Society has forever changed since the adoption of computers and software development. Seems like a very valid statement to programmers in the 21st century, right? Fixed iterations, maintainable software, and software which even others can understand are just a few of the hallmarks of today’s industry; it’s hard to imagine a life without such principles, but I dare you to put yourself in the shoes of Bob who like the enthusiastic developer of his time was eager to toil through the days working on his projects. Working throughout his college years with emacs, Bob had not a care in the world. Well, his mostly peaceful life was about to change…

During the year of 1979, Bob graduated and began working at the Atomic Energy of Canada Limited which was a company which he felt would help many individuals of the world. Bob had heard rumors of a machine which was able to intelligently provide radiation therapy and suggestions of suitable doses for a patient. It was one of those projects which was kept away from the public and only the best and brightest minds of the age were able to work on it. Of course, our friend Bob thought he might have a chance, after all, all those years with emacs by his side surely must have landed him the job. On his first day, Bob entered the building and was greeted by Alice, a fellow programmer. She said that the Therac-25 was going to be one of the best machines this world will ever know. “Bob, there is still much work to be done, so the more helping hands, the better,” she proclaimed. Being on his first job, Bob understood that it was important to keep an inquisitive eye on the programming environment, but at the same time, he needed to be friendly to his fellow programmers even if he would talk to them only on occasion. “Could you show me to my desk?” Bob inquired. Alice knew that such a brief conversation was expected by a fellow diligent programmer, which was perfectly in-line with the
company’s moto: “An isolated introvert is a diligent programmer.” Alice showed Bob down a dark hallway to a dimly lit room with an empty cubicle in the far corner. As they made their way down the aisle, Bob peered at the other cubicles, which seemed more like miniature compartments fully enclosed on all four sides. Each had a small door but only had space for a single developer and were practically sound-proof. As they neared Bob’s compartment, a sign hanging from the door knob came into view. “Empty” was all it said in large bold letters. Alice handed him the key and Bob gingerly took down the sign and opened the door.

He was greeted by a single computer which was already had a development environment set up. “If you need anything, I will be in compartment 201A”, Alice said even though she knew it was very rare to actually have someone show up. Excited, Bob started perusing the existing code base, it had many todo statements which he would focus on after understanding the existing code. Comprehending what was already written was very difficult; it seemed a chicken had typed on the keyboard (if that was even possible). It was only his first day, and he thought it would be best not to pester anyone else. So, instead of trying to understanding the existing code, he focused his attention on the commented portions and assumed that the existing code was free of any errors. The comments were somewhat more clear and large functions had a brief description about what they “did”. Other than the code, another page was open which provided documentation about the system. This, unlike the program, was very well written and explained the system components initially at an abstract level. Bob wanted to tackle the todo statements right away, but took a step back and focused his attention again on the documentation, reminding himself that others would appreciate his efforts if he used a systematic approach.

He first began by writing out a schedule for the rest of the week. He would focus first on reading the documentation which consisted of a few 500 pages or so. If he did not understand
something, he could simply fill in the gaps by systematic interpolation based on prior pages.

Over the course of a week, he set aside his programming skills and read pages and pages of documents. The more he “learned” about the system, the better. At the end of the week, he was only on page 20, but he justified his pace by acknowledging that the documents were a dense read and that he had to use a few days to grok important aspects of how the radiation dosage was being measured. The system was what many mentioned to be life-critical and so if you needed a few months to get up to speed by reading, then that would be best. Not to bore the 21st century reader who hopefully knows better, let’s skip forward two months when Bob has finally finished reading through the documentation. He now can focus on the todo tasks which he expected to have diminished over the past two months. Unfortunately, the code seemed to have gotten sloppier and the initial todos were replaced with new ones as if each was analogous to bacteria in a petri dish. He focused on the first one and read through the comments outlining the task at hand. Since he had read through the documentation, he understood the technical jargon with ease and eagerly started writing the first sliver of code. The inquisitive reader would probably have noticed how the last statement did not consist of “the first sliver of tests.” Bob knew that tests were needed, but only after the actual code was written. Clearly, there would be nothing to test without any code and he did not want a horrendous screen of errors.

Over the next few years, Bob worked tirelessly on the todo statements, writing code here and there, trying to increase function reuse by incorporating several goto statements. Finally after two more years, the Therac-25 was ready for production. Several machines were deployed both in Canada and the US. Bob, being able to contribute to the machine’s software was very pleased with himself and felt that if it wasn’t for him, there would still be many issues. He was so sure of himself, he didn’t bother testing the software as a whole, but did test subcomponents he
was assigned to. Between the years of 1985 and 1987, there were cases of massive doses of radiation which was provided to some patients. Many patients received radiation burns and poisoning. Bob became very worried. He reassured himself that the issues were probably due to someone else’s work. “Some careless programmer probably did not read all the documents or did not test his/her code (surely Alice),” he thought. Nevertheless, after hearing the news, Bob left Atomic Energy of Canada Limited. Was Bob to blame? I’ll let the intelligent reader decide, but we can agree on this much. Software development has surely seen a major change in code quality, practices, and tools, but to assure that Bob and Alice’s story is never forgotten, their names are frequently brought up through the course of time.