Challenges in Teaching Modeling in Agile Software Engineering Courses

Matthew Stephan

Department of Computer Science and Software Engineering
Miami University
Oxford, OH, USA
Email: stephamd@miamioh.edu

Abstract. Formal Model Driven Engineering(MDE) can be considered incongruent with Agile methodologies. However, with the advent of Agile, Software Engineering educators have an obligation to teach Agile development. Many instructors do so by employing experiential learning through Agile classrooms and projects. Teaching formal MDE and convincing students of its benefits can be challenging in such environments. In this paper, we discuss this position by considering established best practices in modeling education and their compatibility in Agile classrooms/projects. We argue that more than half the practices present some challenge in Agile environments, for which we provide initial suggestions. Our goal is to stimulate discussion on these challenges and others, and help guide future research and course offerings.

Keywords: Model driven engineering ·Agile Software Development· Software Modeling·Agile Methodologies·Software Engineering Education

1 Introduction

Agile development methodologies are widespread in Software Engineering. 35% to 45% of software developers use some form of Agile, which is more than the 30.6% that use nothing, 21% that use iterative, and the 13% that use waterfall [16]. Other studies have Agile adaption for small-to-medium sized organizations at 67% [12]. Thus, educators have a responsibility to teach, and have a significant focus on, Agile methodologies. One popular way of doing this is to structure and teach classes in an Agile fashion [11], which can include having class projects conducted using Agile [12], yielding educational benefits not limited to engineering education [10].

One strength of Model Driven Engineering (MDE) is code generation [15] facilitated by carefully crafting models adhering to formal syntax. While there are examples of round-trip MDE [2], whereby models and code can be updated and synchronized automatically, it is relatively underdeveloped. Thus, formal MDE is sometimes viewed as incongruent with Agile methodologies [5], as Agile is lightweight, and focuses on getting simple "working software" and updating it later. Teaching MDE is crucial in Software Engineering education [8]. However,

doing so in an Agile environment/course structure while convincing students of its benefits is challenging.

This paper describes our position that following established practices for teaching MDE in an Agile setting presents unique considerations and challenges that should be addressed by the modeling education community. Our discussion considers these practices in the context of Agile, and includes suggestions when conflicts exist. Our goal is to stimulate ideas at the symposium on these challenges, generate others, and guide future research and practice.

2 Background Information

2.1 Agile Development Methodologies

Agile methodologies involve principles, practices, and methods for software development and other fields. It allows teams to be more flexible, and able to meet the demanding needs and changing requirements of consumers. While not applicable to all projects and products, it has seen notable adoption recently [12, 16]. It is not meant to replace established Software Engineering practice, but rather be employed as a guiding philosophy [3].

Agile Education. There are different approaches to teaching students Agile methodologies, including using a decision support process with use cases [10], Scrum labs [14], pair programming [17], and others [12]. Additionally, classrooms/lectures themselves can be conducted in a Agile fashion with students as the customers. This can include a preparation and planning phase with students, setting specific learning outcomes with them, and applying Agile practices each class session including stand ups and retrospectives [11]. Class projects can be structured using Agile processes [12].

2.2 Teaching Modeling and MDE

The importance and impact of teaching MDE has been discussed by Hamou et al. [8], with Clarke et al. demonstrating students' positive experiences [6].

In Hamou et al.'s work, one of the main considerations they identify when teaching MDE involve the "multitude and the complexity of the (MDE) concepts" [8]. Additionally, Kuzniarz and Staron explicated a list of best practices for teaching UML/modeling based on five years of experience [9]:

P1 Tailoring development processes to participant knowledge, course setting, and course format/restrictions
P2 Defining artifacts and creation procedures beforehand
P3 Consistency awareness and management, making sure all artifacts are consistent with one another
P4 Teaching importance and usage of models and their elements, including relation of elements to code generation
P5 Receiving and encouraging constant feedback from students
P6 Industrial relevance, that is, preparing students for the "real" world
P7 Performing experiments
P8 Including new research and industrial trends

We consider these practices and their plausibility in Agile courses/projects.

3 Related Work

3.1 Agile Modeling

Agile modeling is a light-weight modeling technique intended to derive some benefits from modeling in Agile settings [1]. It is neither as rigid nor as formal as "generative" MDE, and strives to be "just barely good enough" [1]. Support for Agile modeling is sparse [7], but it aims to be practical and has potential in Agile environments.

The modeling community and Software Engineering educators should teach students formal MDE to best prepare them to work on safety critical, secure, and reliable software [4]. Trying to introduce students to modeling and MDE using Agile modeling can impede MDE learning. For example, Ringert et al. taught MDE using Agile MDE for cyber-physical systems and found that students encountered trouble with MDE technology and concepts [13].

4 Challenges

In this section, we discuss how Agile courses clash or fit with formal/generative MDE education practices. We frame our conversation in terms of the modeling education best practices mentioned in Section 2.2, with each item, AX, referring to each PX in that list. For those that present challenges, we include suggestions. This list is only a starting point. We anticipate more challenges and considerations will emerge in the symposium's discussions and in future work.

A1 This practice dictates instructors tailor development processes to a course and its context. However, Agile purists/proponents argue that a project process is either Agile or it is not; it cannot be "somewhat" Agile. While tailoring MDE aspects is possible, tailoring certain process-specific aspects may not be appropriate in an Agile setting. Thus, Agile courses/projects should tailor only MDE process aspects, not Agile process aspects.

A2 Defining MDE artifacts types for students before an Agile course begins is challenging. To teach MDE properly, the choice of artifacts should be sufficient to facilitate and elucidate the formal MDE pipeline, but must be a feasible set students can create/update in each Agile iteration. More research and experimentation is required to address this.

A3 Artifact consistency is easily possible in an Agile course or project. To enforce this, recurring Agile stories/tasks to manage and synchronize artifacts can be created automatically each iteration through tooling, like Jira[1] or Trello[2].

A4 Since each iteration in Agile requires a quality working product, teaching the effective use of MDE models fits nicely. However, considerations include, prior to the first iteration, 1) students should be familiar with MDE basics, and 2) code generation is setup and ready for them. This will likely require a primer on MDE and code generation before any work begins.

[1] `https://www.atlassian.com/software/jira`
[2] `https://trello.com/`

A5 Adaption of the constant feedback best MDE education practice within Agile is seamless, as this is a key Agile principle. Students will be in constant communication with the instructor/customer through both stand-ups and progress story boards.
A6 Industrial relevance in teaching MDE applied in a Agile setting is challenging. MDE in formal domains, like automotive, telephony, and others is often front loaded. This is appropriate as requirements change less, and can be regulation/domain-driven. While students can learn the generative aspect, it may not be as front loaded because Agile involves building small products and incrementally adding features. One way of tackling this may be having a large minimum viable product(MVP) in the first iteration.
A7 MDE Experimentation is possible in Agile courses/projects. For example, in class-wide projects, groups can swap models to analyze a model set's ability to aid in comprehension each iteration. Or, in between iterations, groups can be asked to migrate their models to another tool to compare, teach, and experiment with multiple modeling tools.
A8 There should be no problem in having students learn and employ emerging MDE research and tools in an Agile setting.

Table 1. Best Practices and Compatibility with Agile

Key	**Practice Description**	**Challenge Level**
A1	Tailor Development Processes	*
A2	Define MDE Artifacts Beforehand	*
A3	Keep Artifacts Consistent	-
A4	Effective use of MDE Models/Code Generation	-
A5	Constant Feedback	✓
A6	Industrial Relevance	*
A7	Experimentation	✓
A8	Employ New Research & Technologies	✓

* Challenging
- Mildly challenging
✓ Little to no challenge

We summarize our qualitative assessment of these challenges in Table 1. These will form the basis for discussions at the symposium and beyond in the hopes of generating more challenges and suggestions. Additionally, we plan on applying these MDE practices as best possible in our Agile course offerings in the near future for the purposes of observation and research.

5 Conclusion

Agile methodologies can be considered incongruent with formal MDE. Similarly, employing Agile education techniques can conflict with formal MDE education. As a starting point for discussion, we considered eight established best practices for teaching modeling with respect to their applicability in Agile classrooms/projects. We postulate that more than half of these MDE practices face

at least mild challenges in Agile settings. Some of our preliminary suggestions include automatic Agile task creation for artifact consistency, a larger first iteration/MVP for industrial relevance, and others. We plan on discussing these challenges and more at the symposium, and applying these practices and suggestions in our Agile class offerings in the near future.

References

1. Ambler, S.W.: The object primer: Agile model-driven development with UML 2.0. Cambridge University Press (2004)
2. Antkiewicz, M., Czarnecki, K., Stephan, M.: Engineering of framework-specific modeling languages. Transactions of Software Engineering 35(6), 795–824 (2009)
3. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al.: Manifesto for agile software development (2001)
4. Bezivin, J., France, R., Gogolla, M., Haugen, O., Taentzer, G., Varro, D.: Teaching modeling: why, when, what? In: MODELS. pp. 55–62. Springer (2009)
5. Cabot, J.: Agile and Modeling / MDE : friends or foes? `http://modeling-languages.com/agile-and-modeling-mde-friends-or-foes` (2010), [Online; accessed 15-June-2017]
6. Clarke, P.J., Wu, Y., Allen, A.A., King, T.M.: Experiences of teaching model-driven engineering in a software design course. In: Educators Symposium of the MODELS Conference. pp. 6–14 (2009)
7. Erickson, J., Lyytinen, K., Siau, K.: Agile modeling, agile software development, and extreme programming: the state of research. Journal of Database Management 16(4), 88 (2005)
8. Hamou-Lhadj, A., Gherbi, A., Nandigam, J.: The impact of the model-driven approach to software engineering on software engineering education. In: International Conference on Information Technology: New Generations. pp. 719–724 (2009)
9. Kuzniarz, L., Staron, M.: Best practices for teaching UML based software development. In: MODELS. pp. 320–332 (2005)
10. McAvoy, J., Sammon, D.: Agile methodology adoption decisions: An innovative approach to teaching and learning. Journal of Information Systems Education 16(4), 409 (2005)
11. Reed, P.: An agile classroom experience. In: Agile Conference. pp. 478–483 (2008)
12. Rico, D.F., Sayani, H.H.: Use of agile methods in software engineering education. In: Agile Conference. pp. 174–179 (2009)
13. Ringert, J.O., Rumpe, B., Schulze, C., Wortmann, A.: Teaching agile model-driven engineering for cyber-physical systems. In: ICSE: Software Engineering and Education Track. pp. 127–136 (2017)
14. Schroeder, A., Klarl, A., Mayer, P., Kroiß, C.: Teaching agile software development through lab courses. In: Global Engineering Education Conference. pp. 1–10 (2012)
15. Selic, B.: The pragmatics of model-driven development. IEEE Software 20(5), 19–25 (2003)
16. West, D., Grant, T., Gerush, M., Dsilva, D.: Agile development: Mainstream adoption has changed agility. Forrester Research 2(1), 41 (2010)
17. Williams, L., McCrickard, D.S., Layman, L., Hussein, K.: Eleven guidelines for implementing pair programming in the classroom. In: Agile Conference. pp. 445–452 (2008)