Stages of amateur software development

FATIUL HUQ SUJOY

As foretold by our forefathers since age forgotten, a CS student must bring forth to the world a software. It is as essential to their being as the countless sleepless nights of debugging, using unsolicited programming puns and a failing social life. But it's the stages of developing it that are truly worth the printf().

THE INCEPTION

The beginning is always a spark of genius, or necessity. It could be the brainchild of an enthusiastic individual ready to change the world, a weapon to win fame and glory in competitions a grand total of two people have heard of, an idea from which a colossal million dollar startup will obviously emerge, or a course project forcefully assigned with a deadline systematically devised to induce trauma. But whatever the reason is, it sets in motion events no one was ever really ready for.

TEAM FORMULATION

The worst part about forming a team is when your initial concept of an A Team containing the best of all worlds shatters. At the end of building the team, you'll find that the ace coder, the mad mathematician, the artsy designer, and the crude tester you dreamt of are replaced with your buddies sharing programming memes.

PROJECT MANAGEMENT

This is where the dreams of professional software engineering collide with amateur programming. One or more eager team members decide that it is of utmost importance that a full-fledged automated project management tool be in place for the three people working. International standards are advised to be maintained in all collaborative artefacts and codes to make it maintainable for its thousand years of glory. It is, of course, soon botched up, usually in a matter of two days. The reason, apart from complete redundancy, is the reluctance of coders to use a few more brain cells in anything other than coding.

CONFLICT RESOLUTION

It was a sneaky little bug or a new feature, but after toiling for hours, you finally finished it. You push it to the online repository. As you hear the birds chirping in celebration of a new day, you fall asleep, content with your work and life. The next day you resume coding, still giddy from last night's success, and pull from the repository. With a twitch in your eyes and a sinking feeling in your guts, you find some or all of last night's code gone.

"Look, it showed 'merge conflicts' when I tried to push," says the culprit, busted by the version history, "And I decided to disregard the previous code because my intelligent judgement concluded, without any knowledge of the conflicting code, that it was unnecessary."

DESERTION

At some point during development, usually near the end, a bout of impatience channels through the team. The weaker ones fall victim to it. They find the project unworthy of further effort and decide that abandoning it altogether is clearly the smartest of all possible choices. Although all they need is a little time off and a calm lecture on their misconceptions to get them back in line, the benevolent leader finds it more appropriate to verbally abuse and ruin friendships indefinitely. And sometimes that works out.

THE LAST FEW DAYS

With the deadline in a week, it becomes apparent that more than half of the work still remains, which amounts to at least two years of effort. It's panic hour. Everyone starts by saying goodbye to family and sleep. They pin all the necessary StackOverflow tabs and play YouTube tutorial videos on repeat. The first aid kits are placed nearby in case of finger injuries from all that high-speed coding. After a few hours a few unconscious bodies are found of weaklings who couldn't make it. The rest carry on. Muscle memory is doing most of the work now. The last few remaining have ceased to be functioning humans. Most of them have decided to take on the farming life after this. Others question the meaning of their existence. But in the end, at long last, the coding stops, all the features have been ticked done, the compilation succeeded, everyone celebrates with groans and tears.

DEPLOYMENT

Everything's gone astray. There wasn't any testing done on the software, and it's failing at every turn the coders haven't considered. And even if it's functioning, the clients aren't satisfied. "Where is this impractical feature I never mentioned before?" they demand. In case of competitions, the evaluators are unimpressed. "Why does this software built by students not contain every state-of-the-art technology available and unnecessary machine learning algorithms?" they ask, fuming. In case of coursework, the teachers negate marks for the absence of features they never wanted in the entirety of the project's lifetime but just thought of now.

MAINTENANCE

Regardless of the outcome, it's been a wild ride and the team have with them a software that has potential to become a grand venture, or at the very least, a formidable addition to their résumé. All they need to do now is maintain the code, add some patches and tweaks, and keep it alive.

Three days later, they revisit it. They've successfully forgotten all the code. When run, it's showing 93243324 errors. Most of them can't even open it on their personal devices anymore. They all move on to better things in life, like nine-to-five jobs. The software remains in the online repository, abandoned and forgotten.

Fatiul Huq Sujoy is waiting for that sweet release of public holidays to take him to exotic locations with high throwback value. Suggest him books to read during travels to make him look cool at s.f.huq11@gmail.com