The Bug

Ellen Ullman

The Bug

Ellen Ullman

The Bug Ellen Ullman
In 1984, at the dawn of the personal-computer era, novice software tester Roberta Walton stumbles across a bug. She brings it to its inadvertent creator, longtime programmer Ethan Levin, and the two embark on a hunt for the elusive bug, nicknamed “The Jester” for its tendency to appear randomly and only at the least opportune moments, jeopardizing the fate of the company. Ethan’s attempts to find a solution soon become a frightening obsession that threatens to destroy both his professional and personal life. Roberta, on the other hand, is drawn to the challenge. Forced to learn how to program, and seeking refuge from her own private troubles, she becomes enthralled with learning to speak the computer’s language. Expertly merging code with prose, big ideas with intensely personal stories, Ellen Ullman brilliantly illuminates the space between human beings and computers—a space we occupy every day as we peer into our monitors.

The Bug Details

Date	:	Published July 13th 2004 by Anchor (first published 2003)
ISBN	:	9781400032358
Author	:	Ellen Ullman
Format	:	Paperback 368 pages
Genre	:	Fiction, Science, Technology, Novels, Science Fiction

From Reader Review The Bug for online ebook

Lisa Eckstein says

This novel is about the quest to track down and fix a software bug, and I've never read another piece of fiction that makes authentic programming details such an integral part of the plot. If you're tickled by the idea of "kill -9" as a plot point, you'll like this book. But if you don't know what this means, don't worry, because all is entertainingly explained within the text, and the story is about so much more than a bug.

The setting is the mid-1980s, during the early days of graphical user interfaces. A software tester who wanted to be a linguistics professor discovers a bug that can't be replicated. She passes a report to the engineer responsible for the front end code, and he tries to ignore the problem, the same way he's ignoring his girlfriend and the increasing distance in their relationship. In time, the bug reappears, but it remains elusive, unable to be captured or corrected, and it gradually wreaks havoc on the lives of these two characters.

I admire the way Ullman digs into the depths of both code and human behavior to tell this story. The strong and careful plotting make this is a suspenseful and fascinating read.

Parker says

This one was so engrossing I canceled plans and stayed up late just to finish it. It's a wonderful novel from Ellen Ullman about software development at a database maker during the early- and mid-80s PC revolution, but it's really about relationships between people, and between people and machines, and how we deal with flaws in those relationships. Most surprising to me, is that the software bug named in the title is fleshed out as a specific and realistic programming error, and is actually the source of a great deal of dramatic tension and sort of a character in itself. Very well done.

Robin says

> Debugging: what an odd word. As if "bugging" were the job of putting in bugs, and debugging the task of removing them. But no. The job of putting in bugs is called programming. A programmer writes some code and inevitably makes the mistakes that result in the malfunctions called bugs. Then, for some period of time, normally longer than the time it takes to design and write the code in the first place, the programmer tries to remove the mistakes.

The Bug is an utterly absorbing tragedy in four acts. Set primarily in the mid-80s in a software startup* working to build the first ever database that is *on the network* (!) and that you can interact with *using a mouse* (!), the book follows two main protagonists. These two main characters, despite being extremely prone to making poor decisions (which usually bothers me to the point of book abandonment) are somehow unfailingly sympathetic. Ethan Levin, programmer, and Berta Walton, tester, think they are nothing like one another, but they share a core set of traits that make them both easy to identify with. Above all: frustrated

ambitions, relationship issues**, imposter syndrome, and a troubling habit of caring about quality.

* It's NOT Oracle, okay? The end matter of this book makes very clear that NO resemblance to actual companies, vendors, or products are in any way implied. NOT Oracle. Got it? Not. Oracle.

** Relationship issues is putting it mildly: one wonders how they even managed to get into long term relationships in the first place, and the inevitability and slow-motion progression of their utter failure plays out as if inextricably linked with the mysterious and inscrutable bug at the novel's heart.)

You might love it if:
- you have ever squirmed in agony while watching the TV show Silicon Valley.
- you have ever felt betrayed by a world of uncertainty when you carefully chose courses of study that would allow you to thrive on formal and predictable rules.
- you have ever harbored defenestrative thoughts about computers.
- you have ever speed walked across half of an open-plan office to turn off the infernal beeping sound on the fourth-rate latte machine, and railed silently against the oblivious humans who were standing right next to it as if they didn't even notice the sound.

You might also love it if you simply love well-written stories about humans interacting with the computers that increasingly make up our world. Ullman wrote this book shortly after the .com crash, looking back two decades from there to a time, which now seems like the impossibly distant past, when confronting and examining the psychological oddities of interacting with relatively basic computer systems was still a novel experience.

> The thoughts were gone, decomposed, passed into code, where they worked, where they ran, but could not be reassembled into human-think. All those tumbling thoughts had become marching lines of stars, pointers to pointers to arrays of pointers, functions calling functions calling functions. Layers of code talking to code, machines muttering to themselves in their own language.

This novel rises to five stars due to a few over-the-top bits of startling, utterly true-to-life aspects that resonate strongly with me personally:
1. Literary references. (How many books set in Silicon Valley contain a *Middlemarch* reference? And two flakey servers called Beowulf and Grendel?)
2. Sound and light sensitivity. (Preach!)
3. Interest in linguistic formalism as an indicator for predilection for other logical realms, like engineering. (Why didn't anyone tell me this in undergrad? If I had read this book the year that it was published, would I have made different career choices? Discuss.)
4. Small, immediate tragedies, perfectly described. (Having to take a meeting with "the Bobs" while sitting on an exercise ball. Mandatory training courses that are woefully irrelevant. Accidentally running *rm* * and erasing an entire day's worth of work on a workstation named "hubris.")

Fair warning, however. Be prepared to descend to some dark places. This is not about a software bug. This is about human frailty, and it will kick you where it hurts.

Charles says

First, let me state that I have a lot of respect for Ms. Ullman as an Essayist on computer technology and techie org behavior.

Being a refugee from geekdom, "The Bug: A Novel" accurately describes the technology and socio-dynamics of writing software in those bygone days. However, the novel is wan and bloodless. Ms. Ullman's prose is crisp and clean to read, but it fails to convey strong emotion. In particular, she misses the potential for the humor, ironic, puerile, or otherwise in the story.

This book is a read that evokes in me a lot of nostalgia, but it is hardly, "gripping, exciting, and compelling".

Martin McClellan says

Ullman is an American treasure. So rarely do voices so unique and interesting emerge, and in her case only after a career in a field unrelated to writing, but related to this book: computer programming. This is not a page turner, although I certainly kept my interest. It is not a thriller, or a paint-by-numbers escalation into an expected exegesis.

This is a novel exploring the obsession and devotion it takes to hold a portion of a complex programming problem in your mind, and execute it. In this case, laden with the drama our protagonist (or, antagonist, since the person we spend the time with most is surely that) has wrought himself with his inattention and difficult social manner, and is certainly paying for during the chapters of the work.

Ullman quite recently wrote a New York Times article about her experiences being a woman in the tech world. This book covers much of the same ground without an editorial finger wagging at the reader. Still, the experience of her character, this woman involved in the tech world of the 80s on the verge of changing everything we know, is a precursor to the conversations we are having today over women in the tech world and how welcome they feel, and how effective at their jobs they can be while dealing with the cascading nonsense of a traditional boys club.

This is not a preachy book, though, lest I give you that idea. It is a book that tells a clean simple story, and in the way of it exposes you to the harsh reality, the monastic existence, the soulless sacrifices these early Silicon Valley workers were part and parcel of.

A fascinating read, a historical gem, a woman whose finally wrought prose is worthy of your attention, and who will allow you to arrive at your own conclusion and walk away with your own experiences. That in of itself is rare enough these days, and certainly worthy of the read.

Bertrand says

Lately I have been trying to learn some programming. My mind has been shaped by thirty years of favouring the humanities, so at face value, knowing fifty ways of writing 'hello world!' is not exactly thrilling. Of course there is a lot more to it than that: tantalising analogies with formal logic and linguistics open wide speculative vistas too, though again those were areas I never really dared to look into, precisely because of

the 'symbolic compression' they share with programming.

To keep myself on track, I have tried to fill my prized leisure-time with books not *of* programming but *about* programming, from a general introduction to how hardware actually work, to a history of computing from Pascal onward. I also looked for fiction about programming: code, after all, is probably the root metaphor for culture, and as such could make a compelling theme for a novel.
Surprisingly, however, I found little that deals expressly with the subject. Ullman received some accolades, mostly for her non-fiction and for being among the rare women to witness the personal-computing revolution (mostly leaning in, I think) - and until Dunkle Zahlen is translated (or my German somehow drastically improves) I thought this one would do the job.

In a nutshell, narration is divided between Roberta, a linguistics PHD who ends up testing software after failing to build a career in academia, and Ethan, another failed academic turned computer wiz. Both work, in the early eighties, for a start-up building B2B data-base softwares, of which Ethan is put in charge of the graphic interface. Both are full of resentment with their situation, and their private lives suffer from it, as it does from the strictures the computer imposes on their minds. When Roberta identifies a particularly nasty bug with the graphic interface, they are brought together in a frustrated quest to track down the culprit, which proves to be lastingly elusive. Ethan, plagued by anxieties in all the wrong places, descends into delirium. As the bug, now christened 'The Jester' by company-culture, slowly take over his life, Roberta watches from the side-lines.

If that does not sound like the most original plot in the world, it's because it isn't. Ullman does promise, in the spirit of her celebrated non-fiction, to afford the reader a glimpse of the world of programming, its rites and jargon, its hierarchies and its symbols. That much she does alright, in the colloquial manner of the tour guide rather than in any details: enough to gain some feeling of familiarity with the machine, not enough to actually understand much of how it works. Roberta's own discovery of the subject serves as an introduction, and the tour though predictable, is enjoyable enough.
The main issue shines through pretty quickly, namely the abysmally flat supporting cast. Even her two 'heroes' are hardly convincing. Roberta, and especially Ethan, are fundamentally maladjusted characters, embodying that lasting myth of neck-beard romanticism: the all-too-rational genius, who cannot cope with society's indulgent fuzziness, and seeks refuge in the certainties of science. Along comes of course the smug and naive satisfaction of the Silicon Valley self-made man, foreclosing any sense of political or personal responsibility. The reader cannot decide whether their tedious love-life and dedication to routine is supposed to be tragic or comical, while the slow trickling out of the secondary characters from their unravelling lives is welcomed with a sigh of relief (special mention here to the big-breasted bisexual German sys-admin, who listens to Einstürzende Neubauten: a very nineties trope, though the book is set in the eighties and written in the noughties).

Characterisation, then, is dreadful, but not all is as bad. As probably befits a novel, it is the emotional experience of the programmer which really shines through, and here we have a few moments of bravery: "For the first time, I understood there was a mapping between the symbolic words of the code and the physical existence of the machine. And something in me shifted. I decided I already knew far too much about words . . . Now I wanted to know more about the machine." (171)
This is an epiphany I went through not long ago, realising how successfully stacks and interface eclipse the materiality of computing. Similarly, Ullman has some insights into the ecstatic pleasures of formality, of letting language speaks: "The cleanliness of programming was a balm. I had spent months unlearning the desire to be unique. I was trying to write code so standard in form, so common in expression, that my work, ideally, could not be distinguished from another programmer's. I was striving for a certain clarity and simplicity, a form of impersonal beauty" (176).

Ullman also clearly grew up in the age when video killed the radio star. She takes far too much at heart the common-place dictum 'show don't tell'. To any prospective writers out there, I should like to say: better a well handled expository dialogue, than a poorly-observed reliance on a hollywoodian body-language. The info-dump will only ever be as bad (or as good) as the ideas you put in there. Clichéd mannerisms will fill my minds' eye with visions of bad sitcoms no matter the point you are trying to make.
However Ullman is not a bad writer. Existential encounters with code aside, she also has a few perceptively chronicled episodes, such as this earthquake scene: "How long did it go on? Seconds, minutes, an eternity, as time slowed down, down, down in the accumulating awareness of things that formerly had not been known to express themselves. Walls that had been solid, groaning. Doors that had been closed, rattling to be opened. Windows that had been clear, bending with displeasure." (145)

The problem, then, is not that the author is incompetent, but that she eagerly substantiates the platitude on computer-people being oblivious to social reality, a self-fulfilling observation which luckily is today in the process of being disproved. I suspect it was a side-effect of computing moving from research-tool to consumer-product, and the consequent reshuffling of software engineers class identity.
This also shows in her handling of technology as a theme, whose impact is seen exclusively on the individual, never in collective, let alone political terms. The idea of the bug-haunting, which if not groundbreaking still had potential, is not very much exploited: Ullman's dignified concern with the neighbours' music at night, or the boyfriend not returning the calls leaves little room for technognostic considerations. 'The jester' is thus neither PK Dick nor Kafka, but rather some interesting backdrop to the uninteresting lives of its victims.

All in all, the book is a disappointment. It points to what a novel on programming could be, it gestures toward a rarely acknowledged subjective dimension of coding, and has some poetic insights into the process, but is irreparably mired with very poor characterisation and lacklustre plotting.

M. L. Wilson says

The Bug is the debut novel of writer and computer programmer, Ellen Ullman. The novel is a semi-autobiographical story which is based upon her years working as a programmer in for a company in California’s “Silicon Valley” in the 1980s. Ullman fleshes herself out in the novel through the character of Roberta Walton, a quality tester at a small software firm. It is through her discovery of the presence of a software syntax error—a bug—that breathes life into the novel.

Ullman does a fine job of setting the stage of an early 1980s software firm struggling to write code and develop programs amid heavy competition while detailing the inherent difficulties associated with such innovation. It is through this frenetic atmosphere that Walton is forced to confront a lead programmer over his alleged mistake in the code. The programmer, Ethan Levin, takes a dim view of Walton’s assessment as he does not view her as qualified. That, coupled with his insecurities, provides the impetus of Levin’s downward spiral.

The bug which comes to be referred to as The Jester continues to dog Walton despite Levin’s dismissals. An inevitable confrontation between Levin and Walton leads to accusations of incompetence from both sides, but stings Walton particularly acutely; Levin refuses to accept her conclusions as she cannot program. This challenge forces Walton to learn as much as she can in a limited amount of time in order to retrace the possible errors in Levin’s code and thus find the bug herself.

Amidst all of the drama occurring at the company, both Walton and Levin are suffering through upheavals in their personal lives as well. Whereas these personal conflicts hit Levin particularly harshly, Walton seems to draw strength from the challenges presented to her. Both bury themselves in their work, but for different reasons. Ullman is able to paint an image of Levin's increasing feelings of persecution and isolation particularly well, allowing the reader feel the lack of oxygen; the constriction brought on by the mounting tension and stress.

Levin's boss is a bearded, avuncular man named Harry Minor. He was a man brilliant in his own right, having helped to create Internet Protocol; a man who could easily fill the role of the cool uncle. Minor, a veteran of the computer software world, had been around long enough and had seen enough to understand the cold realities in which they all existed and had learned to cope with such pressures in his own way. In many ways, he was Levin's opposite and could have been something of a mentor had Levin not isolated himself. Instead, Minor winds up spending a good deal of time trying to help Levin cope with the increasing pressures brought on by the inability of Levin to locate the error in the code and rectify the problem.

Levin's one oasis of calm and stability in his day is lunch. Bill Steghman, another programmer who seemed to carry the weight of the world on his stooped shoulders as well, would stop by Levin's office and ask his usual one word question, "Lunch?" The daily routine becomes something of a comfort to Levin even if it remains bereft of any real human contact. The two men talk only of surface topics; questions remain short and answers shorter still.

Amidst this entire work world, was Levin's crumbling personal life. His live-in girlfriend, Joanna—more attracted to the arts than anything Levin is involved with—makes an unlikely partner. Largely due to this incompatibility and Levin's singular focus on his work drive a wedge between the two which winds up predictably splitting the two apart, further causing Levin's downward mental spiral.
In the end, The Bug is more a book about the responses of two people who find themselves placed in similar circumstances. The seasoned, professional not used to failure suddenly finds himself surrounded by events swirling out of control. In Levin's regimented world, this chaos is more than he can bear and after both his personal and work life implodes, so does he.

By contrast, the character of Roberta Walton sizes up the challenges presented to her. "Look, they're not quality-assurance engineers. They're not engineers at all. They can't even read code." Ethan Levin had pointed out. Faced with this realization, Walton sat down and did just that; learn to read and write code. Though Walton also experiences the same level of personal travails as that which dog Levin, she does not suffer the same psychological deterioration as does Levin.
It is this contrast that makes the novel compelling. While the errant code creates the stage, it is the tension which it creates which is the heart of the novel in the same spirit as any suspense thriller. The resolution of the problem—the cause of The Bug—then comes as something which only a computer "techie" can appreciate.

Ullman's first person narrative gives the book a sincere and realistic quality and her richly fleshed out characters allow the reader to instantly connect with at least some of them, however I have to question whether the gratuitous use of profanity is really a part of their professional culture or literary embellishment.

While Ullman delves into the world of programming, a reader neither has to be a programmer nor even familiar with the profession to understand the point of the book. While the true magnitude of some circumstances confronting the various characters will be lost on non-techies, it doesn't hamper the effectiveness that Ullman strives for.

Marie desJardins says

A book about computer programming and debugging -- what could be cooler?

Unfortunately, for me, it just didn't hang together that well.

The characters are so stereotyped, I got tired of the "you can only program if you're obsessive-compulsive and antisocial" theme. Even Berta, who starts off kind of normal, turns more and more antisocial when she starts learning how to program.

The biggest problem, for me, is that the bug that eludes them throughout the book just shouldn't have been that hard to find. Seriously. I did think the reason why it eventually became easier to find was clever and actually made sense. But the main programmer is just hopeless in how he goes about looking for it. He basically didn't try *anything* except repeatedly walking through the code. Over and over again. For, what, a hundred pages or so? It just became really tedious.

I did enjoy the bits of computer lore and scattered fragments of code. Somebody who hasn't programmed (much) might actually like the book more, if they're willing to learn something new. It's a cute idea, I just didn't really like it all that much.

Sean Randall says

"a kind of obsessional energy that was nonetheless pleasingly addictive. As the examples and assignments became harder, I began making errors, having trouble getting code to compile, link, run. Yet this trouble only drew me in, created in me a fierce determination to get it working. I had never before built anything—not a tree house, not a soapbox racer; I'd never even been able to finish a woven pot holder. I was bad with my hands and I lived in my head, and for me there was only one way to build something: programming."

This paragraph certainly sounded very familiar. The whole atmosphere of the work is gripping and thrilling on a couple of levels. We've all, of course, personified bugs in our code, but to see the effects of that on a whole team of coders and testers is quite scary. Then to look back at the systems and tools of the day and to see actually that the testing procedures still work today. TO see another ploughing through Kernighan and Ritchie's book with such evident enjoyment was gratifying, and the last chapter was almost elegiac. SO not a book for everyone, as is so often the case, but with a splendid couple of characters, a dollop of nostalgia and a glimpse into the shadowy world of the computer program too.

Trish says

It's an engrossing depiction of the early days of computer technology and computer start-ups -- similar to Plowing the Dark in the accuracy with which is captures the thought processes, foibles, and lifestyles of those we call geeks. The Bug focuses on two employees of a database start-up: Ethan Levin, a prickly programmer with a neurotic sense of inadequacy and a spiralling personal life, and Roberta Walton, a refugee from academia who first scorns and then embraces the arcana of the computer.

Programming is always an iterative process: code, check, re-code, check, repair code, ad infinitum. Bugs are standard. But there's something almost spooky -- something malevolent -- about bug UI-1107. A bug first spotted by Roberta and assigned to Ethan, a bug that crashes the system in a spectacle of bleeps and smeared pixels, a bug that seems to delight in capering before venture capitalist and potential customers, an elusive bug that evades all efforts to pin it down. Will Ethan find and fix the bug, or will the obsessive quest to track it down destroy Ethan?

Holly says

I was surprised (and pleased) at the extent to which Ullman presages Jaron Lanier. I cannot now recall if he mentions her in his writing, since I wasn't familiar with Ellen Ullman when I read *You Are Not a Gadget* nor with the New Yorker article. Just going to quote two nice passages near the end so that I can come back and read them later:

> ... there is the problem of crossing the chasm between human and machine "thought": some fundamental difference in the way humans and computers are designed to operate. I understand the world by telling stories; the human mind makes narrative, this happens then that, events given shape so we can draw a circle around them, see them relate, cohere, connect. We're built to tell stories to one another, and to be understood. But the computer was built to do, to run. It doesn't care about being understood. It is a set of machine states - memory contents, settings of hardware registers - and a program, a set of conditions that determines how to go from one machine state to the next. [...] So reading the code was a matter of banishing the human story from my mind.
>
> ...my perception of the machine had been changed forever. I knew then it was just an approximation, a fudge, a best-case work-around on the intractable problem of time. The machine seemed to understand time and space, but it didn't, not as we do. We are analog, fluid, swimming in a flowing sea of events, where one moment contains the next, *is* the next, since the notions of "moment" itself is the illusion. The machine - it - is digital and digital is the decision to forget the idea of the infinitely moving wave, and just take snapshots, convincing yourself that if you take enough pictures, it won't matter that you've left out the flowing, continuous aspect of things. You take the mimic for the thing mimicked and say, Good enough.

Emily says

The Bug is a novel that my father gave me for Christmas last year; I put off reading it until the summer because I wasn't sure I could read it in good humor while still taking a programming class.

The two novels it most reminds me of couldn't be more different. It's like Microserfs in the way it chronicles the social fabric of a technology project--the collaborations, rivalries, and moments of shared insight. But it's much more literary than Coupland; it also reminds me of Netherland in the way it plumbs the unique thoughts of someone who is falling apart in the echo chamber of their own head.

The Bug focuses on two figures. The narrator is Roberta Walton, a linguistics Ph.D. from Yale who, having

not found rewarding work in academia, grudging takes a job as a tester at a software company. Over the course of the project, she becomes obsessed with a bug that she discovered, and delights in learning C in order to track it down. But more of the pages are devoted to Ethan Levin, a programmer at the same company. It is his job to fix the bug Roberta found, but he can't. He's a computer science Ph.D. dropout and the bug provides a focus for all his preoccupation and self-doubt. The bug comes to feel like a portent of disaster in his personal life. The bug is a sort of character, too. It's a flaky bug that only seems to crop up in important presentations to investors or customers. No one can reproduce it to study it. When its definition and causes are finally explained, it is satisfying but also has a realistic sense of "Oh, for Pete's sake, it was just _____."

I especially liked the author's skill in explaining concepts like pointers and loops, right in the story, and the way Roberta turns her linguist's eye towards programming.

> I admit that when I first learned all this about computer memory allocation, I was disappointed--no, offended!--in a linguistic sense. Programmers were so inept at metaphor-creation, I thought. Memory leak: this wasn't a "leak" of memory at all. ... Why not name it to show the origin of the problem, with the programmer? "Memory gluttony," it should have been called. Or "memory hogging." Even the routine they used to request memory from the operating system had been named incorrectly. It was called "malloc," short for memory allocation, "m" "alloc." But of course human beings don't read "m" then "alloc" unless there is a separator, a space, a dash. No, by the implicit structures of the English language, everyone pronounces it "MAL-loc." Mal, loc. Mal: bad. Loc: Location. Bad location! But of course they'd have trouble keeping track of memory when they'd named their tool so stupidly!

This quote isn't terribly representative, but it struck me because that is my kind of tangent. Mostly the book is not technical, and has some great scenes and descriptions, like when a drunk person in an argument "felt his tongue start to slide around in his mouth like a bar of soap on a wet shower floor."

This was a very pleasing read for me, and the only reason I'm giving it four stars rather than five is that it has some twists that would lessen its reread value.

Katie/Doing Dewey says

Summary: A thoughtful, beautifully written, character-driven meditation on programming and humanity.

"In 1984, at the dawn of the personal-computer era, Roberta Walton, a novice software tester at a Silicon Valley start-up, stumbles across a bug. She brings it to its inadvertent creator, Ethan Levin...But no matter how obsessively Ethan combs through the depths of the code, he can't find its cause... Meanwhile, the bug...shows itself only at the least opportune times and jeopardizes the fate of the company. Under the pressures of his obsession with the bug and his rapidly deteriorating personal life, Ethan begins to unravel. Roberta, on the other hand, is drawn to the challenge. Forced to learn how to program, she comes to appreciate the intense intimacy of speaking the computer's language." (Source)

This is the third book in my project to read all of Ellen Ullman's books, as part of a larger project to read more deeply (in a more focused, thoughtful way) this year. I anticipated that Ellen Ullman's fiction would be very different in style from her nonfiction. Actually, this novel, as directly about programming as her nonfiction essays on the topics, had essentially the same strengths.

She captures what it's like to be a programmer, from the big picture feel of what it's like to program to the nitty-gritty of the programmer's daily life, in a way I've not seen anyone else manage.
She intimately relates the experience of programming and the daily life of the programmer - whether herself or her fictional characters. For instance, software tester Roberta is telling us about a day in her life where things went wrong and begins by explaining how she intended to try to find a problem with the software she was testing that day: "*A planned disaster was what I was looking for. A failure under my control. A perfect test.*"
She writes thoughtfully about the role of computers in society.
She uses the experience of programming and the analogy of computers to illuminate what it means to be human and uses comparisons to daily life to show what it's like to be a programmer. I really love that she makes me think differently about both these things by considering them together.

These strengths brought to the task of novel writing lead to great characters. The author is able to show their innermost lives, the most private thoughts, and connect them to their daily lives and their work as programmers. These strengths do not lend themselves to writing a particularly gripping plot. The story was slow and predictable. I was enjoying reading enough that I didn't want to put this book down, but regardless of what some descriptions may tell you, I didn't find this a novel of suspense. It's thoughtful, it's beautifully written, but the plot is not exciting. I'd recommend this to anyone who enjoyed her essay collections; to someone looking for a character-driven story who at least doesn't mind hearing a lot about programming; and, of course, to other programmers. I think you'll get a real kick out of it.This review was originally posted on Doing Dewey

---

Stephen Gallup says

I found this quirky novel on a table in the break room where I work, and once I opened it I couldn't put it down.

How could anybody not recognize and identify with this opening scenario/rant:

"And so we waited. Tick-tock, blink-blink, thirty seconds stretched themselves out one by one, a hole in human experience. Waiting for the system: life today is full of such pauses. The soft clacking of computer keys, then the voice on the telephone telling you, 'Just a moment, please.' The credit-card reader instructing you 'Remove card quickly!' then displaying 'Processing. Please wait.' The little hourglass icon on your computer screen reminding you how time is passing and there is nothing you can do about it. ... All the hours the computer is supposedly saving us--I don't believe it ... It has filled our lives with little wait states like this one, useless wait states, little slices of time in which you can't do anything at all but stand there, sit there, hold the phone--the sort of unoccupied slices of time no decent computer operating system would tolerate for itself."

Clearly, the author is someone well-versed in the intersection between human life and the cyber world, and what she has made of it here amounts to a very realistic horror story.

I didn't take the book back to the break room when I'd finished it. It's a keeper.

David says

I love novels that about life at an average workplace, because we spend so much of our time doing this but it is often ignored as a topic. This is a good novel but not a very cheerful one, because sometimes people and things go spinning out of control and all we can do is watch. This book is worth going out of your way to find and read.
