An overview of key aspects in adopting Scrum in teaching process

Boris Milašinović
University of Zagreb Faculty of Electrical Engineering and Computing

Outline

- Motivation for the presentation
- Author's context
- Some excerpts from scientific literature on methods and problems of using Scrum (and agile methods in general) in education
- Conclusion

Motivation

Happy "Scrum in education" papers are all alike; every unhappy paper is unhappy in its own way.

- Quest for the holy grail in teaching software engineering process in own environment
  - many attempts and many failures, but some progress had been made although it would never be perfect
- Enumerate some of the suggestions and possible pitfalls to successful introduction of agile methodologies in software engineering education
  - focus on Scrum as most dominant, or at least the most trending agile methodology

Need for change of required competences

- Traditional teaching is based on theoretical fundamentals supported by hypothetical examples
  - practical experiences in real life projects gives students distinct advantage in competitive and changing market
- The problems frequently occur in the planning and managing phase rather than in the developing phase, or as a failure of development responsibilities
- Some other skills beyond programming and technical excellence is needed
  - those soft skills are not always easy to learn or acquire
  - appropriate ecosystem must be created to avoid teaching agile values without being experienced in practice

Problems in adopting project based learning

- Educational institutions often not allowed or able to change their curriculum
  - various accreditation procedures
  - lack of staff
  - absence of interest or the knowledge to change
- A typical workaround: adapt current courses by introducing real-life problems

Author's context – When?

- Course *Software Design Project* at bachelor level
  - 5th semester up to 10 students per lecturer (future thesis advisor)
- Course *Development of Software Applications*
  - 6th semester, usually 100 students enrolled
  - lack of teaching staff (one lecturer, sometimes no assistants)
- Couse *Project* at master level
  - up to 10 students per lecturer
  - students have good development skills but the course is part of the sequence: *Seminar – Project – Master thesis*
    - used by students to create a prototype for their master thesis
      → teamwork would be counterproductive

Author's context - The first unsuccessful attempt (1/2)

- Course *Development of Software Applications*
  - 6th semester, 100 students, one lecturer, one or no assistant
  - extract requirements from an interview (real user is emulated)
  - develop an application implementing users requirements
- Agile addition: write user stories and divide them to smaller tasks as the development goes further
  - similar to others work but experiences with attempts to introduce Scrum were (somehow) contrary to results shown in reviewed scientific papers
- All key aspects for failure satisfied ☹
  - lack of teaching staff as a trigger for many other problems

Author's context - The first unsuccessful attempt (2/2)

- All groups do the same project and only their implementation was different
  - user stories can be copied among groups
- Progress of development heavily constrained by teaching development process (technology)
  - choosing what to do in which iteration/sprint was not the students' choice but the side effect of the teaching progress.
- Each student should learn every stage of a software lifecycle
  - everyone felt that entering stories and tasks were unnecessary additions to the development process
  - user stories subdivided in tasks after the development had been done, just to earn grading points

Author's context - The second unsuccessful attempt

- *Software Design Project* course at bachelor level (5th semester)
  - up to 10 students per lecturer
- Students not ready for this level od independence
  - I could not risk leaving the Scrum role to an unexperienced student, and I had no time to teach her/him to become Scrum master.
- Schizophrenic crisis of identity and enormous time waste
  - I was Scrum master, Product owner, and (sometimes) lead developer
    - they have to learn how to manage development process, but they do not have the development skills (at least many of them)
    - enormous waste of time for preparation of tasks and for teaching students some development tasks.

(What) can we learn from others?

- Where and when to introduce agile methodology and related problems?
  - Most of authors suggest a capstone project as an ideal place for teaching agile methodology.
  - However, there is no unique view on how long it would be and how would it be organized, and who would do which role
    - Lecturer, student, rotating roles, assistants, lecturer outside the team, agile coaches…
- Common issues
  - lack of training, resistance to changes, problematic teamwork, administrative effort, …

Students motivation and attitude to (agile) methodology

- Diverse students perception about change process, e.g.
  - some are very enthusiastic about methodology practices
  - some feel that things like project management are not important, or are applied just for lecturer's sake
- Usually better students are more aware of the benefits
  - although even excellent coders can cause problems by underestimating the importance of soft skills, or by having a bad attitude to technically less proficient users
- Tendency to follow waterfall-like plan rather that respond to change

Student preparation

- Scrum is not solution to the problem by itself.
  - Scrum as a concept is relatively easy to understand, its adoption and correct usage can be very hard.
- Some methods of preparation
  - prepare in advance (e.g. 4 weeks) *[Martin et al. 2017]*
  - soft skills can be taught in anticipation of potential problems *[Burris 2007]*
  - first observe existing teams for a week and only then start to gather requirements *[Potinenini, Bansal, Amresh 2013]*
  - uses initial zero Sprint as an introduction to Scrum *[Mahnič 2015]*
  - spend at least two sprints for students to adapt to Scrum *[Freitas et al. 2017]*
  - some other approaches using games (e.g. planning poker, LEGO bricks)
    - an alternative to practical work in case there is not enough time or skills for development [review by Mahnič 2015]
    - a game with a ball could improve development duration estimation [May et al. 2016]

Some aspects that usually do not occur in real world

- Team size not chosen by the needs, but by the enrolment process
- No common working place and different schedule cause meeting problems
- Students are usually distracted with some other activities
  - Part time jobs, personal interests, activities, or problems
- More likely to take a sick-leave or even quit the project (fail the course), perhaps more often than an employee resigns.
- Motivation problem
  - students' only "salary" is their course grade, thus many students are only interested in grades and deadlines than the software quality.
    - an interested suggestion from [Murphy 2017]: in the second course of two courses sequence a student must continue working on others work, thus raising awareness of importance of a good code.
  - if not obligated (and graded), usually they would not prepare in advance
- Working in an (Agile) team can mask individual contribution and individual work must be recognized and valued appropriately
  - focus on grading and on individual tracking rather on a product itself

Conclusion

- Cannot clone others' solutions
  - no unique opinion on many aspects
  - students attitude varies by country/part of the worlds
  - significantly less staff
- Paradoxes and inevitable problems
  - catch up lack of development skills
  - Students ≠ Employees
  - teach large number of students all aspects of a software lifecycle by emulating teamwork, but in real teams roles are usually strictly divided
- Pick the things that could suit in own environment, avoid commonly known mistakes, and improve by learning from your own unique mistakes
  - Better to try and make a mistake, rather than doing nothing (expecting that no one could blame you for mistakes)

(Some of) references

- K. A. Alshare, D. Sanders, E. Burris, and S. Sigman, "How Do We Manage Student Projects?: Panel Discussion," *J. Comput. Sci. Coll.*, vol. 22, no. 4, pp. 29–31, 2007.
- L. Freitas Santana et al., "Scrum as a Platform to Manage Students in Projects of Technological Development and Scientific Initiation: A Study Case Realized at UNIT/SE," *J. Inf. Syst. Eng. Manag.*, vol. 2, no. 7, pp. 1–7, 2017.
- V. Mahnič, "Scrum in software engineering courses: An outline of the literature," *Glob. J. Eng. Educ.*, vol. 17, no. 2, pp. 77–83, 2015.
- A. Martin, C. Anslow, and D. Johnson, "Teaching Agile Methods to Software Engineering Professionals: 10 Years, 1000 Release Plans," in *Agile Processes in Software Engineering and Extreme Programming*, 2017, pp. 151–166.
- J. May, J. York, and M. Lane, "Play Ball : Bringing Scrum into the Classroom," Jourrnal Inf. Syst. Educ., vol. 27, no. 2, pp. 87–93, 2016.
- C. Murphy, S. Sheth, and S. Morton, "A Two-Course Sequence of Real Projects for Real Customers," *Proc. 2017 ACM SIGCSE Tech. Symp. Comput. Sci. Educ. - SIGCSE '17*, pp. 417–422, 2017.
- S. Potineni, S. K. Bansal, and A. Amresh, "ScrumTutor: A web-based interactive tutorial for Scrum Software development," *Proc. 2013 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2013*, pp. 1884–1890, 2013.