If I Only Changed the Software, Why is the Phone on Fire? Debugging Methods Revealed
By Lisa Simone, Embedded Systems Architect
March 2007 • ISBN 978-0-7506-8218-3 • Paperback • 304 Pages • $39.95

Features:
• Unique format casts the reader as "technical detective" by presenting a new mystery in every chapter
• Not another dry technical book! Conversational tone and intriguing quandaries draw the reader into the action, while teaching crucial debugging skills
• The final chapter, a summary of the smart debugging techniques introduced throughout the book, is a quick reference to help solve future problems

This new practical reference manages the unthinkable - it conveys crucial technical information to engineers without boring them to tears! In this unique book, author and expert embedded designer Lisa Simone provides the solutions to typical embedded software debugging problems from a fresh new perspective. She introduces a team of engineers who readers will recognize from their own workplaces, and then confronts them with real-world debugging scenarios of progressive complexity, drawing the reader into the “mysteries” with their new fictional colleagues, and guiding them step-by-step toward successful solutions.

CONTENTS: 1 The Case of the Irate Customer: Debugging Other People's Code, Fast 2 The Newest Employee: Learning the Embedded Ropes Through Code Inheritance 3 It Compiles with No Errors; It Must Work! Integrating Changes in a Larger System 4 The Case of Thermal Runaway: Rare Transient Bugs are Still Bugs 5 The Case of the Creeping Slider Carriage and the Case of the Hesitating Clock: Alternate Methods of Understanding System Performance 6 If I Only Changed the Software, Why is the Phone on Fire? 7 The Case of the Rapid Heartbeat: Meeting the Spirit of the Requirement 8 What Kind of Error Message is "lume Fault"? When all of the Symptoms Seem Impossible 9 When It's Not Hardware, It's Software. And Vice Versa. Blurring the Interface 10 Li Mei’s List of Debugging Secrets

Order from Newnes.
Visit Newnes on the Web: www.newnespress.com
Email: usbkinfo@elsevier.com
Real-World Bug [Location: Scotland] “Killer software” was once a joke term. No longer. During the 1990s certain British Chinook military helicopters had their avionics replaced with more sophisticated units running big hunks of firmware. A variety of problems appeared; sometimes the rotors started spinning unexpectedly quickly in what was soon labeled an “uncommanded run up.” Pilots were instructed, naturally, to hit the reset button when such odd events occurred. In 1994, an updated Chinook went down, killing the pilots and crew.

Chapter 2

The Newest Employee: Learning the Embedded Ropes Through Code Inheritance

“0h, I can’t BELIEVE this guy! This code is terrible!” Li Mei drew another red line through a software listing and grimaced. “I thought this part of the software already worked.”

Realizing she had spoken aloud, she quickly looked around to see if anyone had been standing nearby. Talking out loud was definitely a problem with cubicles.

Her first day wasn’t going as expected.

Josie had shown her around and she had met many people, including her new manager, Oscar. He had been in the middle of a big deadline and he didn’t seem that excited to see her. It was nearly time to go home and her only visitor had been the human resources manager with a stack of forms. She was alone, feeling forgotten in the bustle of activity around her.

She’d thought her first day would be more exciting. It was supposed to be her special day - her very first day of work, ever, and no one cared. Did joining the Real World mean that she was just an extra person to clean up software messes?

She flipped through the stack of documentation and the Change Request form again. The title of the CR was “Fix user button function (software)” and it was a Severity 3, Priority 2 CR. The form explained that a Severity 3 bug was expected
to be significant although limited to a confined section of the code, but Priority 2 meant that it needed to be fixed with some urgency. It was confusing having two numbers to describe the problem, but she reasoned that a less severe problem, like a misspelled display message, might be needed immediately, even before a terrible problem, such as out-of-memory, on a different project. So she'd better get moving on a solution.

Most of the documentation was hardware specifications for the narrow-beam ultrasonic transducer inside the Meter Magic electronic tape measure. Josie told her the hardware was working, but prototypes that included the hardware had been delayed. She was more interested in software specifications anyway, but there weren’t any. She would just have to figure it out from the code.

But the code was terrible!

So far, she found this Benjamin person did not like to type very much. His variable names were too short. He didn’t comment anything. The more she read, the more she thought about starting the software from scratch, but she must understand what he coded in the first place to understand what this software was supposed to do!

Oscar said something about having one week to get this working. Shouldn’t she get one of these measuring things to play with? She didn’t even know how to compile the code! The anxiety built as she searched her computer for any type of development tools. Would she be able to work in UNIX, or would she have to use PC-based tools? She didn’t even know what this Meter Magic thing looked like!

With her throat beginning to constrict, she pushed away from the computer and willed herself to stop thinking crazy thoughts. She only knew one thing to do with this horrible software, something she had learned from a very good teaching assistant in college.

She would make a flowchart.

Placing a printout of the main routine next to her (shown in Figure 2-1), she turned to face the whiteboard that hung on her cube wall. Brand-new markers filled the tray. She picked up the red one and wrote “main” at the top of the board.

Reader Instructions: Before continuing, grab a piece of paper and sketch out a basic flowchart for the function main(). If you don’t know what something does, add a box to your flowchart and assign a temporary description like “stores variable” or “performs function on variable.” As you work, do you see any problems with this section of code? List anything you find.
As soon as she began, the challenge of reverse engineering the code eased her tension and the flowchart began to take shape. First, the variable declarations looked okay. Next, she trudged through the `init()` function and found that it included all the initialization software for the hardware - the ports, the LCD display, and the calibration algorithms all lumped together in one big, long routine. It looked messy, so she decided to ignore it for the moment and concentrate on the main routine.

Everything in `main()` was inside one big do-while loop that ran forever. She drew a vertical line on the software listing linking the do and while commands together so she could see where the loop started and stopped. Next, she linked each opening and closing curly brace to encapsulate the if-statement blocks of code. All were properly matched. It would be a lot easier to understand, she thought, if Benjamin had placed the opening and closing curly braces in the same column, rather than staggered like this. Then she could match them up easier.

Inside the loop, `measure()` was called before anything else. She thought that looked odd, and speculated that the distance measurement might occur in that
function. A quick scan through the routine yielded no better prospects, so she added a box to her flowchart and labeled it “Take Measurement.” Next, the new measurement was added to an array, \texttt{dists[]}\@. Distances, she thought. And it looked like only ten could be stored.

Li Mei tapped the back end of the marker against the whiteboard as she stared at the next three lines. This must print the result on a display, and she guessed that \texttt{dmess()}\@ somehow controlled that function. She hadn’t looked at the \texttt{dmess()}\@ function yet, but the next line was a second call to \texttt{dmess()}. That was another strange thing. Wouldn’t the second call just overwrite the previous measurement result on the display? Unless it somehow queued up display messages or something like that. Well, she decided, just make another entry in the flowchart and keep going.

The next line contained a \texttt{getch()}\@. That was logical, since a tape measure should probably receive some input from the user, and she counted three commands that the software would accept: ‘r’, ‘x’, and ‘c’ - whatever they meant. She drew three new decision boxes for these commands, and finished the flowchart by drawing an arrow from the bottom back up to the “Take Measurement” box.

That was it.

Li Mei sat back down, perched on the edge of her chair and stared at the flowchart (shown in Figure 2-2). She was determined to understand this code before quitting time, and quickly realized that the ‘r’ command was easy - the user could reset the index into the array of measurements by setting the measurement counter to zero. The ‘x’ command was also pretty clear; it multiplied two of the measurements together and displayed the result. Then she frowned; ‘c’ displayed a measurement several times in a row. She couldn’t seem to crack the reason for this command and it frustrated her, but it was already after 5 o’clock and she heard people leaving for the day.

The ‘c’ must stand for Confusing, she decided, and solemnly noted “Print Confusing something” in her flowchart, closed down her computer and walked out of the building alone.
The next morning, Li Mei was well into a list of things she wanted to ask Josie. Things that she didn’t understand about this software.

She’d gone home despondent and had eaten alone, while surfing the web for new-job horror stories. They left her more depressed, thinking maybe she was expecting too much. She hoped she would not be too lonely at Hudson Technologies. She would still try her best.

Around her the cubicle area was quiet. Most people were probably working in their labs. Sticking her head out into the aisle, she looked around at her new surroundings. Back at school, she had shared space with others, so she was used to concentrating amidst the comings and goings of other students and staff. This environment seemed much more formal and forbidding.
Chapter 2

She wasn't sure what to do next with the software, so she decided to take a break and wander around. The cubicle area was large and open with street signs at major aisle intersections. Her cubicle was in the same aisle as Josie's, on Feynman Lane. She followed Feynman Lane past Hubble Avenue and Edison Avenue to the far side of the building and found herself in the marketing area. It seemed as if she had penetrated an invisible boundary; the wall color and carpeting had changed from boring beige to a deep green with maroon accents. This area had more real offices with wood doors, and a large glass-enclosed conference room with leather chairs and a large dark-wood table.

There were no more street signs.

Down the next hall she found even nicer offices. Probably the president and vice president, she supposed. She peeked into an open door and caught a glimpse of dark wood and suits as she passed, and looked backward to see a nicely dressed woman carrying folders into the conference room. She suddenly felt as if she were trespassing, and continued without slowing until she reached the end of the next aisle. Beyond a water fountain and set of restrooms, she found the back doors to the cafeteria. Breathing a small sigh of relief at the now-familiar surroundings on Hopper Avenue, she spotted Josie across the cafeteria filling her mug with coffee.

Josie waved her over. “How are you coming along?”

“Okay. I was just walking around. I like the street names.” Li Mei joined her in the cashier’s line.

“Isn’t that cool? Navigating in the cubicle farm is a lot easier with street signs. I love that we are on Feynman Lane - Richard Feynman was an awesome guy.”

“Who was he?” Li Mei considered a muffin.

“A physicist from the ’50s. He worked at Los Alamos during World War II on the bomb, and then taught physics at Princeton and Cal Tech.” She scooted her coffee mug up the line as she dug money from her back pocket. “There’s one story where he’s in the dining hall and got interested in the way that the school medallion on the edge of the plates wobbled as the plate spun. It made him think about complex motion, and he ended up with a Nobel prize. All from a dinner plate. Cool, huh?”

Josie paused to pick up a wad of singles that had dropped to the floor. “I also like that he didn’t really care what other people thought of him. He ended up having great adventures based just on his curiosity about the world around him. I’ve got a couple books about him if you’re interested.”

Li Mei smiled at her as they reached the cashier, “Okay, thank you. I would like to read about this person because we are living on his street.” She was interested, but she really wanted to change the subject.
She blurted, “I am having trouble with the Meter Magic. I have read everything and looked at the code. I think there might be many problems.”

Josie laughed. “I don’t doubt that. Benjamin wasn’t a great programmer.”

Li Mei looked up. “What happened to him? Did Oscar fire him?”

“No, but I think he was about to. Benjamin isolated himself and spent too much time trying to get his code to work. I don’t think he really liked coding. Oscar started to lean on him for not following the coding standards and skipping code reviews, and Benjamin ended up resigning.” She pocketed her change and took a drink of the coffee. “Honestly, I’m glad he’s gone. If he had been a good coder, it would be different, but he didn’t want to work with anyone else.” She shrugged and started walking. “But now you inherit his disaster.”

“I am not sure I know which parts are the disasters, so I made a flowchart to understand it. I think it’s missing things.”

“That is entirely possible. Did you check the documentation to see what the features are?”

“Yes, but it is not useful!” Li Mei felt her frustration start to rise again. “It’s all about the hardware and the ultrasonic beam. It has no software discussion.” Li Mei stuck her hands in her pockets. She knew Josie was busy with Oscar’s deadline, and the tense man hanging out around the lab door was Oscar’s manager, Randy.

“I was wondering if you could look at my flowchart and tell me if I understand it properly? When you are not busy?”

Josie looked at her watch, “Actually, I’ve got some time now.” She briefly told Li Mei of the late night that she and Oscar had the night before, and the status of the Communicator project. “I am sorry that you were abandoned; let’s take a look at it.” Relief flooded Li Mei’s face.

Before Josie had a chance to sit down completely, Li Mei was already pointing to the whiteboard flowchart and counting problems off on her fingers as she recited.

“First, I think one big problem is that the device takes a measurement immediately, before the user presses the button. And it takes measurements all the time.” She handed Josie the printout of the code. “I followed this function call, `measure()`, and it turns the ultrasound transducer on, and then waits for the measurement to be completed. That function is in another file and it is written much better with comments.”

Matter-of-factly she added, “Benjamin did not write that code.”
Josie leaned back in Li Mei’s guest chair and stared at the ceiling, amused that Li Mei was already on a first-name basis with Benjamin. With Li Mei’s level of agitation, Josie wondered if she had some other names for him as well.

Josie told her, “Some of this code was written as the hardware was being developed, so that interface function should already work. Let’s assume for now that it’s good and just worry about the main code. Tell me why you think it takes measurements all the time. What do you mean?”

Li Mei pointed to the bottom half of the do-loop where key presses were processed. “Every time the user presses a button, the program does something and then wraps around and takes another measurement, although I guess that is mainly what the tool is supposed to do.”

“No, you’re right.” Josie let the chair drop to all four feet and leaned forward to point at the whiteboard. “Look back at your flowchart - sometimes it’s easier to see the mistakes in the software that way. If the user presses the button to reset the memory, should that cause the tool to take a measurement? That kind of defeats the purpose of clearing the memory, doesn’t it? You erase memory, then BOOM! Immediately another measurement gets stored in there.”

Josie thought it was a good sign that Li Mei had generated a flowchart to understand the software. But did she know how to use it effectively?

“No,” Li Mei admitted, “it’s not good to take measurements all the time, but there is no button to just take a measurement. So this device has to take measurements at some other times.”

“No button for measurements?” Josie was taken aback, and then smacked her forehead. “It’s got a button for measurements, but how the heck are you supposed to know when I didn’t give you one of them to play with? Let me get one.”

She left and returned a few moments later with a device about the size of a hand-held organizer and handed it to Li Mei, apologizing. “We got samples last week but no one has been available to work on the software. It’s loaded with whatever software Benjamin left.” Li Mei took the device and played with several of the buttons, then leaned across Josie as she attempted to measure the area of her cube. She stared at the display quizzically and then showed it to Josie, who laughed when she saw it.

“1,220 square feet! This says you’ve got quite a mansion here - where are you hiding the rest of it?”

Li Mei allowed a microscopic smile to escape before sliding down into her chair. “See, this software is missing a lot of lines. And it has bugs.”

“It does,” Josie agreed. She paused to think. “Here’s what I think we should do. You made a good start creating the flowchart; now I want to see if you can use that
knowledge to identify as many problems in the code as you can, before we go run the code in the lab. Then I will set you up in the lab with the debugger. Deal?”

“Okay. One thing at a time. It’s a deal.”

For the rest of the morning Josie challenged her to identify everything she thought was wrong with the software. As lunch time neared, Josie reviewed the list carefully printed next to Li Mei’s flowchart.

- Measurement taken too often, and not on correct button press
- No way to take just a single measurement
- Display of product name at wrong times
- Command ‘c’ doesn’t make sense
- Code for several user buttons on device not implemented in software
- Multiply command doesn’t work right - indexes wrong?

Josie nodded her approval. “There are a couple of ways to approach this. On one hand, you could just fix the problems you found, and then add the missing code. On the other hand, there are some architectural issues here that you might want to correct first. Which should you do?”

Although Li Mei had passed the interview coding test, Josie wondered how extensive her experience was with different programming methods. Would she recognize that using a switch statement to process the user commands would be more readable and require less code space?

Josie hoped so. It was much more fun to work with someone who thought about better ways to solve problems, although many developers didn’t think that way. They just focused on getting the software to work, regardless of what it looked like.

Li Mei had been sketching blocks on her notepad. “I think the most important thing this routine does is wait for the user to press keys, and then it should do what the command is supposed to do.” She paused and brushed the hair back from her serious eyes. “I would make a new command ‘m’ to start a measurement. It should be a separate if-statement.”

“Good start. That fixes one of the big existing problems. But what about the missing commands?”

“I would make a new if-statement for each of the missing commands.”

Josie nodded. “You are on the right track. That will move all the code out of the main part of the do-while and into separate sections. However, I do see some issues with the way the if-statements are done now. Each logical is evaluated, even if the command has already been serviced by a previous if-statement.”
Josie paused to see if Li Mei would suggest an alternative, but she didn’t. Instead Li Mei said, “I don’t understand why the if-statements are bad. If it works correctly, then why should we change it?”

“This code will work correctly, but it’s not efficient and not the best solution. Think about what’s being tested by the if-statements: user commands. The user can only request one command at a time. If the command is to reset the array, the command will not magically change to another command after the array is reset, so the code should not check any other options after it finds the right one.”

Li Mei stared at her flowchart. “We could add else-statements. That would help.”

“That’s good! That would limit processing to just one of those code segments each time, but it can get messy. But I am partial to a different way of implementing user commands: using a switch statement. Do you know what that is?”

Recognition flooded Li Mei’s eyes. “Oh yes, I know what that is! I can make this a switch, and then it will be easier to read, too.” She started writing and quickly showed Josie her notepad.

```java
switch(c) {
    case 'r':
        break;
    case 'x':
        break;
    case 'c':
        break;
    case 'm':
        break;
    default :
        break;
}
```

Josie checked her pad and smiled. “That’s exactly it. How about you first rewrite this code using a switch statement and just add more empty case statements for the commands that Benjamin didn’t do the first time. That’ll give you a good shell to work from.

“In the meantime, I am going to update this CR to a Severity 2. The problems are much more extensive than Benjamin admitted. He didn’t even finish this software and I doubt Oscar realizes this. Hiding problems just makes it harder on everyone else.” Josie stood. “Give me a call when you are ready and we’ll start debugging in the lab.”
It was after lunch by the time Li Mei finished her first new version of the code (shown in Figure 2-3), and she was proud of the way it looked. She had replaced all of the if-statements with a switch. It was as perfect as she could make it, and she hoped it would compile the first time without any errors in front of Josie.

She crossed the fingers of one hand and shoved it under her leg as she hit the key to compile her new code with the other. A few seconds later, the screen displayed “Status: Success, Errors: 0” and she slid from her seat and turned to Josie. “See! I told you it would be perfect!”

Josie grinned. “So, make returned zero - it must work!”

“What?” Li Mei stopped short and looked doubtfully at her.

“Nothing - that’s something an old friend used to say. That if the code compiled without errors, it must mean that the software would work correctly. He used to mock another guy on the team who released compiled code without testing it first.”

“Oh, well, I still plan to test it.”

Josie burst out laughing. “I didn’t mean that you weren’t! I mean, I hope you will! Come on, let’s test your new code.”

do
{
    c = getch(); /* wait for user input */
    switch(c)
    {
        case 'c':  /* Mystery command */
            for (j=0; j<i; j++)
            {
                sprintf(tmpstr, "%d %lf ft", i+1, dists[i]);
                dmess(tmpstr);
                getch();
            }
            break;
        case 'm': /* Take measurement and display results */
            dist = measure();
            dists[i] = dist;
            if (i == 10)
                i = 0;
            i++;
            sprintf(tmpstr, "%lf ft", dist);
            dmess(tmpstr);
            dmess("Meter Magic"); /* ??? */
            break;
        case 'r': /* Reset index into memory */
            i = 0;
            break;
        case 'x': /* Multiply last two values together */
            ans = dists[i-2] * dists[i-3];
            sprintf(tmpstr, ".2lf x %.2lf = %.2lf", dists[i-2], dists[i-3], ans);
            dmess(tmpstr);
            break;
        case '+': /* Add last two values together */
            break;
        case '-': /* Subtract last two values */
            break;
        default:
            break;
    }
} while (1);

Figure 2-3  New Code with Switch Program Control.
Very quickly, they ran into problems. Li Mei pressed the measurement button several times in a row and was pleased to see reasonable distance numbers being displayed, but when she tried to multiply two numbers together, the answer was wrong. She continued taking measurements and computing areas, when the device suddenly reported an area of zero square feet.

She sighed and laid the tape measure on the bench.

“You changed the behavior of the device when you created the ‘m’ case to take a measurement.” Josie prompted her. “What did you change?”

“Nothing in the code for area calculation!”

“Yeah, but you changed something that affects the area calculations code. Think.”

“Yes. One thing at a time.” Li Mei took a deep breath and looked back at the original code printout. “If I could use a debugger, I could check the value of the two measurements, and then see if the multiply operation is correct.”

“That’s true, but you won’t always have a debugger. Think with your brain, not with your debugger. What code no longer runs?”

And then she saw it. “I stopped the measurement code from running every time. That means the measurement counter index no longer gets incremented each time. That means it multiplies the wrong values together!”

“Great! Now tell me what two values it uses.”

“Now it ignores the last measurement and multiplies together the two before that.”

Josie nodded, visibly pleased. “Good detective work. I imagine Benjamin couldn’t get it to work originally, and probably used the debugger to randomly change the array indices until he got the right answer. I suspected something was wrong right away because the array indices were strange. You often see paired offsets like [i-0] and [i-1], or like [i-1] and [i-2], depending on when the index counter is incremented, but indexing back to [i-2] and [i-3] is a red flag.

“Remember your palatial office space? All a result of crappy programming practices!”

Without pausing, Josie continued to grill her, “But why did the last area calculation come out as zero when you had just finished a string of valid distance measurements?”
Li Mei turned back to the code. One thing at a time. Think. They’d both taken a bunch of measurements but neither had used the reset function. Where could a zero suddenly come from? Had the measurement counter rolled over? She thought about what would happen if the counter rolled over to zero and then realized the multiply would reference \texttt{dist[-3]} and \texttt{dist[-2]}. She felt excitement rise in her chest; the way Josie was presenting this to her, it was just like one little puzzle after another.

She looked up to see Josie peering at her. “The index rolls over and the multiply operation uses bad values. I can fix this.”

With Josie watching over her shoulder, Li Mei fixed the multiply operation, using a modulo operator, \texttt{\%}, to properly handle the subtraction operation of the array index. Even if the distances array filled up and the index rolled over, it would still access previous measurements at the end of the array correctly. When they had finished removing the bugs in the code Benjamin had left, Josie raised one more issue.

“Before you start adding new code, I’d like you to take one more pass at this code and make the variable names more descriptive and add comments to the code.” Josie stood and stretched. “Your email probably isn’t working yet, but Oscar just sent around the coding standards document again. I’ll send a copy to the printer in the hallway for you.”

“I will make those changes and show you first thing tomorrow morning.” Li Mei rose as Josie gathered her things to leave.

Today had been a much better day than yesterday, and she was happy that Josie helped her debug the software. She was going to be a team member after all.

As Josie walked out of the lab she called after her, “Thank you, Josie!”

On her third morning at Hudson Technologies, Li Mei dropped her backpack in her cube and heard a commotion coming from Josie’s cube.

“The email says the manufacturing run completed without errors, and the first set of Friend-Finder Communicators powered on blue! Success!” Josie’s voice floated over the cube walls, punctuated by Oscar’s directive to “Rock on!”

Li Mei smiled at the excitement; they must have fixed the problem. She walked next door. “Hi. The project is successful?”

Oscar turned to face her. “Yes, Josie and I were reasonably sure everything would work out, but you never know for sure until everything plays out. Randy - that’s my manager - has been pacing for days. Now, officially, that project is delivered and out of our hair.”
Josie interrupted him, “Until they find bugs, and then we get to make at least one revision.”

“True, but that’s expected and already been negotiated. Now that ball-breaker Anders is off my freaking back. Excuse me.” Oscar glanced at Li Mei but she seemed unphased. He pushed off the wall to face her.

“Li Mei, I am officially remiss in not meeting with my newest employee yet, and I would request that you join me for lunch this afternoon. Are you vegetarian?” Li Mei shook her head. “Then we shall venture to Molly’s, a fine establishment for excellent burgers and dark ale among friends.” Li Mei grinned at his mock-seriousness and graciously accepted his offer. He left her with instructions to meet him at quarter to noon.

After he disappeared onto Ritchie Way, Li Mei turned to Josie with amazement in her eyes.

Josie nodded. “He can be a piece of work, but he’s an incredibly smart guy.”

“He seemed really happy. I wasn’t sure my first day.”

“Well, that deadline was stressful for a lot of reasons. Benjamin quit, the project had financial penalty clauses for every day we were late, and he didn’t know much about that product.” Josie added, “Sometimes it gets like that, and you just have to deal with it. We go through periods of late nights and stress, but on the flip side Oscar will randomly give us time off for good behavior.”

Li Mei settled into Josie’s guest chair. “How long have you been here? Has Oscar always been your boss?”

“I have been here for five years and Oscar has been my manager for the last year. I worked for another manager before, but he wasn’t a developer. When I came, Oscar was a senior developer and kind of a superstar debugger, so they promoted him to technical manager and I was put in his team.” Josie fiddled with a pen as she spoke. “I like working for Oscar better than my old manager because he understands the technical stuff. I can ask him questions and he tries to give me real answers, not BS. But,” she conceded, “he’s a new manager and sometimes he gets distracted with projects and I have to bug him for information.”

Li Mei sat listening with her head down. “I am nervous about having lunch with him. I hope he likes me.”

“He will - just keep being honest and showing that you want to do a good job. And don’t order any alcohol at lunchtime, only after work, okay?”

“Deal! Will you look at my new program right after lunch?”

“Absolutely.” Josie added with a smile, “It will be perfect, right?”
Li Mei walked back to her cube after her lunch with Oscar, both energized and nervous from their conversations.

Josie had been right; Oscar was scary smart and also very demanding that she not develop any bad coding habits. After reading the original Meter Magic code, she agreed with him that documentation and communication were Good Things and she made a pledge to herself not to let Josie and Oscar down. Her first few days had been a roller coaster, but she liked Josie and hoped that she would be able to live up to Oscar’s expectations.

She had successfully compiled her changes when Josie’s voice startled her from deep immersion in the new code. She turned quickly to find Josie sitting comfortably with her legs crossed in Li Mei’s guest chair.

“How was Molly’s? Did your carnivore boss order his burger raw in the middle with a slight char on either side?”

“Josie - you scared me!” Caught by surprise, Li Mei felt her back stiffen even beyond her normally perfect posture. “You give me a heart attack!”

Josie doubled over in laughter. “I’m sorry - I didn’t mean to surprise you quite that much.”

Li Mei took a deep breath and gathered her thoughts. “Yes, lunch was very good. Oscar explained things to me and talked about some projects I might do. Molly’s has an interesting atmosphere and was full of people who look like engineers.”

She added, shyly, “Like me now.”

“Yeah, it’s a popular spot for the tech companies around here. Central Jersey has a ton of great restaurants of all types. We should go to Molly’s after work sometime.” She waved her hand. “But now, let’s see the perfect code.”

As Josie dipped her head to the code listing, Li Mei mentally crossed her fingers. She had changed variable names, encapsulated code, and added nicer indentation for readability.

Josie reviewed the code, stone faced, for what felt like forever before responding. “This is a vast improvement.” She fell silent again, then added, “You read the coding standards, didn’t you?”

Li Mei flushed and admitted, “Yes, two times. I found four violations that I fixed.”

Josie snorted up her soda, trying not to laugh. “Li Mei, you crack me up. I know you are trying to make a good impression, but don’t worry about solving all the problems of the world this week, okay?”

Li Mei started to protest as Josie silenced her with a wave of her hand. “Keep going. Walk me through your new code line by line and explain the changes.”
When Li Mei finished, she posed a question back to Josie. “I thought about the if-else statements and the switch some more, and I really don’t understand why the switch is better.”

“Well, I think I mentioned that using a switch is good when the different options are discrete and have nothing to do with one another, like processing user inputs or different commands in a communication protocol.” Josie paused to think. “Or to control large unrelated chunks of code in a state machine. On the other hand, if-statements are much better for continuous ranges of values.”

She grabbed a pen and wrote:

```java
if (x > 10 && x <= 20) {
    (stuff)
}
else if (x <= 50) {
    (other stuff)
}
else {
    (still different stuff)
}
```

“See, you can’t do that easily with a switch. When you design your software, you should think about what method you want to use, switch versus if, because it will give the reader an idea of the type of processing that will occur. This is a way to make the code more self-documenting.

“In your case, I think it makes it more readable, and easier to add new case statements without introducing errors. Also, the compiler can optimize the code, which can make it smaller and faster.”

Josie waited for Li Mei to nod her understanding, and then switched gears. “Do you know what we just did? That was an informal code review. And when you show me that this code compiles and runs as expected, you can go ahead and submit it.”

“Oh okay.” Li Mei stared at her, clearly not sure what was expected.

“So here’s the official procedure. You made major changes to the code, so you are supposed to have a formal code review with three developers. But before that, you must have an informal review with one other person where you explain the changes you made, and you show that the code performs as expected.” Josie made notes for her on a sticky pad. “We’ll finish the informal code review today or tomorrow and then I’ll schedule the formal review for you the first time.”

“Is the code really okay?”
“If it runs correctly, but I think it’s really close. You should have it finished soon. You pulled a couple of really important points from the coding standard that I should commend you on. First, by changing the variable and function names to be more descriptive, you made the software self-documenting. Very nice. You also added several #defines that make it easier to understand and maintain. The layout is also a vast improvement.”

Then she leaned in conspiratorially. “I noticed you used three-space tabs. Others use four- or five-space tabs.”

After a long pause, she added, “It’s a war.” Josie watched Li Mei’s eyes go wide. “Just pick one and stick to your guns. I am a three-space coder too.”

Josie brought the meeting to order. “Has everyone had a chance to review the software? Please tell me how long your preparation time was.”

Li Mei twined her fingers and listened as each of the participants responded. She already knew Ravi, but she had never met Bob or Sundara, and she gave a small smile to each of them. Too soon, Josie called on her to begin explaining the code changes, so she took a deep breath and began.

Reader Instructions: Review Li Mei’s final code (shown in Figure 2-4) and make a list of each improvement that she made. While together her changes make the Meter Magic a properly functioning device, identify the secondary benefits for each of her improvements (e.g., improved readability, maintainability).

“This Meter Magic didn’t have any software requirements document, so Josie and Mike from marketing helped me make one from other documents and Oscar approved it. We will use that for this code review. I started with the first code listing, which ran but did not work. The final code listing has several changes to fix problems and satisfy requirements. Before I review the code line for line, I would like to give a list of overall changes I made.”

Li Mei switched papers and read, “I changed all the variables to more descriptive names and added comments to explain the code. The main loop was originally controlled with many individual if-statements, which I replaced with a switch statement to control user inputs. I made some new functions to encapsulate some code, and also to add error checking. Some arrays could be accessed out of bounds, like the measurement array and also the display string array. I also added white space and formatting for easy reading.” She looked around the table as she added, “I hope it was easy to read.” To her relief, she saw heads nod, and she continued to the final code listing (shown in Figure 2-4).
#define MAX_NUM_MEASUREMENTS 100
#define MAX_LEN 100
#define MEASUREMENT_TYPE 1
#define AREA_TYPE 2
#define ERROR_TYPE 3

void main(void)
{
  /* Variables for distance and area measurements */
  double distances[MAX_NUM_MEASUREMENTS]; /* Storage array */
  double area; /* Area calculation result */
  double meas1, meas2; /* Temporary variables for calculations */
  int meas_ctr = 0; /* Counter for distance measurements */
  char user_cmmnd; /* Input from user from buttons */
  char tmpstr[MAX_LEN]; /* Temporary string to display results */
  int multiply_enable_f = NO; /* Do we have 2 values to multiply? */

  initialization_hardware(); /* Initialization of ports, beam, etc. */
  clear_measurement_memory(distances);
  display_message(TEXT_TYPE, "Meter Magic");

  /* while the device is on, process user commands */
  do
  {
    user_cmmnd = getch(); /* Wait for input from user via buttons */
    switch(user_cmmnd)
    {
      case 'c': /* Display previously stored values one at a time */
        cycle_stored_values(meas_ctr, distances);
        break;
      case 'm': /* Perform a measurement, store the result and display it */
        distances[meas_ctr] = measure();
        sprintf(tmpstr, "%.2lf", distances[meas_ctr]);
        display_message(MEASUREMENT_TYPE, tmpstr);
        meas_ctr = (meas_ctr+1)%MAX_NUM_MEASUREMENTS;
        if (meas_ctr == 2)
          multiply_enable_f = YES;
        break;
      case 'r': /* Reset memory by clearing out values and counter */
        meas_ctr = 0;
        multiply_enable_f = NO;
        clear_measurement_memory(distances);
        break;
      case 'x': /* Calculate area using last 2 measurement values */
        if (multiply_enable_f == YES)
        {
          meas1 = distances[(meas_ctr-1)%MAX_NUM_MEASUREMENTS];
          meas2 = distances[(meas_ctr-2)%MAX_NUM_MEASUREMENTS];
          area = meas1 * meas2;
          sprintf(tmpstr, "%.2lf x %.2lf = %.2lf", meas1, meas2, area);
          display_message(AREA_TYPE, tmpstr);
        }
        else
        {
          display_message(ERROR_TYPE, "Error: Area needs 2 values");
          break;
        }
        break;
      case '+': [READER: code for remaining functions removed for space]
        break;
      case '-':
        break;
      default:
        break;
    }
  } while (1);
}

Figure 2-4 Final Code Listing.
 Nearly an hour later, the code review was complete and developers filtered out of the room. Li Mei gathered software listings as Josie finished jotting notes and rose to face her.

“Congratulations! You have officially survived your first code review in the Real World. You did good, Li Mei. How do you feel?”

Li Mei’s face broke into a smile, tinged with relief. “It was not as bad as I expected. I thought everyone would not like my variable names or they would tell me to change everything because I am not experienced yet.”

“Nah, the focus of a code review is to verify that the code satisfies the requirements and that your code is clean, logical, and follows coding standards. If someone sees a better way to fix a bug or if they find an error, they can make suggestions but we all decide together whether to ask you to change your implementation. Other than that, you can develop your own coding style.”

Josie gathered up discarded notes and listings as they talked. “Now, to finish things off, you need to make the small changes to the software that the group suggested. I will enter the results of our review into the defect-tracking system and call the review a success pending your updates.”

Li Mei nodded.

“Then, you check your new changes into the source code control system like you did before the review, and I will review your changes. Then we can start an official build of the final software and System Test will begin testing it. For now, you are done.”

“Good. I am happy to finish my first software project.” Li Mei stopped suddenly to face Josie. “Josie, I was so worried my first couple of days here. I didn’t think anyone liked me and that I had to work quietly all by myself.” She shifted the papers in her arms. “But since you started helping me, it has been so much better. Thank you so much for helping me. But will we keep working together? Or do I have to work alone on the next project?”

“Well, each project is different and on some you will be the only developer. But most projects have more than one embedded developer, and you’ll also meet folks from hardware and System Test and they will become your allies, too.”

The tension Li Mei was holding in her shoulders eased slightly.

“It’s like this,” Josie explained. “We all try to help one another, and if you have a problem you come to me or Ravi or Oscar. And don’t feel nervous about it, because we are more successful as a team when we communicate. We don’t want another Benjamin. But I don’t think you’ve got it in you to be a Benjamin.”

With a smile on her face, Josie stuck out her hand to Li Mei. “Welcome to the team. You’re gonna do great things here.”
After Li Mei left the code review room, Josie made a final pass and found a notebook sitting on one of the chairs. She quickly located her own amid her stack of listings and comments, and realized it probably belonged to one of the engineers at the review.

Looking for a name, she flipped the front cover and the notebook opened to a page in Li Mei’s clean and precise handwriting.

**Specific Symptoms and Bugs**
- Suspicious indexes into arrays often cause off-by-one errors.
- When something works for a while before breaking, suspect memory problems like boundary condition violations (accessing outside array limits).

**General Guidelines**
- Use programming elements that are appropriate for the function (e.g., switch for unrelated discrete items and if-statements for continuous variables).
- Make code self-documenting with descriptive names, #defines, and comments.
- Sometimes the comments are wrong.
- Consistent tab spacing and white space make code more readable.
- Focus on understanding main() first - don’t get lost in the details.
- Use visual aids like flowcharts and graphs to show functional elements (blocks) and program control logic (connectors). This reveals what the program does, and identifies missing logical and functional elements.
- Coding standards are a good source of bug types and causes, and they also provide fixes!
- Think with your brain, not your debugger.
- Just because it compiles, doesn’t mean it works.
Chapter Summary: The Case of Inherited Code (Difficulty Level: Easier)

Li Mei inherits incomplete and undocumented software for the Meter Magic electronic tape measure; she must reverse-engineer it to understand what the product is supposed to do, and figure out what is wrong with the current implementation. Josie helps her methodically understand, re-architect, and fix the software. She also learns about the code-review process, coding standards, and the importance of documentation.

The Problem Symptom(s):

- Code was not complete; code for some buttons was missing.
- Some code did not work correctly (e.g., multiply was broken).
- Measurements inappropriately occurred for all button presses.
- The product name was displayed at the wrong time.

Targeted Search:

- Divided the software into logical chunks using curly braces to encapsulate functional blocks. Matching braces revealed program flaws.
- Created a flowchart to understand when each function processed.
- Listed many problems from the flowchart and the software listing.

The Smoking Gun:

When the initial software changes caused a square-footage calculation to return zero, she realized that the entire structure of the main routine must be addressed.

The Bugs:

Several software bugs were found.

- Array index off-by-one caused incorrect area calculations.
- Bad program control caused a new measurement on the display to be immediately overwritten with the “Meter Magic” product name, and caused processing to be performed on previously stored measurements.
- Several functions not coded at all.

The Debugging Method Used:

- Using teammates as sounding boards for ideas.
- Reformatting the code and matching curly braces.
- Clarifying functionality by using flowcharts.
- Reproducing the errors with incremental code changes.

The Fix:

The software was fixed and quality improved with these changes.

- Used a switch statement rather than a series of if-statements.
- Added comments, descriptive names, and #defines for improved readability and maintainability.
- Added error checking to ensure arrays were accessed correctly.
- Updated the display function to accept different types of display messages.
Verifying the Fix:
- Developed requirements specification (because it didn’t exist) and used it to understand and then verify the needed functionality.
- Verified array boundary conditions even when memory was full.
- Created Test Plan (in consultation with System Test) for new official load of code.

Lessons Learned:
- Don’t assume software that compiles works correctly.
- Even if the final listing is longer, its consistent format and structure make it easier for anyone to quickly understand and make changes.

Code Review:
The software in Figure 2-1 is a good example of why opening and closing curly braces in the same column can increase readability of code, independent of other changes. In properly formatted code, Li Mei’s handwritten lines would simply be vertical lines connecting the braces. In addition, it is unusual to have the input keys coded as ASCII characters, although this can make testing easier because the code can be tested on a PC using a getc()-type interface before hardware is available.

What Caused the Real-World Bug? After the Chinook helicopter with the upgraded avionics went down, an inquiry was launched. The review board examined the software more closely, running a formal code inspection. The code was so awful they gave up after inspecting a mere 18%, which identified 486 errors, some 50 of which could lead to the loss of the machine.

The lesson: inspections find errors efficiently. The tragedy: the software community has known this since 1976, yet Chinook avionics contractors chose to ignore those lessons. [1]

References


Additional Reading