IA Includes Software Development

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

Sometimes we lose sight of the wide reach of information assurance (IA). In class discussions in the Management of IA course< http://www.mekabay.com/courses/academic/norwich/is342/index.htm > at Norwich University< http://www.norwich.edu >, students recently discussed how software development and quality assurance play a role in IA.

One of the areas that our students study in their software engineering courses< http://www.mekabay.com/courses/academic/norwich/is301/ > is development strategies< http://www.mekabay.com/courses/academic/norwich/is301/04_ch4.ppt >. The traditional system development life cycle (SDLC) puts a great deal of time and effort into the project definition phases; systems analysts must interact with users, encourage them to define their needs, define functional requirements (these two phases can be called the requirements elicitation), get the functional specifications approved by the users, and then design and build the systems to meet those requirements. The SDLC includes system testing and system documentation.

An alternative is agile development< http://www.allaboutagile.com/10-key-principles-of-agile-software-development/ >, which can include rapid application development (RAD< http://www.cs.bgsu.edu/maner/domains/RAD.htm >) and joint application development (JAD< http://www.umsl.edu/~sauterv/analysis/JAD.html >). In these methods< http://www.mekabay.com/courses/academic/norwich/is301/23_ch17.ppt >, analysts follow the requirements elicitation phase by model-building (prototypes). Prototypes< http://csweb.cs.bgsu.edu/maner/domains/Proto.htm > can be created with simple development tools that allow users to see partial or even simulated user interfaces with some of their required functions enabled by fourth-generation languages (4GLs) or application generators. Faced with a model or simulation of functions that can meet their needs – at least, in part – users often respond with corrections and clarifications, thus helping analysts understand and document user needs more thoroughly. Analysts can thus generate improved definitions of the functional requirements and support designers and programmers in building systems that better meet the organization's needs. In addition, agile development methodologies can provide a significant degree of preliminary functionality early in the development cycle, immediately meeting some of the most common needs within weeks or months instead of forcing everyone to wait for all the functionality to be released in the months or even years typically forced by the traditional SDLC.

Decades ago, I defined a rule to describe the effect on users of seeing a model that simulates part of a new system: **the availability of a tool changes the perception of the possible**. I defined that principle partly because of an incident that occurred in 1986, when I was hired to help reorganize the information technology operations of a clothing factory, with special attention on performance and technical support. As I was walking through the office with the vice-president (VP) of information technology (IT), I noticed something unusual off to one side of the office: there was an employee with a thick stack of 132-character by 88-line computer printouts working with a hand calculator. I asked the VP if we could go talk to him and then asked him politely what he was doing. He said he was calculating subtotals based on the printouts. "Ah," I said, "and how often do you do that?" He said he did it every month. "How long does it take you?" A few hours. "And how long have you been doing that?" About three years. Finally, I asked, "Have you ever asked your IT department to include the subtotals in your report?"

The employee looked at me in blank astonishment.

"They can DO that?!?"

One point of the story is that everyone in IT has to focus on user needs and actively interact with users not only to ask them about their perceived needs but also to stimulate cooperation in thinking clearly about all the functional requirements of IT systems, whether they are currently recognized or not. IT staff are supposed to be the experts who are aware of requirements that the users may never have thought about; one of the key areas where users who are focused on their own work either forget to define requirements or are completely unaware of them is security.

Security requirements – protection of confidentiality, control, integrity, authenticity, availability and utility of information – *must* be included in the functional specifications of *every* IT system. Ignoring security considerations for software is like ignoring safety considerations in automobiles: it's unprofessional and potentially dangerous. The Risks Forum Digest has documented thousands of systems over more than two decades in which security considerations were either ignored, poorly defined, or poorly implemented. The year 2000 (Y2K) problem illustrated far-reaching consequences of failure to plan for continued utility and integrity in computer software – thousands of programs all over the world were at risk of failing because they used two-digit date fields that assumed that all dates would be in the 20th century only.

By now, most readers are surely aware that security must be built into systems from the very first, not simply added in as afterthoughts.

As programmers build the systems that meet user requirements, they must apply thorough software quality assurance (SQA) techniques. Many of the problems that SQA tries to find include errors that can profoundly affect security. A brief list includes

- Initialization Errors
- Logic Flow Errors
- Calculation Errors
- Boundary Condition Violations
- Parameter Passing Errors
- Race Conditions
- Load Conditions
- Resource Exhaustion
- Interapplication Conflicts
- Other Technical Errors
- Regulatory Compliance Considerations

For details of these issues, readers are invited to download the PDF or PPTX versions of our class notes, which are based on Chapter 39 of the *Computer Security Handbook*, 5th Edition< http://www.amazon.com/Computer-Security-Handbook-2-Set/dp/0471716529 >, "Software Development and Quality Assurance," by John Mason, Jennifer Hadley, and Diane E. Levine.

In the next column, I'll review how patch management plays a critical role in information assurance.

* * *

M. E. Kabay,< mailto:mekabay@gmail.com > PhD, CISSP-ISSMP, specializes in security and operations management consulting services and teaching. He Professor of Computer Information Systems in the School of Business and Management at Norwich University. Visit his Website for white papers and course materials.< http://www.mekabay.com/ >