested to prove to match what was requested by the customer. This will include incremental releases, a traceability matrix matching the requirements to their implementation and a review of the customer’s satisfaction at each release stage.

The user documentation: the data control inputs, input sequences, options, program limitations, and all other essential information for the software product. All error messages should be identified and described. It guides the users in installing, operating, managing, and maintaining (does not apply when modifying software source code) software products.

Software configuration management plan: manage software and hardware environment that host the system; identification of system configuration and baselines; control the changing of configuration baseline; store the information of the status in the development process; manage the process and tools for building different version of the system; process of auditing configuration.

3. Choose one of the "review processes" described in the IEEE 730-2002 that you think applies to your product and describe its details

Detail Design Review (DDR): determine the acceptability of the detail software designs as depicted in the detailed software design description in satisfying the requirements of the software requirements description. 2-3 hour block schedule and invite participants including some product field experts. Have the requirements specification ready. Evaluate designed functions and others supporting functions, see whether produce the desired effects. Evaluate any changes in the architecture of system, spot any issues might affect the operations of system. Take account the security issues, analysis the factors might affect. Evaluate the standard tolerances of the tests.

Short: undertaken to determine how acceptable and appropriate the detailed software designs are in satisfying outline requirements.

Software specifications review: assure the adequacy of the requirements stated in the SRS. Essentially this review ensures that the requirements that have been gathered for the system are both appropriate and valid for the system. (The Software Specifications Review outlines the process that assures the requirements stated in the SRD are valid, and thus the blocks outlined in the SDD are thus valid.)

Architecture design review: evaluate the technical adequacy of the top-level design of the software as depicted in the preliminary software design description.

Verification and validation plan review: evaluate the adequacy and completeness of the verification and validation methods defined in the verification and validation plans.

Functional audit: prior to the software delivery to verify that all requirements specified in the SRS have been met.

Physical audit: verify internal consistency of the software and its documentation, and their readiness for release.

In-process audits: verify the consistency of the design: code and software design document; interface design; design implementation and functional requirements; functional requirements and test description.

Managerial reviews: periodically to assess the execution of all of the actions and the items identified in the software quality assurance plan.

Software configuration management plan review: evaluate the adequacy and completeness of the configuration management methods defined in the plan.

Post-implementation review: at the conclusion of the project to assess the development activities on that project and to provide recommendations for appropriate actions.

User documentation review: to evaluate the adequacy (e.g., completeness, clarity, correctness, and usability) of the user documentation.

4. Write five features in your product that are related to any of the following aspects of the ISO/IEC 25010:2011 standard: time behavior, resource utilization, interoperability, learnability, user error protection, fault tolerance, maintainability.

Learnability: people can learn the basic skills for handling chrome in short time, from online document in the help features.

Time-behavior: reaction time of loading pages normally less than hundreds of ms.

Interoperability: run on almost OS platform, by using google account in the chrome, could store the data in the cloud, also exporting bookmarks can import into some other supporting browser, extension of chrome enable the possibilities of exchanging information to other systems by external developed features.

Fault tolerance: show a crashing page after crashing occur on the loading pages

Recoverability: previous pages could reopen after crushes

Maintainability: the system modularized, such as plugin, UI, messages subsystem, extension, security

Resource utilization: the memory used by each thread keep under 50MB

User error protection: can recover the user bookmark from the cloud after accidental deleting

Confidentiality: user could only access their data once they login

Portability: can be installed on various type of operating system, can be installed with single installation file, no extra environment setting needed, and can be uninstalled by standard program remover or just deleting its installation directory

4.5 The IEEE 730-2002 and ISOIEC 25010 2011 standards for Software Quality

4.6

1. Review IEEE 730-2002 and its most important parts. What is the purpose of such document? Why is it useful? Which role do you see it playing when a company provides software to be used within another product (a library, for example)?

Its most important parts: documentation

Purpose:

a) Identify the documentation governing the development, verification and validation, use, and maintenance of the software.

b) List which documents are to be reviewed or audited for adequacy. For each document listed, identify the reviews or audits to be conducted and the criteria by which adequacy is to be confirmed, with reference to reviewing section of the software quality assurance plan.

It maintains the adequate documents for allowing further maintain. For using a library in another product, user documentation can provide the usage of the library, and also design descriptions describe the crucial blocks of it, developers could understand the library without looking the whole block of codes. Documentation acts as a media for delivering enough information to relevant people at any time they want.

Explain which document you think is special to your product.

User-documentation should be the most important for sweethome3D. It provides information of target users, installation and updating procedures for the software at different platform, and guides of manipulating the basic function menu.

Comment the different “review processes”.

User documentation review should be the most crucial task, since this software is intended to be used in the interior design, which each function menu should be documented as thoroughly as possible.

2. Regarding the ISO/IEC 25010:2011: Review the quality in use model. Review the quality model, its categories, and provide examples of the aspects to include for your product.

Effectiveness: the software should simulate the 2D and 3D view of home plan and provide corresponding functionality of creating, modifying, and deleting furniture in the home, in addition search furniture in the local model storage. It 2D and 3D view should be export to different resolution, in order to use in different document. It should also a short recording of showing each part of home plan.

Efficiency: the software should handle the home plan manipulation within 10 seconds regarding that a home contain no more than 100 furniture. In addition, it should provide automatic saving for facility user in case they forget to save their work.

Satisfaction: functionality should sit in the top menu bar, and categories into appropriate function group. User should find menu function easily with no knowledge of interior design.

Context Coverage: this software should publish on the web and desktop with trial version. It should support mainstream operating system, such as Mac OS X, Windows platform, Linux.

Freedom from risk: this software should incur not any environmental, health and economic risk at all to user health. It should be a pure interior design software with no extra function of the other field.

System quality

Functional suitability: this software should provide the necessary function for interior design, such as a data storage for wide range of furniture model and allow the possibility of extensions, draw the 2D region diagram for your home, place furniture model in the home, and generating plan graph and 3D view in different angle of the home.

Performance efficiency: the performance is less concerned issue for this software, though it should response user action less than 10 seconds even handling dozens of furniture model. It allows crash sometimes since it mainly for desktop user, but it should recover the data at following startup. It should handle at most 5 home plan at same time.

Compatibility: it should supports different operating system, such as Max OS X, Microsoft windows, Linux. Its model storage should compatible with OBJ, DAE, Max3DS, Lw3d format model file.

Usability: it should provide guide for manipulating the function menu, it should be able to undo or redo any steps of the modification function. It should place summary function par tab on top of the software.

Reliability: it should be open at any time as operating system is up. It should be able recover the data from previous crash.

Security: it should be assessed by operating system user with normal user privilege.

Maintainability: it should be developed under MVC Architecture, separating function controlling logic, menu displaying, and appropriate defined model class of interior design into different modules. Each controller should be self-testing. It should have interfaces for external plugin integration.

Portability: the installation file should be self-contained with any dependency libraries. It should be not effecting with operating system upgrade.

4.7

1. Review the definition of levels of testing

[UnitTest](http://c2.com/cgi/wiki?UnitTest). It is preferable to find and repair defects at the developer's desk before the code is committed and then to find them during integration testing. Integration testing aggregate units/component and apply set of test cases to find any functional, performance and reliability issues. Perform system and acceptance testing, including in an environment which simulates that of the customer, and by testers who can simulate the behaviour of the end user. Should be obvious, but still too many managers (and engineers) consider testing to be overhead.

4.8

1. Think of the quality goals you would like to have in your product. State five of them explicitly together with the procedure to measure its compliance. For example, if reliability is one goal, state how you plan to measure it.

Portability: install the software on the windows, Linux and MAC OS X successfully, ensure the online version of the system could run on almost browser, such as Firefox, chrome, safari etc. Test the main functions of the system on the web, compare the result with the desktop version.

Usability: measure the time taken by users follow the user's guide, and successfully create a model in the tutorial exercise.

Reliability: it should be not fail more than 10 minutes per month during working hours.

Maintainability: react to or fix a bug within a week.

Testability: it provides certain amount of unit test cases for development, log the system running data in files or servers and have predefined expected results for running those test cases

2. Quality is achieved in certain part by reviewing the multiple steps during the development process. Describe what you need to deploy, how you would implement, and what the objectives of the following procedures are: design review, code inspection.

During development process, people should periodically evaluate the certain output of development tasks, such software requirements specification, software detail design ETC to ensure those documents follow standard, and review the design to ensure meet requirements from users and detect any design error or misunderstanding of user requirements before each iteration of implementation.

During the implementation, code inspection and code-walkthrough after each iteration of implementation task, try to spot any logical bug, not good coding style and any missing test cases before deploying into testing environment. During the testing stage, control the progress of testing, periodically report the results of system test cases, the amount of detected bug, progress of fixing the bugs, and the completion time of test cases.

3. Software has to be tested. But the testing can be done at various levels. Provide a definition of three levels of testing and the type of procedures included in each of them. Give examples related to your project.

Unit testing: usually done by programmer upon implementation, ensure units/components work function properly with respect to the detail design, for java based software, the usual technique is Junit test framework.

User acceptance testing: generally host by end-users after a tested stable version of the software, users check functionalities meet expects and needs.

Regression Testing: testing that previously implemented functionality still functions as expected given newly added features. This will include steps that the tester should follow and results that the tester should observe when done correctly. The tester will then rank the test as a Pass, Failure of the test, Failure of the result, Failure of the specification.

Integration Testing: is the phase in software testing in which individual software modules are combined and tested as a group, takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing, to detect any inconsistencies between the software units that are integrated together.

System testing: usually done towards the end of a project, integrated system to evaluate the system's compliance with its specified requirements, include functional requirement and system requirement. Test the design of system and behaviour of target users.

4. Enumerate the tools you need to deploy a quality plan. You must include all tools such as management support, change-control, testing, etc.

HP quality center can store requirements of current version and link them to the test cases.

It has a report showing the coverage for user requirements, ensure all the requirements are tested.

Git code repository using branch to manage the different release version of current developed system, such as branching to development branch, test staging branch, UAT branch.

Quality center can manage the testing progress by progress reports, including report the complete percentage of each tested requirements, the detected bugs with priority, and status of each test cases.