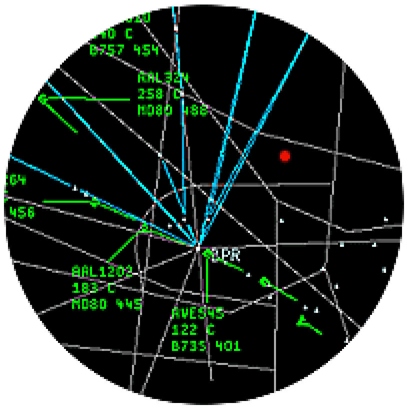

A Walk Through Mike's Code: A Case Study in Software Technical Reviews

by
David R. Luginbuhl
Department of Mathematics and Computer Science
Western Carolina University

Mike looked at his watch. 9:55! It was almost time for the meeting. He picked up his notebook and water bottle and headed for the conference room. He was anxious to see how these tech reviews actually worked, even more so since it was his code being reviewed.

As he hurried down the hallway he heard someone behind him. "Hi, Mike." He turned around and saw Beverly Anson catching up to him. "Guess we're headed for the same place. I pulled reviewer duty."

That surprised Mike. "I thought Paul and Joe were the reviewers today."

"Oh, didn't you hear? Joe got food poisoning and called in sick this morning. But not to worry; since I'm in charge of integration, I started looking at everyone's projects a week ago. I've already been through your code with a fine-toothed comb," Beverly answered.

This was not reassuring. Ever since he arrived at RadAire Systems six months ago fresh out of school, Mike had thought that Beverly was a bit too detail-oriented and "by the book." On the other hand, Mike considered himself more of a rebel, willing to try new things and push the boundaries. He wasn't sure Beverly would understand some of the choices he'd made.

They reached the conference room, took their seats, and waited for the meeting to begin. "OK. This morning, we're looking over one of the Handoff programs—EdgeAlarm. Mike Prager is the author," Annette Rossovich declared. Annette was the facilitator for this review. Mike's code was part of the new Air Traffic Control system, ATC-III. "Mike, do you want to describe your code in general?"

"Well, it's pretty straightforward, I think," Mike said. "This class is responsible for detecting proximity to the edge of the tracking radar's range and notifying the system to prepare for handoff. It's so simple, I'm not even sure why it was picked for review."

Paul Chin—one of the reviewers for this session and one of the more seasoned engineers on the project—chimed in. "Mike, everyone gets to run through this drill at least once per project. Besides, this is an important piece of the puzzle. This code has to be compatible with my search methods. I'd hate for us to bring down the whole system during integration testing because of an incompatibility problem."

Mike tried not to take Paul's last comment personally. He knew full well his code had to work with everyone else's. Furthermore, he knew that everyone's code was reviewed at some point—the instructor at his two-day tech review orientation course had made that abundantly clear. He continued: "OK. Anyway, I found a pretty cool way to determine proximity without having to directly ping the airplane object for its current location...."

"So I noticed," Beverly interrupted. "Mike, the standards call for straightforward approaches and reuse of code wherever possible. Otherwise maintenance becomes a nightmare. Did you even look at John Drake's getAircraftLocation method? It's in the reuse library, it's tailor-made for this application, and everybody understands it."

"OK, people, I think we're getting a bit ahead of ourselves here," Annette finally interrupted. "Let's stick to our checklists and don't get personal. Keep in mind that this isn't a review of Mike. Stay focused on the code." The level of animosity was increasing, and Annette would have to work hard to keep this review productive. "We'll let Mike explain his new approach to location determination when the time is right. I'm sure he can make a good case."

"You bet I can!" Mike fumed. He took a swallow from his bottle of water and tried to calm down. Since he had no choice but to be there, he might as well cooperate with Annette, annoyed as he was now with both Paul and Beverly.

"Mike, let's begin by reviewing your unit testing results," Annette said, trying to get the group back on track. "Take us through it. After that we'll do standards, and then we'll have you walk us through the details of the code."

Mike was calming down. "OK. Of course, I had the unit requirements derived from the detailed design. I knew which other routines I had to interact with. From this I was able to develop a series of black-box tests."

"The Software Quality guys are gonna love you, Mike. You've thought of every contingency." Finally, a compliment from Beverly, Mike thought. "How did you ever think to test edge-detection at zero altitude? That was inspired!"

"Hey, even planes that are crashing might need to be handed off. Wouldn't want to lose a vehicle's location just as it was going down," Mike said. "Besides, it's implied by several of the emergency requirements."

"And this might be one of the reasons Mike didn't use John Drake's code," Paul said. "I don't think John accounted for emergency situations in his methods."

"Exactly," Mike responded, with a sense of satisfaction. "Anyway, once I finished coding, I added some coverage tests, which I think cover the code fairly completely," he continued. "I had 80% success with tests on my first go-round, an additional 15% on round two, and finally achieved 100% on round three."

"How much re-coding did rounds one and two require?" asked Paul. "And did that result in any re-writing of test cases?"

"I had to re-do a small bit of the logic of the rangeDistance calculator, and other than that, it was a few typos here and there. Test cases were unaffected by re-coding."

"OK. Unit testing looks good. Any questions?" Annette worked to keep everyone on track. Nobody had anything else to add. "OK. Let's move on to adherence to standards. Any questions there?"

"Right. I have a question." Beverly clearly had a problem with something. Mike could tell by the tone of her voice. "Mike, your variable names are pretty generic—num1, num2, that sort of thing. These don't follow our usual naming conventions and will be another maintenance nightmare...."

"...and your question is...." Mike was right back at Beverly.

"So why didn't you use more descriptive names, using the naming convention we all agreed to at the project intro meeting?" Beverly replied. "The approach you took will make the program AND the data dictionary more difficult to read and understand."

"Look, Beverly. You've had an attitude about this code from the moment we started this review. I clearly explain all the variables in the comments." Mike had had enough. "What is your problem?"

"Mike, I could ask you the same thing. You have deviated from standards in so many places, I'm not sure how we'll ever fix this thing without a complete rewrite." Mike was right. Beverly certainly had a problem with the program.

"OK, OK. Cool heads, remember?" Annette interjected.

"Not for me," Mike replied. "I've had enough." And with that, he left, fuming.

"Well, that's interesting. Now what do we do?" Beverly wanted to know.

"I will talk to Mike after he lets off some steam," Annette replied. "But first, Beverly, why don't YOU come to my office?"

"Me? What'd I do?" Beverly wondered.

Date Posted: 08/29/02 nas