Up the Waterfall:
Costs of Delayed Validation

by M. E. Kabay, PhD, CISSP-ISSMP
Associate Professor of Information Assurance
School of Business & Management
Norwich University, Northfield VT

The Spring 2011 session of IS342 Management of Information Assurance<
http://www.mekabay.com/courses/academic/norwich/is342/ > in the School of Business and
Management at Norwich University has opened and there are bright-eyed undergraduates
learning about the topics and keeping me on my toes. This semester I am responding to
suggestions in the course reviews from previous semesters by eliminating Death By PowerPoint
(type “death by powerpoint” into the search field of Google Images<
http://www.google.com/imghp > for some hilarious cartoons and illustrations of that concept).
This semester, the PowerPoint files are available as usual, and students received 6-slide-per-page
printouts as review notes, but I’m not going to lecture at them any more. Instead, we are having
vigorous classroom discussions – which are more fun not only for the students but for me!

In the introductory discussion of software quality assurance (SQA) on January 25, 2011, I
mentioned something to the students that I have long argued: “Since the cost of rectifying errors
grows by about ten times with each stage of development, it's sensible to incorporate SQA at
every step of the system development life cycle.”<

During the class, we discussed this concept using an example. I hope readers will find it
convincing.

We started off with a reminder of the steps of the system development lifecycle (SDLC)<
http://www.computerworld.com/s/article/71151/System_Development_Life_Cycle > and then
imagined a scenario in which Albert Analyst starts the requirements definition by chatting with
Barbara User. She tells them that he has to plan for 12,000 transactions per hour as he and his
colleagues design the new system they’re planning. Unfortunately, Albert writes down 1,200
instead of 12,000. However, Albert checks with Barbara before he leaves to be sure that she
agrees with everything he wrote down. They catch the mistake and fix it in a matter of seconds.

Personally, when I am doing consulting work, I ask the user if I can connect my portable
computer to the user's display so the user can see everything I type as I'm typing it. I also record
the conversation; today, digital recorders< http://www.staples.com/Digital-Recorders-Recorders-
Transcribers/cat_CL140515 > that fit in a shirt pocket and cost relatively little can hold
hundreds of hours of recordings that can be uploaded to one's computer for further processing.
Before I leave at the end of an interview, I either print my notes for the person I interviewed or
send them by e-mail so that they can add more details and make corrections.

But what if Albert and Barbara failed to catch that tenfold error in the expected transaction
volumes? Well, no fear, they can always catch the mistake once the requirements are formally
put into a document for the user signatures. Yes, but at that time it will take more like several
minutes – let's say 10 minutes – to spot the error, fix it in the file, and redistribute the updated
file.
You can imagine to the the rest of the story: if the error gets all the way to system design, it might have a significant effect on technical decisions such as whether to include an index field in a data table. Fixing that kind of mistake could take an hour. If the mistake got all the way to coding in the implementation phase, it's conceivable that fixing that level of error might take several hours – maybe a day. And if the error makes it all the way through into production, the consequences could be serious indeed, with potential saturation of a system that was never intended to provide service levels ten times higher than expected.

The class discussion then turned to a reminder of the distinction between validation and verification in SQA. Checking with Barbara User to see that Albert Analyst got the information right is an example of validation: the requirements match what Barbara wants. (The question of whether what she wants is what she needs is a different issue.) Checking the code using software testing procedures to see that the new system can meet the service level agreement (SLA) among the developers, production, and the users is an example of verification.

We should use constant verification to support validation in the SDLC. In my opinion, we should be equally vigilant in all of our business communications. Take the time to be sure that you got it right during a discussion with your colleagues. Take notes and share them; summarize phone conversations by sending an e-mail with clear action items and who is going to do what by when. Check numerical information such as budget targets; be sure you know exactly who should be on an e-mail distribution list instead of guessing. All such verification procedures help to validate our plans. A

* * *

M. E. Kabay, PhD, CISSP-ISSMP, specializes in security and operations management consulting services and teaching. He is Chief Technical Officer of Adaptive Cyber Security Instruments, Inc. and Associate Professor of Information Assurance in the School of Business and Management at Norwich University. Visit his Website for white papers and course materials.

Copyright © 2011 M. E. Kabay. All rights reserved.

Permission is hereby granted to Network World to distribute this article at will, to post it without limit on any Web site, and to republish it in any way they see fit.