

---

**Real-World Bug** [Location: The Equator, Pacific Ocean] In March 2000, on a modified oil platform in the middle of the Pacific, the countdown proceeded normally. Sea Launch was prepared to launch a communications satellite into Middle Earth Orbit. Off it went, and the vehicle was soon out of sight. The second stage ignited on schedule, but then fizzled, dropping the \$100-million payload into the ocean.

---

## Chapter **3**

### ***It Compiles with No Errors: It Must Work! Integrating Changes in a Larger System***

**“H**eads up, Ravi, I broke your code.” Eduardo caught up with Ravi just as he turned into the lab. “Wasn’t hard, either. All I had to do was run it!” He followed Ravi in, needling him about how much he enjoyed the endless challenge of breaking developers’ software.

“Gonna write a Change Request against it and the subject line will be, ‘The amazing two-second-long program - hardware doesn’t even get a chance to power up before it’s all over!’”

Ravi turned back to glare at him and Eddie skidded to a halt, the smile melting from his face. Through gritted teeth Ravi informed him, “Defect tracking system won’t accept a subject line that long. Nice try.”

Eduardo tossed his hands in the air. “Yo, it’s okay, man, just kidding. I wouldn’t really write that in a CR.”

“Eddie, I fixed that software just like they asked me to. All they wanted was someone to port the software from some ancient hardware platform to something newer, and they wanted it done, like, instantly. I ported it; the job is done, okay? If something else is broken, it’s not my fault.”

“Sure, no problem.” Eduardo fiddled with the antenna on his cell phone as he watched Ravi stalk away. It wasn’t like Ravi to be such a jerk, but maybe it was just more girlfriend problems. He shrugged and walked back to the lab to write up the test results for Ravi’s code changes.

Eduardo had been a developer for almost a year before he did a stint in the System Test group, only to find that he enjoyed testing software and trying to break it. It was much more fun than writing software from scratch or fixing other developers’ mistakes. Some developers seemed to look down on the people in System Test, as if they didn’t have enough skills to be good coders and ended up in ST by default, but he found the experience altogether different.

You had to be clever to be an ST God, as he was.

Eduardo smiled to himself at some of the bizarre bugs he had identified. Any slacker could run a test script and print out automated results, but it took talent to decipher the results and then run additional manual tests to figure out why the test failed. Eduardo prided himself in taking the extra step. Rather than reporting the symptom of the failure, he tried to identify why the failure occurred to help the software folks quickly find the error. Ravi was usually a good guy, and Eduardo decided to spend a little more time understanding the bug for him.

Eduardo sauntered back to his bench in the lab and prepared to run more tests on Ravi’s code. For this change, he had no automated test scripts - only the CR that indicated that the software program must still run correctly on newer hardware.

When he was ready, he powered on the laptop-sized device and, after a short delay, names and faces of several wild animals appeared around the perimeter of the display. A strange howling noise started, but almost immediately the display blanked and the device powered off.

Strange. He flipped the device over and looked on the back and sides. Wasn’t very clear what this thing was supposed to do.

The original CR was vague, but he gathered it was an informational interface for nature preserves or wild animal parks. Apparently the old devices weren’t very rugged and a series of severe weather events had cracked the cases. He thought visitors were supposed to touch a picture of an animal to learn more about that animal but, unfortunately, they didn’t get much time to do it before the device shut down.

Eduardo power-cycled a unit and leaned over the table to get ready. When the program started, he immediately jabbed a spot on the screen where he had seen an animal’s face appear the last time. Almost immediately, a lion’s face was displayed followed by 11 other animals’ faces in a ring, and this time they didn’t go away. Perched, unmoving, he waited for something to happen, and was quickly rewarded by the roar of a lion, followed by a monkey’s howl, the trumpeting of an elephant,

some bizarre rumbling noise he'd never heard before, and then a rush of other animal calls until all twelve animals had introduced themselves.

He leaned back and smiled. Cool.

As he watched, the screen cleared and was replaced by a wide-angle shot of an open savanna with a description of the lion's habitat and eating habits, reproduction cycle, and lifespan. Selecting each of these buttons led him to other screens that seemed to work correctly.

By now, Eduardo was lost in the challenge. This was the good stuff, he thought, trying to crack the code. Sometimes he could even give hints to the developer about what might be wrong with the software, like maybe a loop counter was off by one, or that a variable was uninitialized because some repeating behavior was different the first time only. All without actually looking at the code!

He thought the initial ring of animals was an opening screen that was probably supposed to stay there so the visitor could look at each animal and choose one to learn about. He pondered, while mindlessly popping his cell-phone antenna in and out of its holder. Could he select each animal individually, even though that first screen seemed messed up? He restarted the program and jabbed along a different side of the screen, but was still presented with all 12 animals and all 12 voices, starting with the lion. But this time he was rewarded with the habitat and information for the lemur, a funky little creature with a long black-and-white striped tail. Soon, with patience, he was able to access the habitat of all 12 animals, from the mighty lion through the elusive grey wolf.

He pushed back, ready to add more information to the CR, starting with a descriptive title: "Opening screen doesn't wait long enough for user input before ending program." In the description section, he outlined the tests he had performed. He added his idea that the program also had a timing problem because all the animal's faces were displayed within a second or two, but it took a lot longer for the voices to play. Was it possible the software wasn't playing the animal sounds soon enough? Well, that was all he could do to characterize the problem without looking inside the box. Basic black-box testing.

He finished up the CR by filling in the preset fields. The problem was a Severity 1 problem, the most critical because the software was unusable. It was up to management to assign the priority level, so he left the default value of 3. If Eddie was right and they needed this change quickly, they would bump up the priority level to a 2 or a 1. The CR committee would review the new CR within the next 24 hours; most likely, it would be assigned back to Ravi to fix. He hoped his extra work would help Ravi out; he liked him and didn't want to see him get into trouble for being sloppy on this one.

Ravi wondered how long before the CR was assigned back to him.

Eduardo was right. He hadn't checked the software fully. Actually, he hadn't checked it at all. He was trying to make a deadline for Oscar, and just didn't have enough time to finish it right.

But he was sure he ported the hardware code correctly!

He logged into the defect-tracking system and called up a list of open CRs assigned to him. The new CR wasn't there yet, so he returned to his email to find another message from home about his marriage prospects. His stomach clenched into a knot as he closed the email and pushed back from his desk to clear his head.

He was about to get up when he heard a new voice behind him.

It was Li Mei. He smiled, thinking about her prior threat to come bother him for absolutely no reason at all, and he turned to greet her.

"Hi, Li Mei! Are you here to make trouble as you promised me?"

Li Mei rushed into his cube and plopped into his guest chair, clearly excited about something. "Hi, Ravi, how are you? Do you have a minute? Am I bothering you?" She looked at him expectantly, guilelessly. He found himself warming to the idea of a distracting conversation with someone new.

"Okay, Li Mei. You are very happy today; what's going on?"

She fairly bubbled over with enthusiasm. "I finished my first code review! They said my code was okay and Josie is going to approve it. Well," she amended, "I have to make two small changes, but it will be no problem." She leaned in to whisper. "I was nervous because Oscar said I had only one week to finish this, and it was not very good code." Pushing back, she added, "I'm so happy to be on this team. Josie helped me figure everything out. She is very smart, don't you think so?"

Ravi stared at her, nodding for her to continue as he felt his throat constrict over his own experience so far. Oscar was a terse manager. People make mistakes, he thought, and Oscar could cut some slack.

"Sure, yeah. Josie's pretty smart. It's good you get along with her. Oscar likes her."

"I wanted to thank you for helping me my first couple of days; you know, taking me around to get my badge programmed." She made a face, "But my picture is pretty terrible."

They shared a laugh, and Ravi wondered what to make of her.

"I am glad to help. A first day is hard, and this place can be harder."

"Why?"

“Well, Hudson Technologies does a lot of different types of projects, and sometimes we’re supposed to act like consultants and go on the road. There are not many really tight teams that work together on the same project for a long time. So, it can be harder to trust people.” He watched the look on her face change to concern. “But if you get on a good team and get good projects, then . . .”

“Do you like working here? On this team?”

“Well, it’s okay. I mean, I’ve only been here six months.” Then he stopped and mentally reset himself before continuing, “Listen, this place is great to learn new things, and you should get as much from it as you can. That’s what I am trying to do.”

Li Mei sat silently; she was watching the emotions play over his face. She had come here happy with her code review and he had deflated her happiness. Li Mei stood up to go and he jumped up to stop her from leaving.

“Hey, I didn’t mean to scare you off. I am really happy that your code review went well.” He put a solid convincing smile on his face and continued, “You can come by any time to share your good news, okay?”

“Yes, Ravi, I would be happy to talk to you about my next project. Josie has also told me that sometimes the developers go to . . .” she paused, and looked at the ceiling as she intoned from memory, “. . . Molly’s Irish Pub and Grill, the home of excellent burgers and dark ale among friends.”

Satisfied with her recitation, she looked back at him and added, “Will our team be going there soon?” She burst out laughing before he had a chance to respond. “Oscar took me for a burger, and I would like to go back and try ‘a dark ale among friends.’”

“Yes,” he responded with some hesitation, “Maybe we should go out sometime.”

---

“What were you thinking?! This was a high-priority change, and you didn’t even bother to run the software on the new hardware?” Oscar paused, shaking his head, and then continued somewhat less vociferously.

“Again, I ask, what were you thinking?”

Ravi balled his fists and waited. Oscar’s rare outburst, complete with hand waving, had quickly come to a silent, pregnant pause.

“Well, I knew it was a high priority . . .”

“Which is an excuse to submit crap, only faster?”

“Hey,” Ravi shouted back, “I said I screwed up and that I am on my way to the lab right now to fix it.” Grabbing his notebook and pen, he angled out of his cube

and headed to the lab, only to realize that Oscar was following him. To his mortification, Oscar trailed him to his lab bench and actually pulled up a stool beside him. Was Oscar actually going to look over his shoulder while he fixed the software?

“Okay, this needs to be fixed immediately. Where do you start?”

Clearly, Oscar was staying. Ravi pulled up the software on his computer screen and stared at it. Eduardo had said something about it running and then halting prematurely. Where *should* he start?

“Well, I will check the changes I made first.”

“Tell me what you changed.”

“This thing is running on a new hardware platform with a new processor and new LCD display. Mike from marketing told me that it’s already a working product; it just needed the new hardware.” Ravi dug through the drawer, pulled out a handful of spec sheets and dropped them on the table. “I read the hardware specifications for the new microprocessor and display, and changed the lowest level of software, the hardware device-driver interface.”

Having Oscar stare at him made him nervous, so he continued talking out loud.

“I started with the digital I/O ports.” As he talked, he walked a finger down each line of code. “There are five buttons on the case that are digital inputs. Things like up and down buttons to make the display brighter and darker. In the old software, they came into the processor on Port D, but that is reserved for the analog-to-digital converter subsystem, so I moved those five inputs to Port A.” Oscar nodded for him to continue, and he relaxed microscopically. “So in the interface header file, I changed the #defines for the five buttons from pins on Port D to the right pins on Port A.”

“How did you know which were the right pins and ports to use?”

“I got the design documents and new hardware schematics from Tom. His hardware group designed the new interface board. All the hardware input and output signals were labeled, so I knew which port lines they were coming in on.”

“Good,” Oscar said. “And you updated the entire interface?”

“Yes, I think so.” He handed the hardware schematic to Oscar, who noted that it was fairly well marked up with check marks and Xs. “I updated the interface software for each hardware line coming to the processor and display, and then I crossed it off the schematic.” He felt his confidence returning; he had gone over the changes several times. But something was missing because the device didn’t work.

Oscar finished looking at the schematic and placed it carefully back on the bench.

“I am relatively confident that you ported the hardware correctly. You are generally a strong coder. But my concern is your tunnel vision. Your software and changes are part of a larger system.”

Ravi interrupted him, “But I’m fixing that now!”

“I just had the CR assigned back to you. Pull it up.”

Ravi logged into the system. Sure enough, a new Severity 1 Priority 2 CR blinked at the top of his list. Feeling self-conscious, he started to read Eduardo’s analysis and was surprised to learn what this product did. He’d only known it was something for a wildlife preserve. Eddie’s descriptions were pretty thorough.

“It looks like Eduardo did a bang-up job trying to characterize this problem for you. Don’t blow off his work or the debugging information he created for you.”

Ravi scrolled through the test results. From the corner of his eye, he saw Oscar steeple his fingers.

Shortly, Oscar stood up. “Keep reading and figure out where to start. I need a bio break. I’ll be back.”

---

Oscar made his way through the cubical farm to the men’s room, relieved himself and stood staring into the mirror in a distracted fog. He’d avoided confronting Ravi most of the morning, debating with himself about leaving it until tomorrow. All morning, niggling resentment at his new managerial position had finally driven him into the lab to get the confrontation with Ravi over with.

Was it always this hard? Becoming a technical manager brought responsibilities that he hadn’t expected. New annoyances. He knew he’d waste more time in meetings, but he hadn’t expected the people issues. And he still had technical assignments, but apparently the promotion hadn’t warranted an office door. Pulling himself away from the tired face looking back at him, he dried his hands and tossed the crumpled paper into the trash bin.

Walking back to the lab, he suddenly got an idea and detoured to his cubicle. He slid into his chair and pulled a palm-sized external data-storage device from his briefcase, connected it to his computer and launched a search screen. Quickly finding what he wanted, he copied a folder to a memory stick dangling from a spare port and then removed it.

Back in the hallway, he thought again about how Ravi had approached his assignment, only this time he wondered how he could entice Ravi to recognize the error Oscar thought he was making. Without his being told. He was slow to recognize his name being repeated.

“Earth to Oscar, come in!”

He turned to see Josie already walking beside him, and marveled at the large bagel she waved at him.

“Whoa, nice bagel.” He tilted his head in apology. “Sorry, lost in thoughts.”

“No problem.” Josie took a long pull from an oversized coffee cup and added, “Hardware brought in bagels this morning, if you want one. Look at the size of this monster. I won’t be needing lunch today!”

He nodded in agreement. “Hey, thanks for taking Li Mei under your wing the last week or so. Is she settling in okay?”

Josie nodded. “I think so. She just jumps right into anything, even though she doesn’t have much experience yet. She’ll be done with the Meter Magic project in a day or so. What are you going to have her do next?”

“I’m not sure yet. I am in the middle of getting Ravi’s project back on track.”

Josie was silent, and then offered, “Yeah, I heard there was a commotion in the cubes. Is everything okay?”

He hesitated and then the words tumbled out before he realized he was committing himself. “This technical manager thing is not exactly what I expected. I gave Ravi an assignment with a deadline, and he submitted crap and he knew it. I want to trust that people will do quality work - I mean, that’s why they’re getting paid. I shouldn’t be getting blindsided by my team, right?” Clenching the memory stick, he punctuated his words with chopped motions as his worries poured out.

“Take you, for example; I give you an assignment and you get it done on time. Your work is good, thorough, and well documented. I trust you.” Oscar relayed the morning’s events to her, oblivious to the pleased look that appeared on her face at his spontaneous praise.

As his rant wound down he finally admitted, “Yes, I did lay into him. I should have been more private about it, but he still deserved to get called on the carpet.” He finally saw her expression and it stopped him short.

“Why are you smiling?”

“You said you liked my work.”

“Yes, I do. I don’t have to worry about you - you’re one of the good ones.” Oscar was flummoxed. “You didn’t know that?”

“Well, I try my best and I really like doing this kind of work.” She shrugged. “It’s just really good to hear it from you. That means something to me. Thanks for telling me.”

Oscar looked down at the memory stick in his hand, and then slipped it into his pocket as he thought about what Josie had said. Did she really not know how he

felt about her work? He tried to think back to the last time he might have told her and realized that he couldn't recall when that might have been. Here was another management responsibility that he had completely missed, assuming that everything was rolling along okay with her.

"Listen, I'm sorry, Josie. I should have given you positive feedback before now. Let's sit down soon and talk about your work and maybe what you want to work on next, okay?" He saw her nod, smile still firmly in place, and he continued, "Listen, let me get back to Ravi. I have to teach him some debugging skills. As Randy is growing fond of grilling into me lately, I don't have time to do everything myself."

---

Oscar made his way back across the crowded lab and slid back onto the bench stool. "Okay, here's what we're going to do. We'll solve this together so I can get an idea how you approach debugging a problem."

Ravi turned to stare at him, surprised that Oscar's voice no longer carried any anger.

"Like I said, the way you handled this assignment was not appropriate and I don't want it to happen again. But I should not have yelled at you in public. I apologize."

Oscar turned to face Ravi and got comfortable on the stool. "So, as engineers, we tend to want to work in a vacuum, but we miss vital information that can help us do our jobs better. If you talk to people on your team about what's going on in a project, sometimes they have seen the problem before or have ideas how to approach a solution."

Ravi admitted, "I'm not sure what you mean." He tapped his fingers nervously under the front edge of the stool.

"I am drawing some conclusions from Eduardo's report. Conclusions that haven't dawned on you yet, but should." Oscar paused and looked at him, exasperated. "Think about it, Ravi - Eddie could not have gotten as far in the testing as he did if your porting job was bad. In fact, I think the bug has nothing to do with mapping the hardware correctly. It is highly likely that the work you completed was done perfectly."

Had he heard Oscar correctly? Was he really praising his work? Ravi searched Oscar's face for some sign of trust.

"Okay." He gave a small nod.

Events were taking a surprising turn.

---

“Let’s start at the beginning.” Oscar pulled up the CR on the screen. “The biggest thing holding you back is your narrow focus on this assignment. The CR requests a port to the new hardware and then states: ‘verify that previous operation is maintained.’ That’s a big red flag - make sure you validate the ENTIRE system, not just the small part you think you changed.

“Now Eduardo has already done some of this testing for you. With that in mind, let’s start over. What do you do first?”

Ravi thought about Eddie’s test results. “I think the opening screen has a timing problem. So we look at that code first.” Ravi pulled up the code on his monitor and looked at the main loop (shown in Figure 3-1).

**Reader Instructions:** Based on Eduardo’s analysis report and Oscar’s feedback, where is the problem? Only `main()` is shown. Focus on the big picture first, and then list any potential problems you see.

Ravi took a deep breath and tried to concentrate. He quickly scanned the initialization functions that he’d worked on before and returned to the main loop. After a function clears the display, the lion is displayed, and then all the animal sounds are queued up to be played. Next, the other animals are displayed one at a time.

Eduardo had joked with him that the program finished very quickly, but his later tests confirmed that the pictures were actually displayed. From the listing, though, he thought animal sounds should also have been playing.

Oscar’s voice brought him back. “Tell me what you are thinking.”

“Well, the pictures are displayed, which is good, but the sounds didn’t play correctly. So, one thing I would consider is looking at the sound function and how I ported the sound hardware.”

Before he could continue, Oscar cautioned him. “Solve the easy problems first, because sometimes those bugs cause side effects.”

“Okay. Eddie said the program ended very quickly, so I would look for normal and abnormal ways to exit the program. Just from the main routine, I could guess that the variable `selected_animal` wasn’t initialized correctly. There must be an interrupt or something that processes incoming touch screen presses - that’s how the variable gets a valid value the first time. If the user doesn’t touch the screen by the time all the animal faces are displayed, that variable could be garbage. If it were already set to `QUIT`, the program would end immediately.”

“Okay, but how did it work correctly in the first place?”

Ravi considered a moment. “This toolset automatically sets all uninitialized variables to zero, which happens to be the `enum` value for `QUIT`. The tools I used in

```

enum { QUIT, LION, HOWLER_MONKEY, AFRICAN_ELEPHANT, EGYPTIAN_COBRA,
      POLAR_BEAR, ZEBRA, OCELOT, HUMPBACK_WHALE, RING_TAILED_LEMUR,
      GALAPAGOS_TORTOISE, ORANGUTAN, GREY_WOLF };
enum { NO, YES };
int touch_screen_input_f = NO;

void main(void)
{
    int selected_animal;

    initialize_ports_and_interrupts();
    initialize_LCD_display();
    initialize_sound_system();

    do
    {
        /* Present all the animals in ring around display edges */
        clear_display();
        display_picture(LION);
        queue_all_animal_sounds();

        display_delay(198);
        display_picture(HOWLER_MONKEY);
        display_delay(198);
        display_picture(AFRICAN_ELEPHANT);
        display_delay(198);
        display_picture(EGYPTIAN_COBRA);
        display_delay(198);
        display_picture(POLAR_BEAR);
        display_delay(198);
        display_picture(ZEBRA);
        display_delay(198);
        display_picture(OCELOT);
        display_delay(198);
        display_picture(HUMPBACK_WHALE);
        display_delay(198);
        display_picture(RING_TAILED_LEMUR);
        display_delay(198);
        display_picture(GALAPAGOS_TORTOISE);
        display_delay(198);
        display_picture(ORANGUTAN);
        display_delay(198);
        display_picture(GREY_WOLF);
        display_instruction("Select Animal to Learn About");

        if (touch_screen_input_f == YES)
            selected_animal = decode_keypress();

        /* skip if user selects quit */
        if (selected_animal != QUIT) {
            if (touch_screen_input_f == NO) {
                wait_user_keypress(); /* wait 'til they select something */
                selected_animal = decode_keypress();
            }
            /* Display the animal-specific information and wait for user to return */
            display_animal_information_screens(selected_animal);
            touch_screen_input_f = NO;
        }
    } while (selected_animal != QUIT);
}

```

Figure 3-1 Original Software Listing for main().

school only set uninitialized global variables to zero, so it's possible the old software worked by accident."

"Not bad. Actually, a pretty good analysis with your manager breathing down your neck." Ravi turned to see a small grin on Oscar's face. He wondered how much of this was predetermined, and blurted out his suspicion.

“Do you already know what the bug is?”

The question caught Oscar off guard, but he admitted that he might.

“How do you know?”

Oscar looked down at the bench before he answered. “Well, that’s a legitimate question. I could prompt you so you could figure it out yourself, or I could tell you. But if I just told you, what would you have learned?”

Raising his gaze to meet Ravi’s eyes, he continued, “And what good would you be on the team if I had to debug all your assignments for you?” Oscar appeared to be uncomfortable with the conversation, and turned his head away to scratch at his neck.

“Well, tell me what to look for next so I can figure it out. Do I look at the sound function?”

Oscar’s face softened almost mischievously, as he stood to extract the memory stick from his pocket. “Hold off on that for now and deal with the main loop. I want to load a program on a computer to show you.”

---

By the time Oscar returned, Ravi had already fixed the initialization problem by adding a new **NO\_ANIMAL\_SELECTED\_YET** enum, and he verified that the program no longer ended instantaneously. All the animal faces quickly ringed the display and the animal sounds started to play one after the other, but he’d identified the next problem. The animal sounds didn’t stop when he selected one animal from the menu. Other animal sounds continued to play until all 12 were finished.

Oscar called out, waving Ravi over. Then he launched a simple DOS window as he talked.

“I want to show you a program that has the same bug as your program. Well,” he amended, “‘bug’ is not exactly the right description. I should say that it suffers from the same *problem* that yours does.”

Oscar turned back to the screen and hit return. The DOS window turned black and two vertical white bars, perhaps an inch tall, appeared near the left and right sides of the screen. A white ball was positioned in the middle. Oscar used the cursor keys and the right paddle moved up and down, zipping from the top to the bottom of the screen almost instantly. He readied himself and then hit the space bar. The ball shot like a bullet to the left, bounced off the left paddle and sped off at an angle to the right. Oscar’s attempt to move the right paddle to hit the ball proved humorously futile. After two more near instantaneous failures, the window displayed GAME OVER.

“That,” Oscar announced, “was the world’s shortest game of Pong. Ever heard of Pong?”

“Yes, that’s a very old computer game. I’ve not played it.”

“It’s the oldest. Want to play it now?” Oscar motioned to the keyboard, not surprised to see Ravi shake his head.

“It’s too fast! There is no way to hit the ball. Can you get a better version that works right? Where did you get it?”

“I wrote it.”

Ravi tried to hide his grin at the thought that Oscar had made a coding mistake.

Oscar continued, “I wrote this program in 1992. And it ran perfectly. My high score is over 10,000.”

Oscar caught Ravi’s eyes and held them. “So why doesn’t it work now?”

---

“Think about the similarities between your project and my Pong game,” Oscar instructed. “Yours is working software that got a hardware upgrade. The upgrade essentially *broke* the system because it wasn’t compatible, but that’s typical for embedded systems because they are so customized to the application. My Pong game is software that also worked correctly on my top-of-the-line 33-MHz 486DX desktop computer, but it doesn’t work right now. To run it on this computer, I would have to make software changes too.”

Oscar found himself warming to the discussion, reminiscing about college and the radical changes in computing power since then. Was he practically handing Ravi the answer? They were back at Ravi’s bench, having reviewed the limited software listings a second time.

To Ravi’s frustration, Oscar wouldn’t let him dig any further into the code.

Ravi sat, unmoving. “Both programs play too fast. The Pong ball moves too fast, and the animal faces are displayed too fast. Each animal’s face is displayed faster than the animal sound.”

“Excellent. Root cause?”

“There’s a software delay between displaying each animal picture. Those delays aren’t long enough anymore. We could make that 198 number bigger and see what happens.”

Oscar nodded. “Yes, we can change that number, but first tell me why you think that will work. Randomly making changes is not the sign of a good debugger.”

“Well, that’s the only thing between each line to display an animal’s face.”

“Not good enough. Let’s first check what that function does. If that 198 value is in units of absolute time, like milliseconds, the function should run the same on old or new hardware.” Oscar paused as Ravi searched for the function (shown in Figure 3-2).

```
void display_delay(unsigned char delay_counts)
{
    unsigned char i;
    int j;

    for (i=0; i<delay_counts; i++)
        for (j=0; j<5000; j++)
            ;
}
```

Figure 3-2 Original Software Listing for `display_delay()`.

**Reader Instructions:** Is Ravi right? Will changing the input argument to this function fix the problem? Does this give you more hints regarding the big problem?

The function was short.

“It’s not absolute time; just a loop within a loop.” Ravi turned to Oscar. “The loop just counts up, doing nothing. I have seen these before; it just uses up processor cycles and creates a time delay.” His smile spread as he continued, “Since we upgraded to faster hardware, this loop takes a shorter amount of time to run, so the animal faces appear faster! That’s it - we make the number bigger!”

“And that, my friend, is the source of your bug.” Oscar stood up and motioned Ravi to exchange seats with him. “Now, normally I would let you finish this up by yourself, but we have a hard deadline, so let’s do this together. Let me drive.”

Oscar pulled the keyboard to him and began typing. Playing around in the code was a passion, he admitted to himself, and solving this problem had already captured him.

“I suspect your first action would be to just choose a larger number and try it out. However, trial-and-error on this one will take too long. There are two reasons why. Well, really one *main* reason - the design of the main loop sucks. I would like to rewrite it to display an animal and then wait for the animal sound to finish playing before displaying the next one. Alternating display and sound function calls. But I checked how the sound engine works and we don’t want to touch it - just let them queue all the sounds at once.”

He finished launching the debugger and connected the device. “Since all the sounds are queued together, we have no idea in time when they finish. Trying to

tune in the right number by running the program over and over is a crappy way to do it. So we're going to characterize the delay and use some simple math."

Ravi interrupted him, "I know - you're going to measure how long the sounds are, and then measure how long it takes the loop to finish."

"Pretty close. We need to measure the execution time for both the delay function and the function that displays the graphic. The sum of those two execution times should be the same as the sound bite so sight and sound occur simultaneously. We'll try to fix the problem by making the delay function delay a little while longer."

"How do we measure the execution times?"

"With the debugger, there are several ways to do it. Watch me. Since we have no crazy multitasking going on, I can use nonbreaking breakpoints before and after the function we are interested in." Oscar paused to gauge his reaction to the apparent oxymoron, but Ravi just waited.

"Anyway, then the debugger won't stop execution and we will get valid timestamps for each breakpoint. We subtract successive timestamps and then we know how long it takes each function to run."

Ravi interrupted him and pointed to the listing. "But that doesn't tell us what to change the number to."

"Hold on - here are the values." Oscar scrawled numbers on a scrap of paper and then headed to a whiteboard (shown in Figure 3-3). "Let's work this out one step at a time. First, the debugger tells us that the `display_picture` function takes 102 milliseconds and the `display_delay` function takes 40 milliseconds. So the total display-and-delay execution time is 142 milliseconds for each animal."

Known information - Givens

Duration of each sound = 1.5 sec

Measured execution times

`display_picture()` = 102 msec

`display_delay()` = 40 msec

total time = 142 msec

Figure 3-3 Whiteboard Notes for Known Information.

**Reader Instructions:** Oscar's whiteboard notes are shown in Figure 3-3. Based on these measured values, can Eduardo's analysis be correct? Also, what can you derive about the delay function? You now have enough information to propose an acceptable solution - list your proposed changes before continuing.

Ravi consulted his computer and relayed, “If I multiply 142 milliseconds by 12 animals, I get 1.7 seconds to display the entire opening screen. That matches what Eddie reported in the CR - about 2 seconds until the program ends prematurely.”

Oscar nodded. “So far, so good. Now we calculate the new time delay we need to make the sound and pictures match up. If each sound bite is 1.5 seconds and the display routine currently takes 102 milliseconds, the delay function must be changed to waste 1.398 seconds. Now how do you compute the new reload value to do this?”

“I would figure out what the value 198 means in time. We need to know how much time each count really takes before we find a new reload value. And since the code is a nested loop, the function actually loops  $198 * 5000$  times.”

Oscar grabbed a calculator. “The current time delay for 198 is 40 milliseconds. Dividing  $198 * 5000$  into 40 milliseconds gives a time delay of 0.04 microseconds for each count. Pretty darn small.”

He continued to write on the whiteboard, thoughts racing ahead to the magnitude of the new reload. “Finally, we compute the new loop counts needed to waste 1.398 seconds. Since each count is 0.04 microseconds, that comes out to be . . .” Punching again at the calculator he concluded, “about 35-million counts.”

After working his own calculations, Ravi added, “That means that the reload value for the outer loop now becomes 6990. That would be the 35-million number divided by the inner loop counter of 5000, right?” (Their calculations are shown in Figure 3-4.)

“Bingo. Now, that number may not be exact because we made some estimates, but it will be pretty close.”

“That’s it? That was a lot easier than plugging in numbers starting from 198.” Ravi headed back to his bench. “I will go make the change now and tweak it in.”

Oscar was standing to follow him when his cell phone beeped with a text message from Randy asking for the final code. They were nearly done, and he glanced at his watch to gauge their progress before calling out to Ravi while answering Randy’s message.

“Again, Ravi, think before you code. If you just change that number, will the code work? Quick hint - no.”

Ravi turned back. “What do you mean? We just worked it out - that value will delay the right amount of time.”

“In an ideal world, yes. What will the `display_delay` function do with a number like 6990?” He tapped his fingers on the edge of the bench, and then pointed to the function declaration. “The variable is an unsigned char. What will happen?”

1) New Time Delay Needed:

$$\begin{aligned} \text{delay} &= \text{sound time} - \text{display time} \\ &= 1.5 \text{ sec} - 0.102 \text{ sec} \\ &= 1.398 \text{ sec} \end{aligned}$$

2) Find Time Elapsed for each Loop Count

$$\begin{aligned} 1 \text{ count time} &= \frac{\text{total delay time}}{\text{total loop counts}} \\ 1 \text{ count time} &= \frac{40 \text{ msec}}{198 * 5000 \text{ counts}} \\ &= 0.04 \frac{\text{usec}}{\text{count}} \end{aligned}$$

3) New Counts Needed:

$$\begin{aligned} \text{New time delay} &= 1.398 \text{ sec} = \text{new counts} * \frac{1 \text{ count}}{0.04 \text{ usec}} \\ \text{so...} \\ \text{new counts} &= 1.398 \text{ sec} * \frac{1 \text{ count}}{0.04 \text{ usec}} \\ &= 34,950,000 \\ &= 6990 * 5000 \end{aligned}$$

$$\therefore \text{New reload value} = 6990$$

Figure 3-4 Whiteboard Notes for Delay Mathematics.

“The max value is 255. Oh, I didn’t think of that.”

Ravi dropped his chin in one upturned palm, staring at the code with a confused look on his face.

“Change the variable to an int and you’ll be okay for delays up to 32,767.”

As Ravi bent to work, Oscar tapped out a response to Randy, but before he could finish the text message, Ravi groaned and waved him over.

“I found other references to this delay function.” He leaned aside for Oscar to look. “It’s buried in the sound-driver code, even though that code has nothing to do with the display. It looks like they’re also using it for a generic delay.” Ravi looked up at him and asked, “If we change the delay code, what’s going to happen to the rest of the system?”

Ravi’s words hit him like a brick, and Oscar felt his face get warm. He glanced back and forth between the function calls and realized Ravi was right.

“Crap.” He dropped down on a stool, his mind racing between the deadline and the new information. The software was more unstable than he thought. If they fixed the delay here and the function was called elsewhere in the code, they might break other software without realizing it.

Something nagged at him not to change the main interface, but to leave all the 198s and scale the function internally to contain the code change. To further minimize risk.

This seemed to be spiraling out of control.

Ravi looked to him nervously, clearly unable to provide a robust solution.

Exhaling forcefully with his decision, he motioned again for Ravi to let him drive. “We can’t change the display calls in the main loop. We have to change the delay routine itself so it still accepts 198 but converts it to the new delay value, and hope that the scaling translates to the rest of the system as well. It won’t be exact, because we don’t have floating-point math, so the speed ratio will be truncated to a whole number.”

The editor was open to Ravi’s changes, and Oscar quickly added to the file. (See Figure 3-5.)

```
#define HARDWARE_UPGRADE_SPEED_RATIO    (6990/198)

void display_delay(unsigned char delay_counts)
{
    int i;
    int j;
    int new_delay_counts;

    /* CR060805 Convert incoming delay argument to new loop counter value
       to support new (faster) hardware. Measured conversion for old reload
       of 198 becomes 6990. */

    new_delay_counts = delay_counts * HARDWARE_UPGRADE_SPEED_RATIO;

    for (i=0; i<new_delay_counts; i++)
        for (j=0; j<5000; j++)
            ;
}
```

*Figure 3-5 Final Software Listing for display\_delay().*

“Check what I did, figure out the error and tweak in the timing. Then test it for basic functionality. I’ll let Eddie know it’s coming, okay?” Oscar pushed off from the bench to leave, but changed his mind and turned back.

“Add a comment block for that function to explicitly state that it depends on the underlying hardware platform. Also put in that time-per-count conversion factor we derived and the maximum allowed input value to the function. That’ll help the next person.”



Ravi felt nervous walking to Oscar's cube. The day had been a rush of emotions and he felt a little worn out. The changes were done and tested. Working with Oscar had been stressful, but oddly enjoyable. And he felt he'd learned something. It would have taken him a lot longer to find the bug on his own; Oscar just seemed to know where to look. Taking a breath, he tapped the edge of the cube and stuck in his head.

"I made the changes and tuned in the reload value. It was pretty close; the timing error is less than 1%. Eduardo's rerunning the system test now." He paused, and added, "Thanks for showing me how to find that problem."

Oscar tipped back in his chair and nodded. "It was a good experience."

"I had a question, though. Why couldn't we just write a real delay function, one that measures milliseconds or something like that?"

"That's a good thought. But what we have here is something called 'legacy code.' Working code from the past that we don't fully understand, but we have to maintain. Truthfully, the way that sound system is implemented scares me and I don't want to mess with the timing in that subsystem. I don't want to risk touching the timer subsystem and interrupts just to rewrite the display delay function. With legacy code, it's better to make only small isolated changes that affect the fewest number of systems, and test each change individually." [1]

"That makes sense." Ravi crossed his arms. "I also had another question. That Pong program. You had that old program here, at work, after all this time?" It seemed insane that he had that program so close at hand, coincidentally on the exact day he needed to show it to Ravi.

"I don't throw anything out. The joys of a dirt-cheap data storage; everything's with me. All the software I have ever written."

"That's how you knew what my bug was."

"Turns out the bug in the Pong program is very similar. I generated the delays by calling a library function that was provided by the compiler vendor. The function generated delays in one-millisecond units, and the delay length was specified by an integer argument. It turned out the vendor implemented the delay function as a simple decrement loop calibrated to the standard speed of the PCs of the time. Newer systems are much faster, screaming along at 3+ GHz, so the 'standard millisecond' using that delay loop is now *substantially* shorter."

Ravi blurted, "And that's why the little ball is screaming across the screen!" Oscar shared in his humor.

"And just one more question." Ravi couldn't believe he was about to make the suggestion. "Li Mei heard that sometimes people go to Molly's after work and I told her that we should go. I thought maybe all of us. I mean, if you want."

Oscar continued to rock back in his chair but he didn't answer right away. Ravi bit his lip. Just because Oscar helped him with a debugging problem didn't mean he wanted to hang out with him. He was the manager. But before Ravi could retract the offer, Oscar nodded at him.

"Actually, that is an excellent idea. All three of you have finished projects in the last couple of weeks and Li Mei is a new member. It's high time we all go out and celebrate."

"A toast." Oscar raised his glass, and Josie, Ravi, and Li Mei followed suit. "We're here to celebrate some tough project deadlines and a new employee. So here's to dead bugs and clean compiles."

Cheers from around the table mixed with the background noise of the pub. Oscar looked around at everyone laughing and helping themselves to plates of nachos, potato skins, wings, and dip.

His team.

It would be difficult to do this right. To balance needs. He'd practically taken over Ravi's bug fix to get it done in time. But the experience of the last few weeks seemed to click something in his head. He saw Josie raise her glass.

"Another toast." Josie looked straight at him as she spoke. "This one's in honor of Oscar: a quote befitting our daily activities. 'Computers allow you to make more mistakes faster than any other invention in human history with the possible exception of handguns and tequila.' Here's to his teaching us to solve them." [2]

"Oh, that's a good one!" Li Mei took another drink of stout and grimaced. "But I think I do not like black beer." She turned to Ravi. "Did you get your code review approved too?"

"Well, I guess so." Ravi looked uncomfortably around the table.

"That's good. My Meter Magic is completely done now, even the new functions. What was wrong with your program?"

Li Mei's forwardness surprised Oscar, and he wondered how honestly Ravi would respond.

"Well, I had to port software to a new set of hardware, but it turned out to be more than just remapping hardware lines." He described the animal preserve product and explained how the old program had implemented a time delay. "Even the variable initializations didn't port over correctly. I can't believe how much could break just moving to different hardware."

“Another round, folks?” The waitress dipped her head in, patting her sleeve dry with a towel. “I am sorry,” she added with a laugh, “Johnny tapped another keg and spilled it everywhere. It’s fresh if you want it, Oscar.”

Josie smiled and elbowed him. “First-name basis, huh?”

“Thanks, Maria, I’ll have another.” He drained the rest of his glass. “Say, you’re a dishy one. You want to come home with me tonight?”

“Sure, I’m off at 8.” Maria mopped up a spot on the table and turned to wink at him as she left with the empty glasses.

The table was suddenly silent.

Oscar straightened sharply and announced with a finger in the air. “This reminds me of another quote. ‘Programming is like sex: one mistake and you have to support it for the rest of your life.’” [3]

“Here you go, Oscar, fresh as it gets.” Maria carefully set the full glass in front of him.

“Thanks, Maria, you’re the best sister-in-law in the world. I’ll call Toni and tell her you’re coming over.”

“I can’t believe you did that! I knew you’d never cheat on your wife.” Josie smacked him in the arm as he tried to open his cell phone, and he scrambled to catch it before it hit the floor.

“Oh, I’m sorry!” Josie bent to assist, but he waved her off.

“Tis but a scratch. Just a flesh wound.” [4]

“Oh, no. I get it,” Maria groaned. “It’s not me you want to see. You just want me to bring over more Monty Python videos.”

---

## *Things Ravi Learned about Debugging Today*

### *Specific Symptoms and Bugs*

- *Initialize all variables. Don’t assume the compiler will do it for you.*
- *Document any underlying hardware assumptions.*

### *General Guidelines*

- *Debugging tools allow simple timing characterizations without stopping the program execution.*
- *Randomly making changes is not the sign of a good debugger.*
- *Bug report descriptions can be misleading.*

## Chapter Summary: The Case of the Abbreviated Program (Difficulty Level: Easier)

Upgrading an existing embedded information device for wildlife preserves to newer hardware causes unexpected problems when previously functional software breaks. Porting is more than obvious changes (I/O port lines, buttons, LCD display) and subtle interactions involving timing are caused by old hardware-dependent code. A hard deadline forces a trade-off on possible solutions to reduce risk and System Test burden.

### The Problem Symptom(s):

- After the animal device powered up, it displayed a ring of animal faces and then quickly powered down.
- Appearance of animal faces did not coincide with audio playback of each animal voice.

### Targeted Search:

- Methodically verified previous hardware port changes using a pencil, old and new schematics, and old and new software.
- Brainstormed the cause of immediate halt and display-sound mismatch using information from the System Test report.

### The Smoking Gun:

The delay routine used to synchronize presentation of animal faces and voices used hard-coded loop counters to create a time delay.

### The Bugs:

Two bugs directly related to porting software were found:

- The variable that ends the program was “accidentally” initialized to the ENUM value for QUIT, causing a premature end to the program. (After a compiler organizes the application object files into text (code), data (initialized variables), and bss (uninitialized variables) sections, the linker creates an executable that causes these sections to be loaded into the appropriate memory locations (text first, then data and bss). These locations are system dependent, but generally include ROM, disk, FLASH, and RAM. Vendor-supplied startup code normally handles initializing data and clearing the bss section before jumping to main(), although old startup code may not do this and variables may end up with weird values (for instance, a pattern left over from a power-up RAM test). As a rule, operations on uninitialized variables are undefined. Initialize a variable or assume the value is garbage.)

- A for-loop was used to generate a time delay. (On slower hardware, these do-nothing commands take noticeable time to execute. This was a common time-delay method. However, compilers that optimize programs for speed look for do-nothing code like this and remove it. Therefore, this calibrated delay loop would execute in zero time. This compiler optimization can be defeated by including a simple global function call - that doesn't have to do anything - in the inner loop.)

#### The Debugging Method Used:

- Characterizing the device using *black box debugging* without looking at the software. Two major symptoms were identified that directly correspond to the two critical bugs later found.
- Using *white box debugging* methods including code reading to understand the main loop.
- Using debugger breakpoints to measure execution time of different functions in order to compute new loop indices.

#### The Fix:

- Initialized the `selected_animal` variable.
- Changed the delay function internally to accept the old input arguments, which were then converted locally to new loop counter values to mimic the old observed time delays. This reduced the risk of introducing other problems, and reduced the number of required code changes.

#### Verifying the Fix:

Used nonbreaking breakpoints to measure the new delay time, and to verify that animal faces and sounds occurred simultaneously.

#### Lessons Learned:

- Make small, planned changes to legacy code, testing each in isolation to ensure nothing else is affected.
- Don't assume anything will work correctly when porting code to new hardware or a new compiler. Methodically locate and test all obvious and subtle interfaces between hardware and software.
- Expand your focus beyond the part of the system you touched.

#### Code Review:

This software segment is reasonably self-documenting, although it is not apparent that the animal pictures and sounds should occur simultaneously. Explicitly triggering both to occur simultaneously is desirable. Hard-coded delays of 198 aren't in real units. Consider a CR to evaluate rewriting the function to accept time in real units. This would require changing all software that uses the function, so the benefit should be weighed against the risk.

**What Caused the Sea Launch Failure?** The Sea Launch mission was doomed by a change to one conditional statement in the code that did not initialize a variable, leaving a helium valve open prior to lift off. An excessive amount of gas leaked away, leaving nothing to pressurize the second-stage fuel tanks. The rocket could not reach orbital velocity and the flight was terminated.

The million tests automatically conducted in the hours prior to launch missed the one-line code error.

The lesson is that software is very complex and tests don't check everything. However, code coverage mandated by the DO-178B Level A software standard would have picked up the problem. Though this is an expensive standard to support, it's a lot cheaper than the \$100-million lost payload. [5-6]

---

## References

- [1] Feathers, M. (2002), "Working Effectively With Legacy Code," Object Mentor, Inc. Accessed from [www.objectmentor.com/resources/articles/WorkingEffectivelyWithLegacyCode.pdf](http://www.objectmentor.com/resources/articles/WorkingEffectivelyWithLegacyCode.pdf).
- [2] Wikiquote, "Quotes about Computers and Computer Technology," Mitch Ratcliffe. Accessed online September 9, 2006 from <http://en.wikiquote.org/wiki/Computers>.
- [3] Quote Garden, "Quotations about Computer Programming," Michael Sinz. Accessed online September 9, 2006 from <http://www.quote garden.com/programming.html>.
- [4] Gilliam, T. (Director) and Jones, R. (Director), (1975), *Monty Python and the Holy Grail* [motion picture]. United Kingdom: Mark Forstater and Michael White.
- [5] Boeing (July 1, 2000), *Sea Launch - Summary of Investigation and Return-to-Flight Preparations - July 2000*, news release. Accessed from [http://www.boeing.com/news/releases/2000/news\\_release\\_000714t.html](http://www.boeing.com/news/releases/2000/news_release_000714t.html).
- [6] Sea Launch (2000), Past Launches: ICO-F1. Accessed from [http://www.sea-launch.com/past\\_icof1.html](http://www.sea-launch.com/past_icof1.html).

### Additional Reading

Allen, M. (2002), "Bug Tracking Basics: A beginner's guide to reporting and tracking defects." *STQE* magazine, Vol. 4, Issue 3, pp. 20-24. Accessed online May 18, 2006 from <http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectId=5898>.

Ganssle, J. (2004), *The Firmware Handbook*, Burlington, MA: Elsevier (Newnes imprint).

