Education to Agile: Fostering Team Awareness with Essence

Paolo Ciancarini(✉) and Marcello Missiroli

DISI, University of Bologna, Bologna, Italy
paolo.ciancarini@unibo.it

Abstract. This paper explores the incorporation of Agile practices in our undergraduate courses leveraging the Essence approach, a meta-notation for describing software processes, roles, and best practices.

Exposing students and young developers to the Agile mindset and related methods is important to let them to cope with the challenges of modern software development and digital transformations. Agile methods and practices can also help students to develop valuable soft skills such as communication, teamwork, and adaptability.

Essence clarifies and guides several key Agile practices, thanks to guidelines and checklists, such as: team building, customization of Agile ceremonies, promoting the effectiveness of retrospectives, tool selection and configuration. We found that the usage of Essence helps students to develop critical thinking and a sense of ownership and responsibility related to their teamwork. They achieve a better understanding of what is expected of them, and, as a result, they are more motivated to achieve their goals.

1 Introduction

During the last few years, we have been researching on the practice of Agile methods for teaching Software Engineering. We introduced the principles of Agile development in our Software Engineering courses for undergraduates, an ongoing process that has transformed the way we teach, support students, and assess their results in, hopefully, a way more compliant with the Agile vision.

Our first involvement with Agile was the *AMINSEP* research project [4], which confronted us with the problems of digital transition in a strongly structured environment in an Italian Public Administration. In AMINSEP we developed and enacted *iAgile* (which stands for *improved Agile*), a process model that encompassed the mainstream Scrum framework to adapt it to the necessity of a public administration engaged in projects of an Agile digital transformation in a critical context requiring high security [30].

After the termination of the project we began experimenting by introducing the Agile vision to inexperienced programmers, both in high school and in our university courses. Our preliminary results showed that Agile had indeed a potential in this field [22].

A. Capozucca et al. (Eds.): FISEE 2023, LNCS 14387, pp. 69–84, 2023.
https://doi.org/10.1007/978-3-031-48639-5_6

We have introduced Cooperative Thinking [6] as a combination of Computational Thinking and Agile collaboration. We studied which collaborative tools are especially useful for Agile developments [7], and evaluating outcomes of Agile development projects in high schools or university courses [8,22].

We came to the conclusion that students and young developers need to be exposed to both Agile methods and powerful open source tools to be able to cope with the challenges of modern software development.

Following the results of a survey on tools used by the Agile community worldwide [7], we started building and experimenting with a fully open source development environment that is compatible with the principles of Agile and privacy, the *Composable Agile System* (CAS) [5]. CAS was especially useful during the lockdown periods caused by the pandemic, because students got powerful collaborative tools at their disposal, which allowed them to work on their project and at the same time allowed us to collect process data to be analyzed [8].

The most difficult challenge was to convince the students to adopt a discipline of agile collaboration. Convincing students that software is a social construct, in fact, was not easy. We focused on some specific practices: team building, product ownership, negotiating requirements, and collective analysis during retrospectives [21]. We found that students need some specific guidance in understanding a process model like Scrum, and to acquire self-awareness as a team. Essence - a meta-tool used to describe processes [14] - proved to be a useful addition. Essence helps the students in two ways: firstly, it helps to describe the process they follow, allowing some degree of customization according to their necessities. Essence helps in explaining that some practices are not mandatory, and can be substituted by alternative practices. For instance, after beginning to use Scrum, they rapidly realize that the daily meeting is not suitable – as it is not compatible with their course schedule. A second crucial help consists in using Essence during the retrospectives to understand and track the progress of the process as a whole, check its weaknesses, and possibly introduce adjustments for the next sprint.

Agile software development is an iterative and flexible approach to software development that emphasizes collaboration, customer satisfaction, and rapid delivery of working software. Open source tools are software tools whose source code is made freely available to the public, allowing anyone to modify and improve upon them. The relationship between Agile software development and open source tools is, in our view, a natural one. Open source tools are often used in Agile development because they offer developers the flexibility and the adaptability they need to quickly respond to changing requirements.

The open source tools we suggested to use are flexible and customizable, allowing developers to tailor them to meet their specific needs. This flexibility makes them ideal for use in Agile development, where requirements can change frequently and rapidly – the relationship between Agile software development and open source tools is one of mutual benefit.

In summary, we found that our courses greatly improved by introducing the combined use of Agile methods and Open source tools, and Essence provided a solid framework for teaching, monitoring, customizing and reflecting on the process within a Cooperative Thinking perspective.

The rest of this paper has the following structure:

- Section 2 presents the status of research on these topics;
- Section 3 describes the process of refocusing our courses on the Agile perspective and introducing a Project-based assessment;
- Section 4 highlights the role covered by Essence in the process;
- Section 5 shows the outcomes of the changes introduced;
- finally, in Sect. 6, we draw our conclusions and describe the forthcoming further changes that we are planning for the next edition of the course.

2 Literature Review

Teaching Agile software development is now a popular approach, and several papers report about experiences of Agile projects developed by Computer Science undergraduates. For instance, [9] discusses how Agile software development is taught to students of computer science and software engineering in various universities; the authors describe their experiences and provide some recommendations for instructors, like keeping up the pace of the developing teams using some form of minimal competition.

In the paper [17] we found the description of a course on Agile methodologies taught to software engineering students analyzing their personalities and attitudes to teamwork, thus adopting some form of team building before the start of the actual projects. In [20] there is a discussion of how the use of collaborative practices needs some maturity by the students. The idea of using real-life problems, with an approach called Problem-Based Learning, to study Agile development is discussed in [12].

A recent paper considers how product management and Agile can be taught to undergraduates is [24]. Product management is relatively unexplored in software engineering, yet it is an important topic in any effort of digital transformation.

The Essence way to teach Agile practices and train Agile students and developers is presented in [15]. A paper from a different group concerning how to use Essence cards and approach is [34]. Two papers from Peraire and Sedano validate the Essence approach used by students in retrospectives [25,26]. A paper discussing how Essence can be used during an academic course project work is [16]; this paper discusses the difficulties of adopting Essence with undergraduate students.

Essence can be considered as a playful framework for team building and support [32]; the role of serious games in the education of Agile developers and related team building activities is reviewed in [28].

We have devoted some effort to search for recent evidence on teaching Agile development using open source tools only. The match seems natural, however we have found only few papers: the report [33], which uses tools quite different from those we used, in particular they used Redmine for project management and Bugzilla as issue tracker; and the paper [36], which focuses on software reuse in a community of student developers.

Documentation of process choices, tools selections, and their rationale is an important factor in software development projects in order to support product quality and future maintenance. While several research publications address this topic, systematic approaches and tools are rarely found in practice, and are not well covered in software engineering education. Lack of suitable process documentation is especially an issue in Agile software development, where processes and tools are often seen as less important than working products [31]. We found some reports describing the use of a Scrum-like approach adaptable by the students, see for instance [1,29].

Concerning the evaluation of product, process and teams, our work has been inspired by the quality model described in the article by Hoegl and Gemuenden [13], and further developed by Lindsjørn et al. [18]. The maturity model for Agile teamwork, on the other hand, was proposed by Yin et al. [37], based on the one created by Chetankumar et al. [3]. Gren et al. have also discussed the concept of teamwork maturity in Agile teams [11].

3 Extreme Development

We have been teaching a "Software Engineering" course (part of the CS degree curriculum, 3rd year) for several years. Its syllabus had a traditional structure, i.e. mostly lecture-based with some workshops, and focused on traditional, waterfall-style software engineering: software requirement specification (SRS) - including several UML diagrams - test specs and some design patterns. The final exam was based on writing a project plan, drawing some UML diagrams, and answering a few questions.

When the pandemic began we reorganized the course structure, with the following goals in mind:

1. Promote the idea that developing software is a social activity, for instance with team building activities based on games;
2. Emphasize Agile development principles, including the possibility of choosing the most suitable practices;
3. Foster a product-oriented mindset educating the students to the role of product owner;
4. Provide open source tools to experiment and support Agile collaborative practices;
5. Offer continuous monitoring and feedback to students during their projects (not just at the end).

We introduced these changes related to these goals over a three-year period, and some elements are still subject to adjustments.

Implementing the Agile vision in our course presented challenges, as it encompasses various concepts, including values and principles as in the Agile Manifesto, some best practices, a lightweight project management framework, emphasis on versioning and deployment disciplines (like pipelines based on GitLab and systematic usage of docker). We adopted the Scrum framework, combined with some XP practices, like pair programming.

To put Agile into practice, we incorporated a project into the course format. Students were required to form teams since the first days of the class. Thus, the project activities began early in the 12-week course, with sprints spanning three weeks. We did not fix the number of sprints, but we asked if possible to conclude the development by the end of the fall term. The required non trivial product was a Twitter client capable of aggregating posts using visual analytics techniques. Most teams successfully delivered their products by the end of the lectures, with only a few teams requiring additional sprints. Each team provided a report documenting their process, including productivity data, a product demo, and a final retrospective.

We faced two main challenges:

1. Students lacked experience with teamwork since the current Computer Science curriculum typically focuses on *individual* programming. To address this, we introduced team-building activities to help them to develop and test some teamwork skills.
2. Initially each team had the freedom to choose its development tools, which sometimes resulted in complex and demanding solutions. This led to variations in team productivity and introduced unnecessary complexity. Furthermore, the COVID-19 pandemic forced all students to work online, depriving them of face-to-face communication, a key aspect of Agile. Therefore, we provided a standard set of tools for all teams, in order to provide a working environment with minimal configuration. This prompted us to develop an "Extreme development" praxis, as detailed below.

3.1 Our Motivation

During the pandemic the necessity of working remotely forced us to choose a number of collaborative tools to support students working online. We wanted students to be in complete control of their development environment, including process data and artifacts. We restricted the choice to open source software suitable to offer complete control of all artifacts and data they produced. This was challenging, since several students were used to online tools that were closed-source, with unclear privacy terms of use, or both.

To this end, we identified a series of FLOSS products that, when combined, provided the basic services similar to their commercial counterparts, such as *Trello* or *Slack*. These open source software include *Git+Gitlab*, *Jenkins*, *SonarQube*. We added *Mattermost* for communication (an open source alternative to *Slack*), and *Taiga* for project management (a more comprehensive and open source alternative to *Trello*). These tools formed the core of our CAS [5] system. We configured some virtual machines as instances of the development environment, and gave students access. Some teams opted for a self-hosted solution however. The students could use any open source IDE, like *Eclipse*, *IntelliJ*, *Atom* or *Visual Studio Code* (the latter one was the preferred choice).

We also required each student to track the time spent on the project. To avoid commercial products, we provided a self-developed IDE plugin able to log the

actions of the developers client-side; these actions could then later be analyzed either individually or collectively, team-wide, using a dashboard.

We named this praxis as "*extreme Agile development*". It is an *extreme* form of development because it required the combination of:

- an Agile framework, as given by Scrum, tailorable by the students;
- the necessity of using open source only tools;
- the requirement to track the productivity data both as individual developers and as a team;
- the freedom to choose and adapt the Agile best practices most suitable to the team, as for instance pair programming, the daily scrum, and the retrospectives;
- the requirement to use systematically Essence cards for the retrospectives and for any additional tool or best practice adopted by the team;
- pervasive teamwork, including team building activities and using collaborative tools.

While these issues are not necessarily correlated, we assume that modern would-be developers should be exposed to all of the above to be ready for the real-world challenges. Our goal here is to pull students out of their comfort zones as individual developers, and reinforce their collective behavior and reciprocal trust.

3.2 Fostering Extreme Development

We hold that writing software is a social activity performed in teams. However, education to programming tends to focus on *individual*-centered learning and sometimes discourages cooperation, even penalizing it as cheating. Agile's best practice of collective code ownership often contradicts this approach [2]. To address this issue, we needed to promote self-organization, positive team building, and personal accountability.

- *Self-organization* was achieved by letting students form teams with minimal limitations. Specifically, we wanted all teams to have more or less the same number of participants, usually five or six. The process of team forming usually required 3–5 days, and our intervention and advice was very limited.
- *Team building* was integrated in the preliminary iteration of the project (defined as "Sprint 0"), during which the students performed some training games, such as "Scrumble"[1], and "Escape the Boom"[2]. These games can be played remotely online - this was a requirement during the pandemic. The performance of each team when playing these games was self-evaluated by the team itself using a GQM approach.

[1] http://scrumble.pyxis-tech.com.
[2] https://escape-the-boom.com.

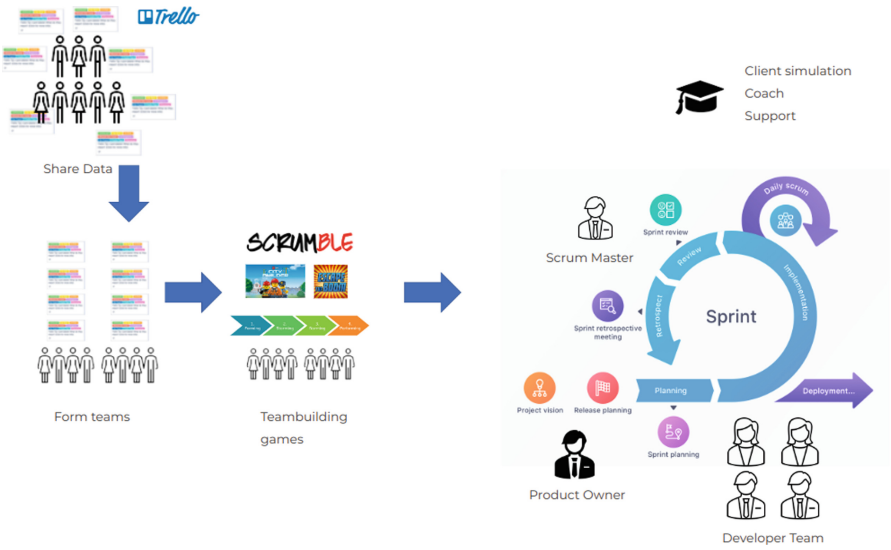


Fig. 1. The structure of the project process

– To promote *personal accountability* we committed to providing frequent and precise feedback to teams, and let them handle the results. This was achieved by requiring reports and surveys at the end of each sprint, as well as offering online discussion sessions with faculty on a regular basis.

Git was instrumental in providing several important data, such as the number and the size of commits and their temporal distribution. More to the point, we utilized *gitinspector*[3] to present *git* log data in a simple and graphical format to the team. For example, the tool is able to detect common usage patterns, such as who is the person most responsible for a given file, or the person that contributed the most lines of codes, in a weekly breakdown. This data is then used as a basis for a teacher-student discussion. Figure 1 shows how the project process has been organized.

Students form teams with *Trello*, as it is a well known and offers a good app over the smartphones. Each student builds a self-presentation in the form of a card explaining personal abilities and preferred activities, eg. Python programmer, User interface designer, etc. This is a format derived from Agile *unconferences* worldwide, emphasising the self-organizational aspect. Then, they look for companions to form teams. After that, the first activity we suggest is to engage in some team building games in order to practice Scrum roles and assign them. By far, the most successful game we introduced is Scrumble, that we adopted in an online variant we have implemented. Most students reported its usefulness in

[3] https://github.com/ejwa/gitinspector.

getting the idea of what Scrum works and what is expected from Scrum roles. Most teams reported that they assigned the roles of product owner and Scrum master after playing the game.

4 The Role of Essence

The introduction of an Agile project in our course required, for most students, a complete change of mindset: no "think ahead", requirements negotiable and not imposed or predefined, a lot of teamwork, focus on product and customer, using powerful development tools, and more. *Lecturing* students on these principles is one thing, have them *understand* and *apply* them is completely different. Students receive an overwhelming amount of information in a short time period and, consequently, their overall understanding is shallow.

We needed some framework that could help and guide their learning even outside class hours. We choose Essence, a standard for software engineering methods [23]. Essence is based on a collection of guidelines and checklists in the form of cards[4], that describe the process roles and activities to be performed. The cards, among other things, are used in retrospective games and provide a solid mental scaffolding that helps students understand "where they are", possibly "where they are heading" and share this information among themselves and with us.

At first we used Essence cards only as a teaching support during lectures. We added the book [14] as a suggested reading, but we skipped a formal introduction and simply started using the cards. This is one of the proposed learning approaches proposed by its authors, defined as "stealth mode"[5].

We started with the so-called **Kernel Alphas**, which help to define the basic software engineering concepts such as "Requirements" and "Software System". To that, we added some concepts related to Agile, specifically *Product Backlog* and *Product Backlog Item* (which includes *User Stories* and *Tasks*). Depending on the context, the cards were used as visual teaching aid, cheat sheets, or recap elements. Table 1 shows a timeline of the introduction of the various Essence cards over the course.

4.1 Monitoring the Status of a Project

The Essence cards can be used in an interesting and active way during the development of a project, namely to understand the status of the process [27]. Even in real-life development projects it is difficult to have a clear idea of the status of a software project; in case of inexperienced developers tackling a complex assignment for the very first time, this becomes a very hard problem indeed.

[4] available at https://practicelibrary.ivarjacobson.com.

[5] https://paulemcmahon.wordpress.com/2019/06/09/scaling-essence-in-stealth-mode-and-why-you-should-care.

Table 1. The usage of Essence elements as the project progresses

Timeline	Focus	Essence elements	Tools
Lectures	SE Basics, open source value	Customer; Requirements	Planning poker app
sprint 0	Teambuilding, System setup	Team, System, Git, Gitlab	Gitlab
sprint 1	1st MVP, Retrospective	User Stories, Retrospective games	Taiga, logger, IDE, gitinspector
sprint 2	2nd MVP, Agile values and techniques	Pair Programming, Test Coverage	JUnit, Jest
sprint 3+	3rd MVP, Code quality, Refactoring	Evolve a Releasable Product, Manage technical debt	SonarQube
Release	Deployment pipeline	Devops Essentials	Jenkins

We suggested that teams should pick some Essence cards to track and assess their process status. We proposed some cards, such as "Stakeholder", "Software System", "Team" and, with the help of the checklists linked to the cards, students were able to provide a reasonable estimate of the "well being" of the project. As the project progressed, they used cards to assess their overall project progress, sometimes indicating the desirable state to be achieved at the end of each iteration. Though this practice was not mandatory, most groups used it during their retrospectives.

4.2 Retrospectives with Essence

Reflection is one of the Agile principles [35] (#12), but it is also is a key element in effective learning [10]; unfortunately, our students traditionally do not have much experience (if any) in this technique. Essence provided the perfect mechanism to do so. *Practice Patience*[6] is a gaming activity that drives the retrospective session – required at the end of each iteration:

1. The team selects a number of cards that are considered of interest. During the first retrospective we required the use of at least four cards, including the Team card.
2. Cards are positioned in a two-dimensional board, where the vertical axis represents the importance of the practice associated to the card, and the horizontal axis shows how well the team performed in relation to that topic, possibly adding comments.
3. The team discusses each card, and produces a list of improvement actions.

From a teaching perspective, it was easy to notice a general improvement of the teams over time; teams used more cards and were able to write useful comments and improvement suggestions. As many retrospectives were executed online, they developed an *Excalidraw* template which they shared among them.

[6] https://www.ivarjacobson.com/publications/blog/agile-tools-coaching/better-scrum-through-essence-part-2-1.

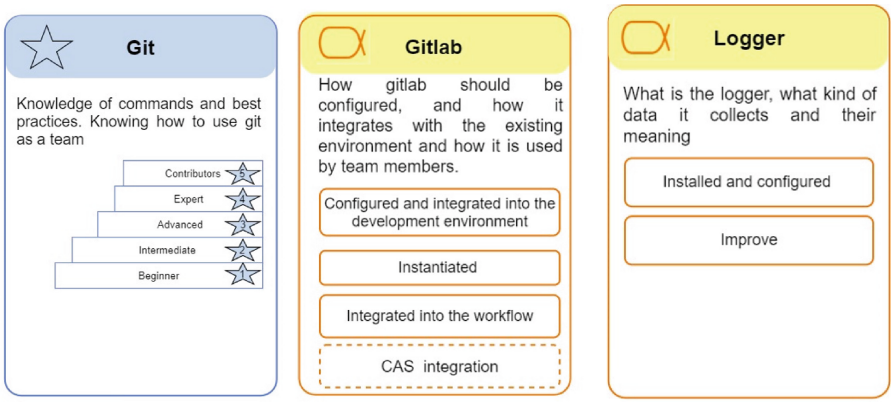


Fig. 2. Three new cards supporting Git, GitLab tool, and Logger. Following the Essence convention, the Git card is a skill, while GitLab and Logger are alphas.

4.3 Process Organization

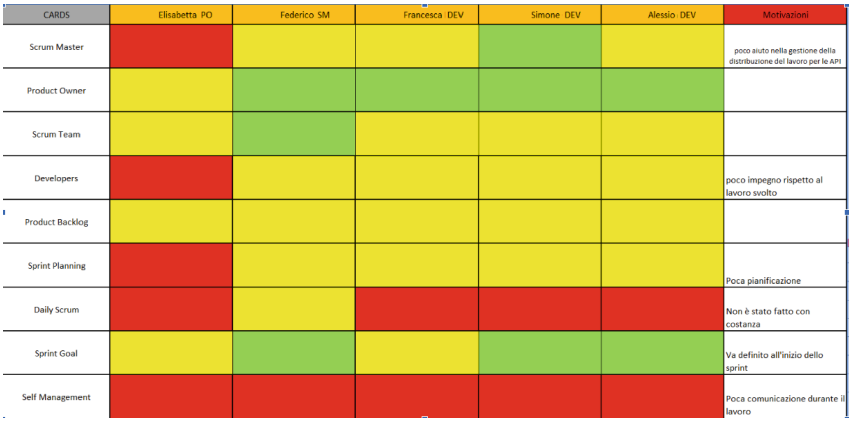
Some Agile principles clearly suggest that the software development process should be sustainable and teams be self-organized [35] (#8, #11). While we presented a standard development method based on Scrum with several XP elements, teams were free to organize themselves as they thought it was best for them. Essence offers a vast library of practices that students were free to implement or not.

For instance, the Daily Scrum practice was interpreted differently by groups: some skipped it altogether, some performed it online early in the morning, other opted for a bi-weekly face-to-face version. As a further example, pair programming was an Agile practice that was intensively used by some teams and completely ignored by others.

In fact, we can say that every team used a different, customized development process – further promoting the Cooperative Thinking mindset.

Though very comprehensive, Essence did not provide all the elements we wanted our students to learn and use. Since Essence is an open standard, we created some cards, shown in Fig. 2. Some are of general use, such as the ability to properly use Git commands and sharing a GitLab repository, others describe some services we offer in CAS, such as the Logger. Some other cards have been developed for other tools. Table 1 shows some Essence elements introduced as the project progresses.

Some teams proposed and customized changes to the process. Several teams decided to implement deployment pipelines from the beginning of the project, even before they were lectured on DevOps pipelines. Some team enriched the retrospective practice by tracking individual card evaluation over time, an interesting piece of information that is otherwise lost as individual opinions are merged and averaged - see Fig. 3.



CARDS	Elisabetta PO	Federico SM	Francesca DEV	Simone DEV	Alessio DEV	Motivazioni
Scrum Master						poco aiuto nella gestione della distribuzione del lavoro per le API
Product Owner						
Scrum Team						
Developers						poco impegno rispetto al lavoro svolto
Product Backlog						
Sprint Planning						Poca pianificazione
Daily Scrum						Non è stato fatto con costanza
Sprint Goal						Va definito all'inizio dello sprint
Self Management						Poca comunicazione durante il lavoro

Fig. 3. A matrix showing the results of a retrospective discussion. The leftmost column lists the practices chosen by the team to be discussed. The columns 2–6 are one for each team member: red-the practice is going bad, yellow-the practice needs care, green-the practice is going well. The last column contains some notes. The second column is from the PO, the third is from the SM: they disagree on the state of the practices (Color figure online)

5 Outcomes

It is very difficult to evaluate the impact of changes we applied, especially given that several modifications have happened over a relatively short amount of time: we modified the syllabus, the teaching strategies, the outcomes expected from the student teams, and the evaluation method. Therefore, our discussion here tends to be limited to anecdotal value, but in this case there are some data suggesting clearly that our changes have been beneficial.

Team responsibility is achieved by basing a large part of the final grade (around 80%) on the project outcome, and specifically on *process* evaluation rather than product evaluation. Personal reflections confirm that this approach has been understood and, generally, approved.

The number of students passing the exam has improved: while the average number of enrolled students has remained more or less constant, averaging 90, the number of non-passing students after two months from the end of the lectures has rapidly decreased to zero, from an average of 15% related to the previous course structure.

Another positive indicator is the students' feedback concerning the course. The course remained popular among students. Many students commented that the course was "more engaging" and provided "practical experience" that could be exploited in the job market. Some remarked that more effort (hours of work) were needed, but this is hardly a negative point from a teacher's perspective. The results of the first two instances of the class project, mostly executed during the strict lockdown in winter 2020–21 and 2021–22, were positive because all teams

Table 2. Perceived usefulness of tools.

	Mean	Std. err.	Mode	Median
Taiga	4,05	0,13	4	4
GitLab	4,92	0,05	5	5
SonarQube	3,41	0,17	3	3
Mattermost	2,89	0,28	1	3
Jenkins	2,38	0,26	1	1
Scrumble	3,32	0,20	3	3
Escape the Boom	2,38	0,27	3	3
Essence	3,49	0,15	4	4

completed their product before starting the spring semester. This experience is discussed in [19], written with two students who developed the project.

Table 2 shows the students' feedback related to some the proposed tools used within the CAS Environment (Taiga, Gitlab, SonarQube, Mattermost, Jenkins), the teambuilding games (Scrumble and Escape the Boom) and Essence cards. The excellent evaluation of both Taiga and GitLab was no surprise, whereas the low score of Jenkins and Mattermost was justified by the easy-to-use GitLab internal pipeline for the former and the limited features and usability of the latter, which is improving by the day. SonarQube fared reasonably well. The Scrumble game was deemed generally useful–some teams even played it twice–, but the Escape the Boom not very so.

The logger tool caused the most concerns, mainly because the program did not support all IDEs and all platforms and the continuous IDE upgrades created increasing incompatibilities with our software. By direct observation, we must add that the data recorded concerning productivity were not very reliable, since students were not precise in tracking their own activities, be it automatic or manual. We are currently looking for a viable solution to overcome this problem.

In general, the students evaluated positively the retrospectives based on the Essence method, provided as part of the final team report. Most students described the Essence cards as "clarifying" during the lectures and also "very beneficial" in understanding how some methods and techniques were intended to be used, specifically during Sprint Retrospectives.

The final question to be answered is whether all this reorganization effort was worth it or not. To answer that, we collected and organized team data, commented their results, and compared team performance to the final grades obtained by each student [8]. We used two evaluation models, a quality model (measuring overall code quality), and a maturity model (measuring how the team applied Agile values in their work).

Analyzing the results of the two models we have observed a linear relation (as shown in Fig. 4), suggesting that teams that closely followed the proposed work methodology, on average, produced better quality code. In our view, this confirms that our efforts in fostering performing teams using Agile method were fruitful.

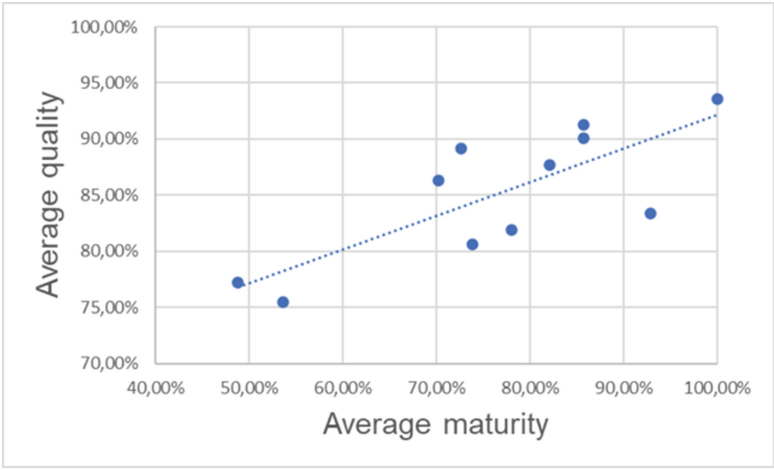


Fig. 4. Linear correlation of maturity and quality models [8]

6 Conclusions and Further Work

We and our students believe that Essence cards are quite useful to achieve a good Agile proficiency. In order to simplify and improve the introduction of agile practices, we plan to reinforce the use of Essence cards. Firstly, we are expanding their use during the lectures, since students like the synthesis offered by the cards and the freedom to select their own practices and tools. In addition, we are preparing additional cards, with the goal to address the details of new agile practices (for instance in product management), or to describe specific tools, like for instance the logger and the other tools that we are adding to the CAS environment for supporting diagramming and retrospectives.

Another challenging idea consists in managing a multi-team project, enacting some scalable process model like *SAFe* or *LESS*. Essence cards offer support for SAFe, however we need a reorganization of the courseware, and probably some new management tools, that we are currently studying.

Acknowledgment. We ack the support of CN-HPC under PNRR (National Recovery and Resilience Plan), and of CINI and CNR-ISTC.

References

1. Bass, R., Pejcinovic, B., Grant, J.: Applying scrum project management in ECE curriculum. In: Proceedings of the Conference on Frontiers in Education, pp. 1–5. IEEE (2016)
2. Bird, C., Nagappan, N., Murphy, B., Gall, H., Devanbu, P.: Don't touch my code! Examining the effects of ownership on software quality. In: Proceedings of the 19th ACM SIGSOFT Symposium on Foundations of Software Engineering, pp. 4–14 (2011)

3. Chetankumar, P., Ramachandran, M.: Agile maturity model (AMM): a software process improvement framework for agile software development practices. Int. J. Softw. Eng. **2**, 01 (2009)
4. Ciancarini, P., Messina, A., Poggi, F., Russo, D.: Agile knowledge engineering for mission critical software requirements. In: Nalepa, G.J., Baumeister, J. (eds.) Synergies Between Knowledge Engineering and Software Engineering. AISC, vol. 626, pp. 151–171. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-64161-4_8
5. Ciancarini, P., Missiroli, M., Poggi, F., Russo, D.: An open source environment for an agile development model. In: Ivanov, V., Kruglov, A., Masyagin, S., Sillitti, A., Succi, G. (eds.) OSS 2020. IAICT, vol. 582, pp. 148–162. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-47240-5_15
6. Ciancarini, P., Missiroli, M., Russo, D.: Cooperative thinking: analyzing a new framework for software engineering education. J. Syst. Softw. **157**, 110401 (2019)
7. Ciancarini, P., Missiroli, M., Sillitti, A.: Preferred tools for agile development: a sociocultural perspective. In: Mazzara, M., Bruel, J.-M., Meyer, B., Petrenko, A. (eds.) TOOLS 2019. LNCS, vol. 11771, pp. 43–58. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29852-4_3
8. Ciancarini, P., Missiroli, M., Zani, S.: Empirical evaluation of agile teamwork. In: Paiva, A.C.R., Cavalli, A.R., Ventura Martins, P., Pérez-Castillo, R. (eds.) QUATIC 2021. CCIS, vol. 1439, pp. 141–155. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85347-1_11
9. Devedžić, V., et al.: Teaching agile software development: a case study. IEEE Trans. Educ. **54**(2), 273–278 (2010)
10. Ertmer, P.A., Newby, T.J.: The expert learner: strategic, self-regulated, and reflective. Instr. Sci. **24**(1), 1–24 (1996)
11. Gren, L., Goldman, A., Jacobsson, C.: Agile ways of working: a team maturity perspective. J. Softw. Evol. Process **32**(6), e2244 (2020)
12. Heberle, A., Neumann, R., Stengel, I., Regier, S.: Teaching agile principles and software engineering concepts through real-life projects. In: 2018 IEEE Global Engineering Education Conference (EDUCON), pp. 1723–1728. IEEE (2018)
13. Hoegl, M., Gemuenden, H.G.: Teamwork quality and the success of innovative projects: a theoretical concept and empirical evidence. Organ. Sci. **12**(4), 435–449 (2001)
14. Jacobson, I., Lawson, H., Ng, P., McMahon, P., Goedicke, M.: The Essentials of Modern Software Engineering. ACM Books, Morgan & Claypool Publishers (2019)
15. Jacobson, I., Sutherland, J., Kerr, B., Buhnova, B.: Better scrum through essence. Softw. Pract. Exp. **52**(6), 1531–1540 (2022)
16. Kemell, K.-K., Nguyen-Duc, A., Wang, X., Risku, J., Abrahamsson, P.: The essence theory of software engineering – large-scale classroom experiences from 450+ software engineering BSc students. In: Kuhrmann, M., et al. (eds.) PROFES 2018. LNCS, vol. 11271, pp. 123–138. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03673-7_9
17. Layman, L., Cornwell, T., Williams, L.: Personality types, learning styles, and an agile approach to software engineering education. In: Proceedings of the 37th SIGCSE Technical Symposium on Computer science education, pp. 428–432 (2006)
18. Lindsjørn, Y., Sjøberg, D.I., Dingsøyr, T., Bergersen, G.R., Dybå, T.: Teamwork quality and project success in software development: a survey of agile development teams. J. Syst. Softw. **122**, 274–286 (2016)

19. Marzolo, P., Guazzaloca, M., Ciancarini, P.: "Extreme development" as a means for learning agile. In: Succi, G., Ciancarini, P., Kruglov, A. (eds.) ICFSE 2021. CCIS, vol. 1523, pp. 158–175. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-93135-3_11
20. Meier, A., Kropp, M., Perellano, G.: Experience report of teaching agile collaboration and values: agile software development in large student teams. In: Proceedings of the 29th International Conference on Software Engineering Education and Training (CSEET), pp. 76–80. IEEE (2016)
21. Meyer, B.: Agile! The Good, the Hype and the Ugly. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05155-0
22. Missiroli, M., Russo, D., Ciancarini, P.: Learning agile software development in high school: an investigation. In: Proceedings of the 38th International Conference on Software Engineering Companion, pp. 293–302 (2016)
23. OMG. Essence - kernel and language for software engineering methods, version 1.2. Technical Report 18-10-02. OMG (2018)
24. Pal, K.: Reflection on teaching practice for agile methodology based product development management. In: Teaching Innovation in University Education: Case Studies and Main Practices, pp. 135–155. IGI Global (2022)
25. Péraire, C., Sedano, T.: Essence reflection meetings: field study. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–4 (2014)
26. Péraire, C., Sedano, T.: State-based monitoring and goal-driven project steering: field study of the SEMAT Essence framework. In: Companion Proceedings of the 36th International Conference on Software Engineering, Hyderabad, India, pp. 325–334. ACM (2014)
27. Quintanilla-Perez, D., Mauricio-Delgadillo, A., Mauricio-Sanchez, D.: Essboard: a collaborative tool for using Essence in software development. In: Proceedings of the 10th International Conference on Software Engineering and Service Science (ICSESS), pp. 20–23. IEEE (2019)
28. Rodríguez, G., González-Caino, P.C., Resett, S.: Serious games for teaching agile methods: a review of multivocal literature. Comput. Appl. Eng. Educ. **29**(6), 1931–1949 (2021)
29. Rush, D.E., Connolly, A.J.: An agile framework for teaching with Scrum in the IT project management classroom. J. Inf. Syst. Educ. **31**(3), 196–207 (2020)
30. Russo, D., Taccogna, G., Ciancarini, P., Messina, A., Succi, G.: Contracting agile developments for mission critical systems in the public sector. In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Society, pp. 47–56 (2018)
31. Schubanz, M., Lewerentz, C.: What matters to students - a rationale management case study in agile software development. In: Proceedings of the SEUH Software Engineering im Unterricht der Hochschulen, volume 2531 of CEUR Workshops Proceedings, Innsbruck, Austria, pp. 17–26 (2020)
32. Sutherland, J., Jacobson, I., Kerr, B.: Scrum essentials cards: experiences of scrum teams improving with essence. Queue **18**(3), 83–106 (2020)
33. Teel, S., Schweitzer, D., Fulton, S.: Teaching undergraduate software engineering using open source development tools. Issues Informing Sci. Inf. Technol. **9**, 63–73 (2012)
34. Tüzün, E., Üsfekes, Ç., Macit, Y., Giray, G.: Towards unified software project monitoring for organizations using hybrid processes and tools. In: Proceedings of the International Conference on Software and System Processes (ICSSP), pp. 115–119. IEEE (2019)

35. Agile alliance - 12 principles behind the agile manifesto (2001)
36. Villarrubia, A., Kim, H.: Building a community system to teach collaborative software development. In: Proceedings of the 10th International Conference on Computer Science & Education (ICCSE), Cambridge, UK, pp. 829–833 (2015)
37. Yin, A., Figueiredo, S., da Silva, M.M.: Scrum maturity model. In: Proceedings of the ICSEA, pp. 20–29 (2011)