Using Agile Process Methods in Virtual Software Engineering Course

Lessons Learned

Tarik Hrnjić

Core values of Agile Manifesto

1) Individuals and interactions over processes and tools.
2) Working software over comprehensive documentation.
3) Customer collaboration over contract negotiation.
4) Responding to change over following a plan.

Difficulties in virtual classrooms

- Lack of “hands-on” experience
- Reduced interaction with teaching staff
- Lack of student engagement
- Difficult to conduct exams

Sprints, not marathons

- Key principle of agile process is the incremental product development.
- Whole semester long project is a marathon.
- Dividing marathon project into week long sprints enables us to change & adapt.
- Every week students present “ready-to-ship” application (deployed on the server, installed...) to the “customer” and get feedback.

Lesson learned: The importance of weekly sprints

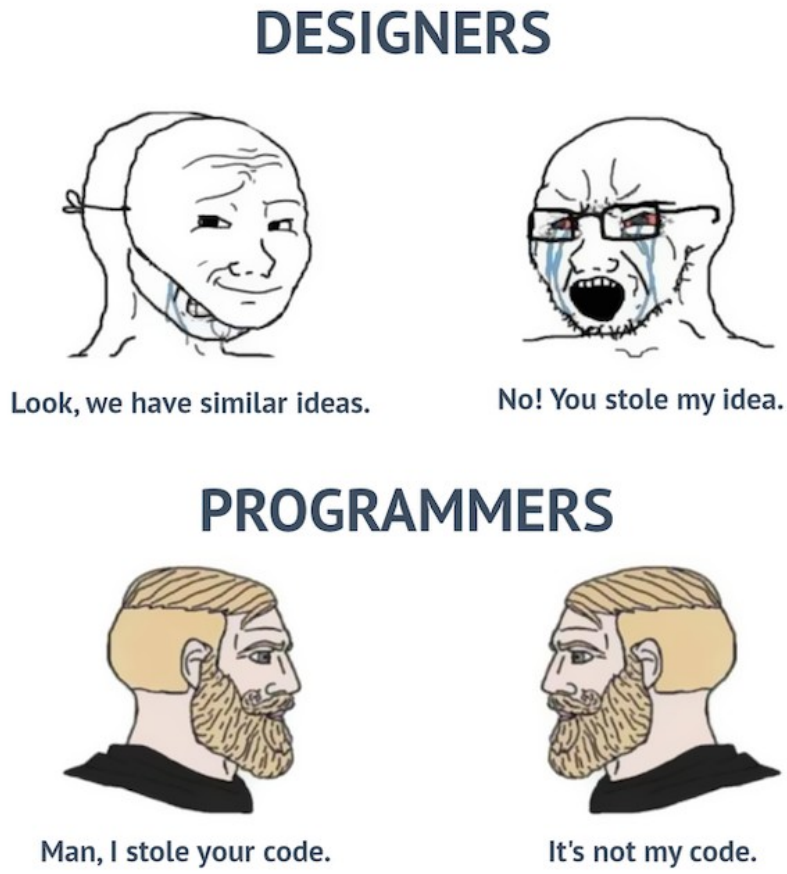
- Students meet every week with the “customer” to report the progress and discuss new user requirements.
- Must have something to show to the “customer”. If the “customer” notices that certain features are not successfully implemented after several sprints, requirements may be slightly altered (we might be expecting something that is too complex).
- The idea is to establish rigorous framework that encourages continuous work from each student (ideally every day, in practice several days per week).

Lesson learned: Increase student engagement by offering interesting project(s)

- In prior courses (Intro to programming, OOP, Web dev...) students have done store/library/*insert similar generic programming project name here* multiple times. In those courses, learning outcomes are different compared to Software Engineering course.
- The idea is to offer a project that is more complex and engaging. We want students to have to research how to do certain tasks.
- Example: “Snapshot” application

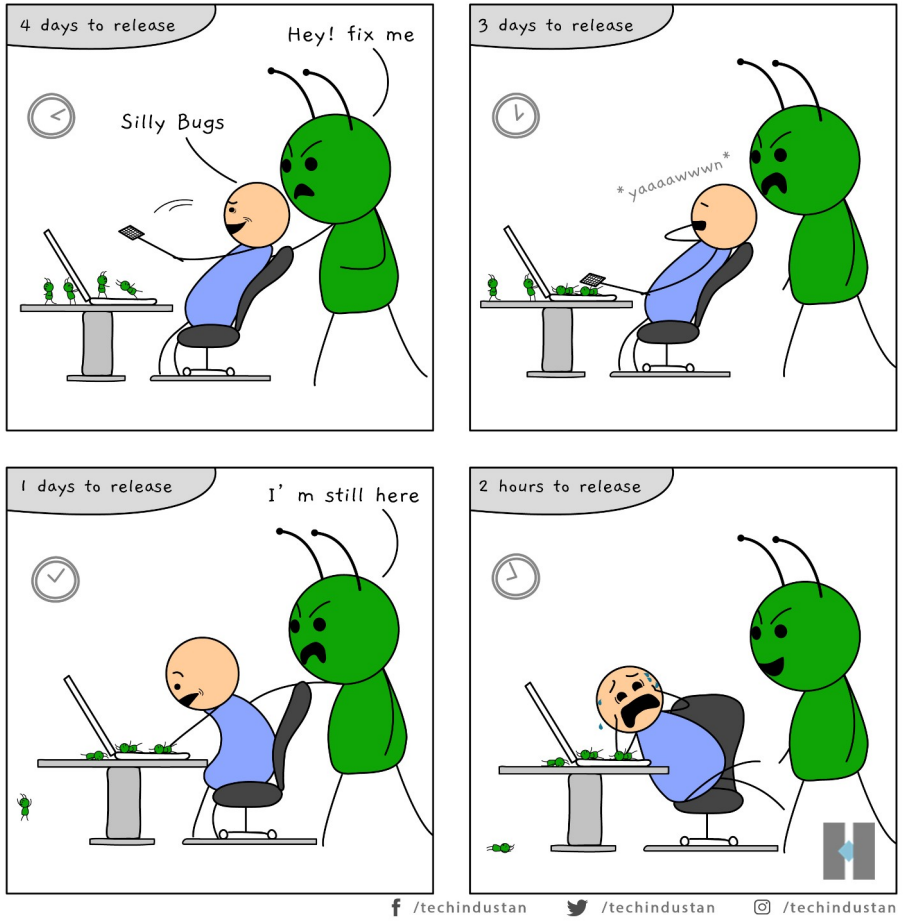
Lesson learned: Do not offer immediate solutions to problems/conflicts – personal issues

- Initially, students come to us with questions about implementation details related to the task at hand (“How can I do this?” “I didn’t learn this anywhere.”).
- Important to stress to students that this is not the course where we teach programming, but software engineering; and that numerous online resources are available that address similar, if not identical, problems.
- At this educational stage, the primary response to a problem should be independent problem-solving rather than seeking immediate assistance from the teaching staff.



Lesson learned: Do not offer immediate solutions to problems/conflicts – team issues

- Each team has a leader chosen by all team members.
- Work (user stories) is assigned to the team. Team members are expected to volunteer to take on certain tasks.
- In the case of conflict (multiple team members want to do same task, no team member wants to complete certain task) team leader is expected to resolve the conflict. Each student should be able to perform every role (coding, testing, writing docs…).
- Very rare that the student does not show any engagement during the entire semester.



Lesson learned: Do not offer immediate solutions to problems/conflicts – cross-team issues

- Most common problem (especially in the first few sprints), develops as multiple teams work on different parts of the same software and do not communicate properly (front-end team does not receive necessary routes from back-end team).
- Crucial to emphasize to all teams that they are collaborating on a single software product, which must align with the “customer's” expectations, rather than merely being a collection of disparate parts.
- Grading is done based on the “customer’s” satisfaction with the software product.

What we noticed

- Initially, reluctance to produce results every week:
  - “Can we have two week sprints?”
  - “Give us enough work (user stories) for two weeks, isn’t it the same thing in the end?”
- After several (2 or 3) sprints students adjust to the work-flow:
  - Important to not crumble and give in to student complaints early

What we noticed

- Improved time management:
  - The importance of time management and prioritization is key component in agile methods.
  - Initially, effort/time estimates are highly inaccurate, leading to (usually) only partially delivered sprints.
  - By the end of the course, students learn to manage their time effectively, they know what to prioritize. Scrum poker is more precise and effective.

What we noticed

- The feeling of the project ownership is important:
  - With every successive sprint, students become increasingly invested in the project.
  - Towards the conclusion of the course, at the weekly “customer” meetings, students start offering suggestions on what can be improved in the application. They are willing to put in the extra effort to make it the best it can be.

What we noticed

Thank You