CMSC 435 - Software Engineering
Spring 2010
Quiz 3

Name: SOLUTION

Answer the questions below. Be clear and concise in your answers, writing in complete sentences and providing examples for support as appropriate. No answer should take more than three sentences.

A friendly reminder: Please keep your quiz responses to three sentences per question - not just to make it easier on the TA :), but because explaining these concepts concisely is an important skill for a practicing software engineer!

1. (5 pts.) Milestones and daily builds are two of the cornerstones of the "synch-and-stabilize" approach and other similar, more flexibile models of software development. Define milestone and daily build, including the role of each in synchronizing and/or stabilizing.

   Daily builds involve compiling and automatically testing the system in order to synchronize parallel development teams and determine what works and what doesn't. The goal of the daily build is not to fix problems, but to identify them.

   Milestones are incremental versions of a product used to stabilize development by parallel teams. If you added that milestones synchronize, I didn't count off, because they are associated with a specific point in time. However, their main goal is stabilizing development.

   You were asked specifically to include the role of each in synchronizing and/or stabilizing. See last slide of set 2.

   Points:

   - +2 for correctly defining daily build
   - +2 for correctly defining milestone
   - +1 for correctly identifying stabilization and synchronization roles

   Samples:

   - "Milestone (is) a set progress point to achieve ... parts achieved at each milestone create a stable partially complete product."
   - "A daily build synchronizes the work of multiple teams on a project while a milestone stabilizes the work into a more polished build."

2. (5 pts.) We can easily verify the fulfillment of a functional requirement, but is it possible to verify non-functional requirements? If so, how? If not, why not?

   It is possible to verify non-functional requirements by adding quantifiable measures of reliability, performance, usability, etc. The key is that we must have some measurable objective criteria. Some of you focused on developer requirements here, which is OK - the answer is still the same. Beta and usability testing was a common answer and received majority credit, but remember that user feedback is still subjective and only verifiable if we have some objective criteria/measure associated with it.

   Points:

   - +2 for identifying that NFRs can be verified

- +2 for explanation of how
- +1 for correctly identifying the need for objectivity/measures

Samples:

- "If a requirements is correctly specified as a verifiables non-functional requirement, then we can verify it by using its numerical constraints and comparing them to real world results."
- "This [verifying non-functional requirements] can be done by ... timing a program's run, checking to see if the system has certain properties, or some other kind of constraint test."
- "by interacting with customers and users through Beta releases and demonstrations. This type of testing involves gathering statistics on the user's opinions." (Note: Technically, testing involves actually executing code. Non-functional requirements verification falls under the umbrella of "quality assurance," which includes testing and many other techniques, such as the empirical study described here.)