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. The most significant challenges are those pertaining to the MDE education practices of tailoring the MDE processes to participant knowledge and course context, defining the types of MDE artifacts before the project or course begins, and ensuring industrial relevance to prepare students for post-graduation. Discussion of these challenges at the symposium provided additional insights and suggestions, which we summarize in this paper. Notable suggestions include tailoring both the MDE and Agile processes to complement each other, employing a minimal set of MDE artifacts necessary for code generation, and having a large initial Agile iteration to form a realistic minimal viable product, respectively. We plan on applying these MDE educational best practices in our Agile course offerings, incorporating our initial suggestions and those from the symposium. Our goal is to stimulate discussion on these challenges and others, and help guide future research and course offerings.**

I. 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 [1]. Other studies have Agile adaption for small-to-medium sized organizations at 67% [2]. 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 [3], which can include having class projects conducted using Agile [2]. This provides educational benefits not limited to engineering education [4].

One strength of Model Driven Engineering (MDE) is code generation [5] facilitated by carefully crafting models adhering to formal syntax. While there are examples of round-trip MDE [6], 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 [7], 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 MDE 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. We include our own suggestions and those that arose at the symposium. Our goal is to stimulate ideas on these challenges, generate others, and guide future research and practice.

The paper begins in Section II with a quick primer on Agile Methodologies and Education, and MDE education. In Section III we argue Agile Modeling is inadequate for educating students in both Agile and Modeling. Our discussion begins in Section IV, which frames the conversation in terms of eight established formal MDE education practices. We conclude the paper in Section V.

II. Background Information

A. Agile Development Methodologies

Agile methodologies involve principles, practices, and methods for software development and other fields. It allows teams to be more flexible, and capable of meeting the demanding needs and changing requirements of consumers. While not applicable to all projects and products, it has seen notable adoption recently [1], [2]. It is not meant to replace established Software Engineering practice, but rather be employed as a guiding philosophy [9].

B. Agile Education

There are different approaches to teaching students Agile methodologies including using a decision support process with use cases [4], Scrum labs [10], pair programming [11], and others [2]. Additionally, classrooms/lectures themselves can be conducted in an Agile fashion with students as the customers.

For example, having 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 [3]. Class projects can be structured using Agile processes [2].

C. *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 [12].

One of the main considerations Hamou et al. 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 [13]:

P1 Tailoring development processes to participant knowledge, course setting, and course format/restrictions. For example, having instructors decide if a project/course will be iterative and incremental, the modeling languages and technologies employed by students, and other characteristics.

P2 Defining artifacts and creation procedures beforehand. That is, will students be using use case diagrams, state diagrams, sequence diagrams, et cetera.

P3 Consistency awareness and management to ensure all artifacts are consistent with one another. This can include establishing consistency rules for the students to use throughout the semester.

P4 Teaching the importance and usage of models and their elements, including relation of elements to code generation. A key principle to impart is that "if the model is not used anywhere, it should not be created."

P5 Receiving and encouraging constant feedback from students. The course must be at a level students can appreciate and follow. Eliciting feedback from students is helpful in realizing this.

P6 Industrial relevance, that is, preparing students for the "real" world. The problems covered in the course and in projects should be real-world problems.

P7 Performing research and academic experiments. Kuzniarz and Staron provide an example of their experiments that assessed the usability of UML stereotypes to help increase UML model comprehension.

P8 Including new research and industrial trends in the course material and project. The purpose is to raise the students' interest in emerging technologies and developments in MDE. Also, to have them better prepared when they graduate and pursue either industrial or academic pursuits.

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

III. Related Work

A. *Agile Modeling*

Agile modeling is a light-weight modeling technique intended to derive some benefits from modeling in Agile settings [14]. It is neither as rigid nor as formal as "generative" MDE, and strives to be "just barely good enough" [14]. Support for Agile modeling is sparse [15], 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 [16]. 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 [17].

IV. 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 II-C, with each item, A*X*, corresponding to each P*X* in that list. For those that present challenges, we include suggestions based on our initial thoughts and our discussions at the symposium. This is only a starting point. We anticipate more challenges and considerations will emerge in future work.

A. *A1 - Tailor Development Processes*

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 and projects should tailor only MDE process aspects not Agile process aspects.

At the symposium, a participant argued MDE purists would say MDE processes should not be tailored, and there are several Agile methods that are highly customizable for different contexts. A specific example of this is the Scaled Agile Framework [18] in the systems engineering domain that deals with standardization and safety-critical requirements. It can be argued that these Agile methods/frameworks are made to be customized, much more than methods that focus on MDE.

B. *A2 - Define MDE Artifacts Beforehand*

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. However, the choice of artifacts must also be a feasible set students can create/update in each

Agile iteration. More research and experimentation is required to address this.

The symposium attendees agreed that it is challenging to find the right balance of artifacts that are sufficient and feasible. One person posed the question if MDE is ever complete. Another person responded it should be capable of code generation at least.

C. A3 - Keep Artifacts Consistent

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].

An industrial symposium attendee confirmed that they see this in industry. That is, they have explicate stories and tasks intended to ensure consistency. They verify correctness at the modeling level of abstraction.

D. A4 - Effective use of MDE Models/Code Generation

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.

At the symposium, participants were in strong agreement that educators must teach, or review, "good design" and modeling to students beforehand.

E. A5 - Constant Feedback

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.

The symposium attendees believe that constant communication both early on during the project and course will not impact MDE education, and having Agile-type retrospectives after milestones will be beneficial. No conflicts were anticipated.

F. A6 - Industrial Relevance

Industrial relevance in teaching MDE applied in an 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 aspects in an Agile setting, 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.

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

One of the symposium attendees from industry pointed out they use their models mostly for diligence and auditing. So, they were less concerned with students being familiar with code generation. They develop their models in an Agile fashion but not the product itself. Of course, this is organization specific. Hutchinson et al. [19] performed a survey on MDE use in industry and found approximately 12% of organizations use code generation, 34% use models for testing, and 39% use models for execution and simulation. Our suggestion is that each course instructor should use this survey, surveys on Agile [1], [2], their industrial collaborations, and own research to decide how to make both their Agile and MDE curricula as industrial relevant as possible.

G. A7 - Experimentation

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.

H. A8 - Employ New Research & Technologies

There should be no problem in having students learn and employ emerging MDE research and tools in an Agile setting.

I. Summary

We summarize our qualitative assessment of these challenges in Table I. These helped form the basis for discussions at the symposium and, hopefully, beyond in 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.

V. 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 and course 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 release for industrial relevance, and others. We discussed these challenges at the symposium, and are conducting research by applying these practices and suggestions in our Agile class offerings in the near future. It is our hope that this paper helps guide future research and furthers the conversation on the educational interplay between Agile and MDE.

TABLE I
SUMMARY OF BEST PRACTICES' 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	Research and Academic Experimentation	✓
A8	Employ New Research & Technologies	✓

* Challenging
- Mildly challenging
✓ Little to no challenge

ACKNOWLEDGMENTS

We would like to thank the Educators Symposium attendees for participating in our discussions, providing feedback, and brainstorming suggestions. We also want to express our gratitude to the symposium organizers for helping us facilitate the discussion.

REFERENCES

[1] D. West, T. Grant, M. Gerush, and D. Dsilva, "Agile development: Mainstream adoption has changed agility," *Forrester Research*, vol. 2, no. 1, p. 41, 2010.

[2] D. F. Rico and H. H. Sayani, "Use of agile methods in software engineering education," in *Agile Conference*, 2009, pp. 174–179.

[3] P. Reed, "An agile classroom experience," in *Agile Conference*, 2008, pp. 478–483.

[4] J. McAvoy and D. Sammon, "Agile methodology adoption decisions: An innovative approach to teaching and learning," *Journal of Information Systems Education*, vol. 16, no. 4, p. 409, 2005.

[5] B. Selic, "The pragmatics of model-driven development," *IEEE Software*, vol. 20, no. 5, pp. 19–25, 2003.

[6] M. Antkiewicz, K. Czarnecki, and M. Stephan, "Engineering of framework-specific modeling languages," *Transactions of Software Engineering*, vol. 35, no. 6, pp. 795–824, 2009.

[7] J. Cabot, "Agile and Modeling / MDE : friends or foes?" http://modeling-languages.com/agile-and-modeling-mde-friends-or-foes, 2010, [Online; accessed 15-June-2017].

[8] A. Hamou-Lhadj, A. Gherbi, and J. Nandigam, "The impact of the model-driven approach to software engineering on software engineering education," in *International Conference on Information Technology: New Generations*, 2009, pp. 719–724.

[9] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries *et al.*, "Manifesto for agile software development," 2001.

[10] A. Schroeder, A. Klarl, P. Mayer, and C. Kroiß, "Teaching agile software development through lab courses," in *Global Engineering Education Conference*, 2012, pp. 1–10.

[11] L. Williams, D. S. McCrickard, L. Layman, and K. Hussein, "Eleven guidelines for implementing pair programming in the classroom," in *Agile Conference*, 2008, pp. 445–452.

[12] P. J. Clarke, Y. Wu, A. A. Allen, and T. M. King, "Experiences of teaching model-driven engineering in a software design course," in *Educators Symposium of the International Conference on Model Driven Engineering Languages and Systems Conference*, 2009, pp. 6–14.

[13] L. Kuzniarz and M. Staron, "Best practices for teaching UML based software development," in *International Conference on Model Driven Engineering Languages and Systems*, 2005, pp. 320–332.

[14] S. W. Ambler, *The object primer: Agile model-driven development with UML 2.0.* Cambridge University Press, 2004.

[15] J. Erickson, K. Lyytinen, and K. Siau, "Agile modeling, agile software development, and extreme programming: the state of research," *Journal of Database Management*, vol. 16, no. 4, p. 88, 2005.

[16] J. Bezivin, R. France, M. Gogolla, O. Haugen, G. Taentzer, and D. Varro, "Teaching modeling: why, when, what?" in *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2009, pp. 55–62.

[17] J. O. Ringert, B. Rumpe, C. Schulze, and A. Wortmann, "Teaching agile model-driven engineering for cyber-physical systems," in *International Conference on Software Engineering: Software Engineering and Education Track*, 2017, pp. 127–136.

[18] D. J. Reifer, F. Maurer, and H. Erdogmus, "Scaling agile methods," *IEEE software*, vol. 20, no. 4, pp. 12–14, 2003.

[19] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen, "Empirical assessment of mde in industry," in *International Conference on Software Engineering*, 2011, pp. 471–480.