Subject: ECE 1111: where do all those bad programmers end up?
From: Joseph Picone <joseph.picone@gmail.com>
Date: 6/27/23, 11:45 AM
To: ECE 1111 <temple_engineering_ece1111@googlegroups.com>

This is a somewhat amusing anecdote about an event that happened before last semester's class. From time to time I share these kinds of stories with you so you understand how important it is to be very meticulous in the software design and testing processes.

You are growing up in an era where users come to accept software is buggy and breaks. You just move on to the next app. In some applications, this is fine - like shopping. In other applications, however, like transportation, defense and healthcare, software bugs can cause fatalities or other disastrous outcomes.

We stress careful testing of your code in this class. Penalties for submitting untested code will be huge.

-Joe

=====
Subject: Re: where do all those bad programmers end up? - Part II
Date: Thu, 5 Jan 2023 15:51:00 -0500
From: Joseph Picone <picone@temple.edu>
To: nedc_research <nedc_research@googlegroups.com>, ECE 1111
<temple_engineering_ece1111@googlegroups.com>

Now, here is the other side of software design and testing...

I recently bought a 2023 truck. My insurance company uses a dongle that plugs into the OBD II port on a car and monitors your driving. It is actually a pretty cool device - it communicates with the insurance company web server via a wireless connection and uploads all my driving data. Everything is visible from the web - even those spine-chilling abrupt stops on Roosevelt Blvd. to avoid accidents.

Because I don't drive much (due to my love of SEPTA 😄, I save tons of money because I pay per mile and I am a defensive driver. Ironically, these per-mile plans were designed for your generation - the so-called zoomers who love their phone apps and ride-shares.

When I bought the truck, and called to change my insurance, I was told they can't issue a policy because the dongle hasn't been tested on 2023's yet. So I have to pay a lot more for insurance - about $1000 per year 😕 That made my new truck purchase much more expensive!

Now, what is good and what is bad about this story.

On the one hand, I applaud the insurance company for taking software testing conscientiously. Instead of assuming things would work, they have decided not to release support until they have tested their devices on the new models. Understandable... Of course they rely on a subcontractor for this, so in reality they need a better subcontractor.

But... 2023 models have been available for at least 6 months. The OBD II port uses an industry-standard communication protocol that hasn't changed in a long time. If their software is designed properly, it should work because the interface hasn't changed. Lagging the model year by what is probably going to be a calendar year is not a sustainable business model - either light a fire under your testing organization, or outsource to someone who can be more responsive.

The latter is why hungry Internet startups scrapped a lot of rigorous software testing - too slow and expensive. Today, companies use what they call agile development paradigms which often means, in reality, customers end up testing software for the developers. We see this in our college where we have purchased packages such as AEFIS and Retain that are extremely buggy and under constant development while being sold as product. Google famously also followed this approach with the development of most of their tools like Chrome. The software is free, so the users have to live with bugs.

Microsoft and IBM, on the other hand, who support a lot of traditional business users, still do a

lot of internal testing before release. Boeing, for obvious reasons, has to thoroughly test their flight software before it goes into production.

It is a challenge to be profitable developing software and yet be very conscientious about testing. If you pursue careers in engineering that involve software or embedded systems, you will experience this dilemma first-hand.

In ECE 1111 and ISIP, we do emphasize testing and debugging. We emphasize debugging as you write the code - it is an integral part of the software development process.

-Joe

==========
On 1/3/23 5:29 PM, Joseph Picone wrote:

> Perhaps the most important part of programming is debugging. To debug code you have to think of all the important test cases - what we often call edge cases - that might break your code. It only takes one fatal bug to cause a disaster.
>
> So let me begin with an interesting anecdote:
>
> Some of you know that I am a big fan of public transportation. Like Sheldon in The Big Bang Theory, I especially enjoy trains 😉
>
> Last night, I traveled home from the airport attempting to use public transportation. I normally take a train and then a bus. The SEPTA app reported that buses and trains were on holiday schedules yesterday, and their wonderful Next To Arrive app wasn't working properly. It showed that no buses were running close to my house.
>
> Since I arrived at 5 PM, there was no chance to catch the last #28 bus. It was hard to get home otherwise. I ended up taking the RR to Fern Rock, the #70 to Cottman and Algon, and then walking home (1 mile with luggage - not fun 😄
>
> Just as I arrived home, a #28 bus arrived at my house. I was <expletive deleted>. Supposedly, they weren't running, but they were actually running after all...
>
> I called SEPTA this morning. Turns out it was a bug in their system. The buses were on normal schedules while the trains were on holiday schedules. They spent all day trying to fix it (supposedly) but couldn't.
>
> How does this relate to ECE 1111 and coding in ISIP?
>
> I bet someone who programmed the app never expected buses and trains to be on different schedules. They probably just assumed both would always be on the same schedule. This is what we call an edge case.
>
> The buses it turns out were on a normal weekday schedule even though Jan. 2 was technically a holiday. The trains, however, were on a holiday schedule. That is a test case they obiously never thought of. SEPTA said they spent all day trying to fix it, but my guess is it was a fatal design in the software.
>
> As a programmer, you have to put a lot of thought into edge cases. This involves an intimate understanding of the application space, the algorithms and the software.
>
> ECE 1111 Students: We are going to stress debugging in this course. You have to be very meticulous about debugging every line of code you write! Do not turn in buggy code - you might end up with a score less than 0 on an assignment or exam (yes, this has happened!).
>
> -Joe