

**Gerard Holzmann is a Faculty Associate at Caltech in the Department of Computing and Mathematical Sciences and is the Lead Scientist of the Laboratory for Reliable Software (LaRS) at the Jet Propulsion Laboratory (JPL). He was recently part of a small team of NASA and JPL engineers commissioned by the U.S. Department of Transportation to study the possibility of software triggers for unintended acceleration in Toyota vehicles.**

## Ruling Out Bad Behavior: Designing Software to Make Extremely Dangerous Consequences Not Just "Unlikely" but "Impossible"

**ENGenious:** What inspired you to become an engineer?

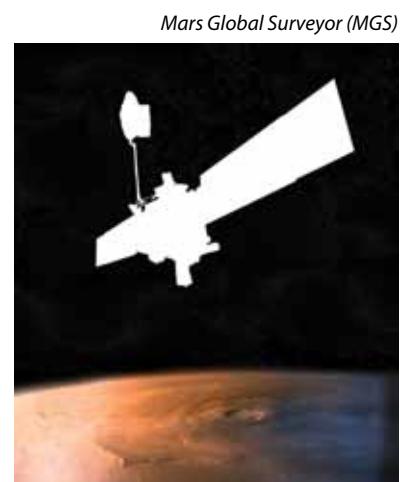
**Holzmann:** You can view engineering as the art of combining components in such a way that the whole becomes greater than the sum of its parts. This is an effort to strive for perfection: the illusion that we can build things that work perfectly all the time and that accomplish things that we as humans cannot. The most interesting part for me is that no matter how hard we try, the perfection that we aim for almost always remains elusive.

Engineering is interesting because it perpetually confronts us with the frailty of our understanding of how things work. A computer program, for instance, can be “perfect” in the sense that it will make a machine do precisely what we tell it to do, in precisely the order in which we tell it to do it. But almost inevitably things still go wrong, not because the computer misunderstands our instructions, but because we as programmers don’t always appreciate the complexity of what we are trying to do, which

means that we often get the instructions wrong in subtle ways.

**ENGenious:** Can you give an example?

**Holzmann:** A few years ago, NASA lost contact with the Mars Global Surveyor (MGS). The spacecraft had been orbiting Mars since September 1997. It all started with a regular maintenance action involving a minor update to some parameters to increase their precision. But the update for one of these parameters was off



Mars Global Surveyor (MGS)

by one word in the memory. This meant that this key parameter (and the one next to it in the computer’s memory) was corrupted and ended up having the wrong value. It went unnoticed at the time. Six months later, though, the solar panels’ positions had to be adjusted from winter to summer, but because of the first corrupt parameter the solar panels rotated too far. This automatically put the spacecraft in “safe mode.” Safe mode is programmed to have two priorities. The first is to be power positive—that means to make sure

that the batteries are always charged. The second priority is to maintain a communication link with Earth. Clearly, not doing so can lead to a loss of the mission. Since the solar panels were considered stuck, the only remaining way to point the panels at the sun to charge the batteries was to rotate the entire spacecraft, which was done automatically. As the spacecraft was charging the batteries, the fault protection system noticed that they were heating up. Typically, this means that they’re overcharging. So

the fault protection system decided that the batteries must be full and stopped charging. But the batteries were actually getting hot because the rotation that the spacecraft underwent in order to point the solar panels at the sun exposed the batteries to the sun as well. The fault protection system did not know this. To act on the second priority, the spacecraft had to point its antennas at Earth, but the Earth-pointing parameter was next to the soft-stop parameter for the solar arrays, and had also been corrupted in the earlier update. So the spacecraft was unable to find Earth as it tried to send out its calls for help. Next, the fault protection system noticed that the batteries had cooled off and were almost depleted—so it went back to its first priority. This cycle repeated a number of times until the batteries were fully depleted and the spacecraft became uncommandable. The curious thing is that the fault protection system was doing precisely what it was programmed to do, but there was this circumstance that nobody had thought of until it happened. How do you predict these things? Well, that is

very difficult, but it is precisely what makes this fascinating. You think you’ve covered all the possibilities, but you probably didn’t even scratch the surface.

**ENGenious:** How is JPL’s Laboratory for Reliable Software making flight software more reliable?

**Holzmann:** We started the Laboratory for Reliable Software when I joined JPL in 2003. It has the daunting task of trying to achieve long-term improvements in the reliability of the software we use to fly interplanetary space missions. So far, we’ve introduced the use of state-of-the-art static source code analyzers as part of the software development process at JPL. These analyzers can intercept a lot of common software defects that otherwise slip through. We’ve also



Gerard Holzmann

11100100110111101110101110001  
10011001110010110010101110001  
011011001000100001100101011011  
100001110100011100100110011001  
110110101100101011000111100  
00100000110010001100001011011  
0110111011001110011001100011100  
01010110010010110010101100110001  
X001011101010110010101100110001  
010101100110010000110111001101100  
00110001001110101011100  
0001001011000100110100011000110001

developed a new Institutional Coding Standard for all flight code developed at JPL, we initiated a new and more thorough code review process, and we’ve started a formal “certification” course for our flight software developers. We’ve made good progress in the last few years, but we don’t take anything for granted.

**ENGenious:** Tell us about being asked by the U.S. Department of Transportation and NASA to study the possibility of software triggers for unintended acceleration events in Toyota vehicles.

**Holzmann:** I was very fortunate to be part of the team of software experts that could work on this problem. I was asked to apply some of the techniques I developed for these types of problems in my years as a computing

science researcher at Bell Labs. We were given unlimited access to the source code that drives Toyotas and to the technical experts who could explain its working in detail. I learned more about the software controls in cars than I could have imagined. We immersed ourselves in this problem

*“...to make sure that unacceptable events are actually rendered ‘impossible’—and not just ‘unlikely’ ... we first have to recognize that no single part of a complex system is ever perfect, and that includes the software.”*

for about five months in 2010, working full-time at Toyota facilities in Los Angeles, and I believe we were able to complete a really thorough analysis of the code. The puzzle was the usual one: Can we find out how something that is not supposed to be happening might happen anyway? We were able to rule out a number of potential causes for unintended acceleration, although much of our analysis has not been released publicly. The complexity of an analysis like this immediately leads back to my original fascination with software complexity: it should be possible to design software in such a way that we can rule out bad behavior conclusively. My colleagues and I are today even more determined than ever to develop such a method for use in safety critical systems.

**ENGenious:** What are the main research challenges in reliable systems design?

**Holzmann:** The main challenge in reliable systems design is to make sure that unacceptable events are actually rendered “impossible”—and not just “unlikely.” To do this, we first have to recognize that no single part of a complex system is ever perfect, and that includes the software. The key is to build reliable systems from potentially unreliable parts. Nothing is foolproof. So we often try to

find a compromise between cost and benefit, but extremely dangerous consequences should be firmly placed outside such a cost-benefit analysis. Many have not yet fully embraced this approach, partly because it is tempting to interpret events with a very small probability of occurrence as virtually impossible. We only have to look at how nuclear power plants sometimes fail to see that extremely low-probability events are still very much possible.

**ENGenious:** Are engineering students trained well to design reliable systems? What, if anything, should change?

**Holzmann:** I think there are two possible answers to this. In most areas of engineering, the answer is yes. Civil engineers, for instance, can design a building or bridge to successfully withstand an earthquake of a certain magnitude. In software engineering, though, the answer is often negative. The prevailing belief is that the hardware has known failure modes, but that software can be perfect. The fault protection software onboard a spacecraft is designed to recover the spacecraft when a hardware problem strikes, but it is often powerless when a software problem occurs. The fault protection software itself, furthermore, can also be faulty or subtly incomplete. We should design

safety critical applications in medical devices, cars, power plants, and spacecraft with knowledge of the failure modes, including software failure modes. This is something that we are not very good at today.

**ENGenious:** One way of improving the reliability of systems is to have them tested extensively. Should members of the community participate in testing? Can systems such as OnStar help?

**Holzmann:** Direct measurement of the true performance of a system in practice is invaluable. It is how we learn the hidden flaws and what gives us the opportunity to adapt our designs to improve them. In a sense, all spacecraft that are currently active across the solar system have the equivalent of an “OnStar” button. Every time a spacecraft presses that button, so to speak, we learn something new about how the spacecraft we built yesterday works today and how it could be designed even better tomorrow. ■ ■ ■

*Gerard Holzmann is a Faculty Associate at Caltech in the Department of Computing and Mathematical Sciences and the Lead Scientist of LaRS at the Jet Propulsion Laboratory.*

Visit [lars-lab.jpl.nasa.gov](http://lars-lab.jpl.nasa.gov).

```
0100011101110110000101110010
011001010010000011101000110111100100
000011011010110000101101
01011001001
0010000011001000110000101101
10011001110 X1100101111001001101111011
1010101110011001000001100011
```

**ENGenious** sat down with Eva Graham, the Director of the Caltech Center for Diversity (CCD), to learn more about the center and how it has been serving the Caltech community since its creation in 2008.

## The Caltech Center for Diversity

Providing a Real Pathway to Membership for Underrepresented Students at Caltech

**ENGenious:** How was the CCD created?

**Graham:** It has been three years since the Office of Minority Student Education at Caltech was combined with the Women’s Center to form the Caltech Center for Diversity. In this area, Caltech was following a trend already established at several other educational institutes. It has not only been more cost-effective, but it also has given us the opportunity to reach communities that we may have missed before, because now we have overlap in expertise and programming areas. With the overlap, we are able to reach a woman who is African-American and Lesbian, Gay, Bisexual, Transgender, Questioning and Allies (LGBTQA). We are also down the hall from International Student Programs, which provides us with the opportunity to work with international students.

**ENGenious:** What has stayed the same?

**Graham:** We have maintained our programs for women and minority students. Their foci are the same. With women, we still focus on helping them access resources and become part of the community. Long before the CCD was created, the Women’s Center was doing a great job helping women understand their roles in science and engineering. I’m happy to say that now women are seeing themselves as part of an even larger community. There are larger numbers of women on campus, and we’re hearing less and less about inappropriate behavior toward women. Certain programs are as popular as ever, such as self defense, assertiveness, and programs related to being women in a laboratory environment (because they might still be the only woman in a lab). With minority students, the focus has always been on outreach, recruitment, and retention. We also focus on community building, because the numbers of minority students remain low.

**ENGenious:** What has changed?



Eva Graham

**Graham:** For the LGBTQA community, we’ve focused on providing a safe space for coming out and gathering. We continued a working group made up of students and staff. The students are representatives of PRISM, which is Caltech’s campus social group for lesbian, gay, bisexual, and transgender students with support from staff, faculty, and their straight allies. In addition, we have created a Safe Zone program on campus led by one of the CCD Assistant Directors, Linda Webb. The program is designed to help build community, increase awareness, and support a safe space for the LGBTQA community. Student members of the program have designed their own logo, which is displayed in the offices on campus that are safe spaces for LGBTQA students who are struggling or have questions. In addition, we hold monthly community lunches. By