An adaptive 3D virtual learning environment for training software developers in scrum

Ezequiel Scott a and Marcelo Campo b, c

a Institute of Computer Science, University of Tartu, Tartu, Estonia; b NICE, Facultad de Ciencias Exactas, UNICEN National University, Tandil, Argentina; c CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Buenos Aires, Argentina

ABSTRACT
Scrum is one of the most used frameworks for agile software development because of its potential improvements in productivity, quality, and client satisfaction. Academia has also focussed on teaching Scrum practices to prepare students to face common software engineering challenges and facilitate their insertion in professional contexts. Furthermore, advances in learning technologies currently offer many virtual learning environments to enhance learning in many ways. Their capability to consider the individual learner preferences has led a shift to more personalised training approaches, requiring that the environments adapt themselves to the learner. We propose an adaptive approach for training developers in Scrum, including an adaptive virtual learning environment based on Felder’s learning style theory. Although still preliminary, our findings show that students who used the environment and received instruction matching their preferences obtained slightly higher learning gains than students who received a different instruction than the one they preferred. We also noticed less variability in the learning gains of students who received instruction matching their preferences. The relevance of this work goes beyond the impact on learning gains since it describes how adaptive virtual learning environments can be used in the domain of Software Engineering.

ARTICLE HISTORY
Received 1 December 2020
Accepted 22 October 2021

KEYWORDS
Adaptive learning environments; scrum; agile software development; learning styles

1. Introduction
Teaching agile practices is at the cutting-edge of Software Engineering education since agile methodologies are widely used in the industry. As university-level teaching aims at preparing students for both facing common software engineering challenges and facilitating their insertion in professional contexts, the software engineering curricula should respond by providing students with knowledge and meaningful experiences on how to use Scrum in practice. Therefore, there is an increasing need to introduce Scrum into training courses, and many educators are faced with how to incorporate Scrum into their software engineering courses.

Over the last years, researchers have focussed on teaching Scrum practices and have proposed many approaches. These go beyond the traditional lectures in which all the students listen to the teacher and eventually take notes because, as a general consensus, educators claim that teaching Scrum should include some practical experience to strengthen comprehension and achieve deep learning (Mahnič, 2015). For this reason, the use of Scrum in capstone projects is the most adopted strategy worldwide. However, capstone projects may still preserve characteristics of one-
size-fits-all education, where the same teaching strategies are considered suitable for all kinds of learners.

For this reason, it is crucial to provide students with personalised learning scenarios that might offer more meaningful learning experiences. The personalisation of the learning process is aligned with learner-centred principles, in which professors should consider students’ learning preferences, their strengths and weaknesses (McCombs & Whisler, 1997). In this context, learning styles are not only useful tools to detect how learners perceive, interact with, and respond to the learning environment but also valuable indicators for the cognitive, affective, and psychological students’ behaviours (Keefe, 1988). Moreover, previous studies have shown that students’ learning experiences related to Scrum can be enhanced along with their educational outcomes when learning styles are considered (Scott et al., 2014a; Scott, Rodríguez, et al., 2016).

This paper proposes an adaptive 3D virtual learning environment in which Scrum topics are tailored to the developer’s learning style. We model the developer’s profile using the Felder-Silverman learning style model (Felder & Silverman, 1988). The profile is used by an adaptation mechanism responsible for adapting the virtual environment interface and providing learning content to the developer conforming to their learning style. The adaptation mechanism is included in a virtual environment that supports software development based on Scrum, namely Virtual Scrum (Rodriguez et al., 2015), which has shown favourable results in previous works and in the context of teaching Scrum. We conducted a controlled experiment with students, and our results show that students who used the environment and received instruction matching their preferences obtained slightly higher learning gains than students who received a different instruction than the one they prefer. These results are promising since they support the use of adaptive virtual learning environments in the domain of Software Engineering.

2. Related work

2.1. Virtual worlds in education

Virtual Learning Environments (VLEs) have the specific purpose of enabling teaching and learning by providing valuable features for students and teachers (Aldrich, 2009). Virtual Environments can also be considered part of a broader category: Virtual Worlds, as defined by Nevelsteen (2018). Virtual World is a more popular term that groups several interactive environments with different characteristics and properties. Virtual Worlds have been attractive to researchers in the educational field due to their potential benefits from a technological perspective.

Virtual worlds present unique characteristics that allow educators to achieve several ongoing challenges. Many recent studies (Duncan et al., 2012; Kavanagh et al., 2017; Lau & Lee, 2015; Park & Lee, 2003; Reisoğlu et al., 2017; Warburton, 2009) have shown that Virtual Worlds can contribute to engagement and interactivity, collaboration, experimentation, formation of group identity, and idea generation. Virtual worlds aim to foster students to participate in simulated activities and collaborate in distributed settings, similar to how they do face-to-face activities.

These benefits are partially due to the high fidelity to the reality that virtual worlds offer. For this reason, there is a trend towards building complex virtual environments in which both pedagogical and gaming elements take relevance. Numerous technical requirements challenge the development of virtual worlds with these characteristics; nevertheless, several technological solutions and platforms make creating Virtual Worlds easier.

Vircadia is a 3D open-source multiplayer software project that allows for creating social and educational environments. Vircadia consists of many projects and hundreds of contributors. Although the platform is relatively recent, researchers are paying attention to its capabilities for Virtual Reality (Kwon & Hudson-Smith, 2020). VoRtex (Jovanović & Milosavljević, 2019) is another platform for creating Virtual Worlds. One of the main elements of the platform is Vortex School, which adapts the concept of MicroLessons. MicroLessons include challenges like problem-solving.
puzzles for different use cases such as learning a new programming algorithm. This platform is free and open-source. Among other open-source initiatives for creating Virtual Worlds, we can mention Tivoli Cloud VR\(^3\) and Sansar.\(^4\) Popular games have also introduced creative modes to enable the use of a customised version of the game in the classroom. For instance, Fortnite Creative\(^5\) allows teachers to build a lesson plan to teach content-specific learning outcomes. Second Life\(^6\) is probably the popular and mature platform for creating Virtual Worlds (Reisoglu et al., 2017; Warburton, 2009). Second Life has many users that use the platform for many purposes, ranging from entertainment to education.

Virtual Worlds for education also present several issues and limitations that current studies have reported. For instance, Kavanagh et al. (2017) report on problems related to cost, training, and software and hardware usability. The authors also reported that students found the implementations to be insufficiently realistic. Girvan and Savage (2012) point out that Virtual worlds provide the researchers with new ethical issues to consider, such as obtaining informed consent, privacy protection and identity. In a meta-review (Reisoglu et al., 2017), the authors remark that virtual worlds should focus on developing high-order cognitive skills. Moreover, a better evaluation of these environments is needed to reveal students’ achievements more clearly. These findings are in line with other studies in the field (Scott, Soria, et al., 2016). Despite these issues, when virtual worlds are carefully designed and evaluated, their outcomes can be encouraging, which make them still prominent as a learning technology.

### 2.2. Approaches for teaching scrum

The Scrum framework is simple to understand yet difficult to master (Schwaber & Beedle, 2002). For this reason, many approaches have arisen to address this issue. There are mainly three directions that go beyond the traditional lectures on Scrum: teaching Scrum through the use of games, through the experience of capstone projects, and using the aid of virtual environments, sometimes including games. When Scrum is taught using games, learners follow the well-defined rules of the game in a partial or complete simulation of the software development process. These games are generally suitable for learners as their opinions about the learning process have been satisfactory. There are currently several examples of these learning approaches, such as the work of (Baumann, 2020; Fernandes & Sousa, 2010; Guzmán & López, 2019; Krivitsky, 2011; Paasivaara et al., 2014; Ramingwong & Ramingwong, 2015; von Wangenheim et al., 2013).

Another direction proposes courses that take advantage of capstone projects, in which professional working environments are simulated as much as possible. These courses require learners to work in teams to develop a software project, usually within the scope of a software engineering course. Capstone projects present many advantages, such as helping students to develop transferable skills that are better acquired within a practical project rather than through formal lectures. Several articles report the use of capstone projects for teaching Scrum (Devedzic & Milenkovic, 2011; Mahnič, 2012a, 2015; Rico & Sayani, 2009; Tan et al., 2008).

The third direction comprises different approaches in which software tools are crucial to support Scrum practices in educational and professional contexts (Wisnu Wirawan & Adhy, 2014; Yu et al., 2014). Several 2D tools can be used to support the teaching of Scrum, for instance (Begosso et al., 2019; Lee, 2016). However, 2D tools usually fail to represent the physical development environment to carry out a realistic simulation of the Scrum process. Virtual worlds arise as an adequate solution to address this problem because there is a lack of physical space and resources that prevent creating Scrum team rooms in universities. Moreover, recent studies have shown that Virtual Learning Environments rapidly become an integral part of teaching and learning in higher education (Callaghan et al., 2009). Video games and virtual worlds are moving into the mainstream as traditional media industries struggle to keep with up digital natives and their desire for information, technology and connectivity. Researchers in the field are convinced that these technologies are potentially profitable to education (Anderson, 2008; Rodriguez et al., 2012).
In the light of the above, 3D virtual environments allow developers to know about tasks performed by their peers and even hold meetings regardless of their physical location. Also, these environments can be used to provide a physical topology of both a software project and a process. This characteristic allows for faster access to information than a 2D tool plus video conference system to carry out meetings within a distributed team (Whitehead, 2007). This benefit may be attributed to the possibility of integrating isolated tools in a single development environment, leading to a context in which it is viable to share software artefacts. Therefore, using a virtual world has many advantages in comparison with 2D tools. For instance, students use avatars to communicate through gestures (Herbsleb, 2007), and they also can manipulate elements in the modelled room going through a 3D experience of Scrum. Additionally, the sense of immersion and the simulated team room, equipped with the Scrum artefacts and rules, is crucial to engaging users in a hands-on experience of using Scrum, participate in collaborative work, and improve their comprehension of this agile method.

Several studies have used virtual environments for teaching Scrum. For instance, the recent work of Radhakrishnan and Koumaditis (2020) proposes creating a collaborative virtual simulation of Scrum named VRScrum. SimScrum (Begosso et al., 2019) is another work in progress that proposes to use gamification principles to create a simulator for teaching Scrum in Software Engineering courses. There are also more mature projects with promising results. For instance, ScrumVR (Caserman & Göbel, 2020) is an educational game to teach Scrum through a game scenario where players have to participate in activities. The game uses virtual reality technologies, and experiments show that it is helpful to teach the foundations of Scrum. Lee (2016) proposes Scrum-X, another simulation game to teach Scrum. During the game, players can plan, execute and manage a software development project, and they will be able to experience the entire Scrum process. The author conducted a pilot study with promising results.

Virtual Scrum (Rodríguez et al., 2015) uses a virtual world to simulate a work environment handling 3D displays of the Scrum artefacts. Virtual Scrum also supports a task board for planning and tracking user stories; a daily meeting artefact for solving problems, adapting rapidly to changes and removing impediments; and a burn-down chart for reflecting on the past Sprint and making continuous process improvements during retrospective meetings. This way, Virtual Scrum uses a
virtual world to harness the 3D interfaces for training students in their performance in a simulated Scrum environment.

Table 1 summarises the recent virtual environments for teaching Scrum and compares them. We included JIRA in the table because of its popularity and extensive use for project management, although the main purpose of JIRA is not educational; thus, the tool serves as a baseline for feature comparison. Table 1 shows that student profiling and simulated scenarios are the less explored features when teaching Scrum using virtual environments.

In summary, there have been several research directions for teaching Scrum in academic and professional contexts. Although many approaches have shown improvements from many perspectives, they have neglected personalising issues. A recent literature review indicates that adaptive 3D virtual learning environments (Scott, Soria, et al., 2016) have shown to be suitable software tools to support personalised learning, providing learners with unique learning experiences due to their 3D features. Adaptive 3D VLEs have been used in many domains, but to the best of our knowledge, a specific adaptive environment still has not been developed for training developers in Scrum. In this paper, we introduce Adaptive Virtual Scrum, an extension of Virtual Scrum (Rodriguez et al., 2015), which includes an adaptation mechanism to support the personalised learning of Scrum.

3. Overview of the approach

The approach involves two actors: a developer and a coach. The main actor is the developer, who interacts with the environment to learn the concepts of Scrum. The coach has the responsibility of supervising the learning process. The coach might be a professional trainer in an industrial setting or a university teacher.

The approach exploits the benefits of capstone projects since they allow students to put Scrum into practise and simulate the professional working environment as much as possible. This way, the approach covers all the phases of the Scrum framework, from organising User Stories to delivering the product. Similarly to capstone projects, the coach sets the initial requirements for the project that students will develop. These requirements can be given to the students through an interview where they have to elicit the requirements directly from the product owner (i.e., the coach), a textual narrative, or even a set of Epics.

Figure 1 shows an overview of the proposed approach as an iterative process with intermediate steps in which the actors mentioned above are involved. The first step involves knowing the student’s learning style according to the Felder-Silverman model (Felder & Silverman, 1988). Currently, these values are obtained from the student’s answers to the ILS instrument, but these values may also be obtained through automatic detection of the student’s learning style, as shown by several studies (Feldman et al., 2015). The ILS instrument is a 44-item questionnaire that allows for reliably collecting the student’s learning style; the instrument is delivered through an online form connected to the environment.

The second step is one of the most important since the developer receives content according to their learning style. A component of the environment fires rules for tailoring the user interface according to the previously collected learner model, thereby adapting concepts and learning activities to the student’s learning style. Since our environment does not pretend to be an intelligent tutoring system that replaces the role of the teacher, we stress the importance of the role of the coach. The coach could provide the student with valuable feedback and answer additional questions that arise from the explanations of the topics. The environment also helps the coach deliver the Scrum topics because they are already organised and planned with learning activities.

The third step allows the developer to put the Scrum topics into practice. Working as part of a Scrum Team, the developer participates by following the Scrum topics learned before. The team start with the Sprint planning meeting and end with the Sprint Review and Sprint retrospective meeting. During the Sprint, team members meet regularly at Daily Scrum meetings to inform each other of their current activities and possible impediments.
The fourth step involves tracking information about how the developers address the Scrum practices since they can record all the information into the artefacts provided by the environment. For example, the environment provides a virtual Product Backlog in which the student can record their specification of the requirements in the form of User Stories and their tasks and estimates. This tracking is done similarly to when developers interact with project management tools such as JIRA.7

The fifth step takes part at the end of the Sprint, when developers, as a team, have to deliver the product increment to the product owner (i.e. the coach) by performing a Sprint Review meeting. Here, the coach analyses the tracking of the project by interpreting the software development metrics collected in step four; then, she/he makes suggestions and give feedback to the student. Finally, in the sixth step, the developer and the team are ready to advance to the next Sprint, repeating the second step when necessary, yet carrying out steps 3 to 6 until the end of the project.

### 3.1. An illustrative example

We introduce a simple example to illustrate the idea of the proposed approach. In this example, there are two developers, John and Mary. These developers are part of a Software Engineering Course, and they do not have any prior knowledge about Scrum. During the first step, the developers answer the online version of the ILS questionnaire (Felder & Soloman, 1997), which allows for obtaining the developer’s learning style (Felder & Silverman, 1988). We only concentrate on the processing dimension of the model because this dimension has shown to be strongly related to Scrum (Scott et al., 2013, 2014a). After completing the questionnaire, we know that John is an active developer whose learning preferences are aligned to studying concepts with real examples and conduct
group activities; in contrast, Mary is a reflective developer who prefers to receive theoretical explanations of the concepts and have space to reflect on them (Felder & Silverman, 1988).

In the second step, students receive tailored instruction according to their learning styles. The environment has a database with a predefined set of Scrum topics to tailor the instruction. That is, the environment can modify the user interface according to predefined teaching strategies suitable for active and reflective students. For instance, given the topics on User Stories’ definition, these contents are delivered John (active) using active strategies. They include showing a brief explanation of User Stories and their syntax, and delivering an activity that encourages John to create two User Stories from a narrative by interacting with other learners. On the other hand, Mary (reflective) receives the same content on User Stories definition but using passive strategies. They include showing the explanation on User Stories and their syntax and then triggering a question to reflect on the topic: “What could be the different roles, desires and benefits of the User Stories?”. Moreover, the coach might give additional explanations and examples on the topic or even answer questions.

Figure 2 depicts the environment delivering contents to the developers. In particular, Figure 2(a) shows the scenario in which John receive a brief explanation on User Stories and immediately is encouraged to discuss the building of User Stories with other developers. In contrast, Figure 2(b) shows the scenario in which Mary receive a deeper explanation of building User Stories and immediately how she is asked a question that allows her to reflect on the concept.

Step 3 involves fostering practical experience to strengthen the developer’s comprehension of Scrum. Our approach encourages developers to carry out a capstone project. The coach delivers to the developers the requirements for the capstone project in the form of Epics. This step requires that both John and Mary work in teams by following the Scrum practices learned before. Additionally, developers use the environment to track all the information about the execution of the practices, such as the User Stories they define, the story points they plan, the estimation values, and the status of the user stories they develop. At the end of the Sprint, teams provide an increment of the required functionality that is shown at the Sprint Review meeting.
After the Sprint Review meeting, the coach has the opportunity to give developers valuable feedback on their performance and explain the issues that eventually arise from the development process. Moreover, the developers and the coach can analyse the traditional Scrum metrics such as work capacity, the accuracy of estimation, and burn-down charts by using the environment. Once the feedback is given, both developers John and Mary are ready to continue moving through the next Sprints by the same process.

4. The adaptive virtual environment

Adaptive virtual learning environments are computer-based systems that support the user in acquiring knowledge by adapting the system to the individual user (Kavcic, 2000; Wenger, 2014). To be able to adapt itself to an individual user, the system has to be aware of the individual characteristics of the user as well as the teaching domain. Moreover, the system has to have a mechanism for tailoring contents according to the user model. Hence, the user, the domain knowledge, and the adaptation mechanism can be considered essential parts of any adaptive system. In the following sections, we show how these components are represented in the context of this study.

4.1. The learner model

Defining the learner model involves any issues related to identifying, representing, and updating the learner’s information. This model is considered to be crucial since it is the primary source used by adaptive systems to achieve the learning personalisation (Brusilovsky, 2001; Cristea & Garzotto, 2004). In this study, the user may be a student in a Software Engineering course or even a novice software developer in an industrial setting. Since our approach aims to align the learning process to the learner-centred principles, we consider the individual characteristics and preferences in our learner model.

To model these learner’s characteristics, we rely on learning styles as useful indicators since they are defined as the characteristic cognitive, affective, and psychological behaviour that serve as relatively stable indicators of how learners perceive, interact with, and respond to the learning environment (Keefe, 1988). Many studies report that the usage of learning styles in teaching is a factor that can improve the quality of education (Felder & Spurlin, 2005; Hawk & Shah, 2007). Out of the learning style models, we use the Felder-Silverman Learning Style Model (FSLSM) (Felder & Silverman, 1988).

Three main reasons support the selection of the FSLSM. First, previous evidence on the relationship between Scrum and learning styles in the context of an undergraduate software engineering course was found (Scott et al., 2014a; Scott, Soria, et al., 2016). Second, the FSLSM has been widely used in the context of Computer Science education (Graf et al., 2009). Third, the learning styles can be obtained using the Index of Learning Styles (ILS) instrument, a questionnaire based on 44 items which each student responds to according to their learning preferences (Felder & Spurlin, 2005). The instrument has shown to be reliable and valid (Felder & Spurlin, 2005), and can be administered to students by an online form.

The results obtained from the ILS distinguish the learners’ preferences according to the four dimensions of the model, which are perception \((D_1)\), understanding \((D_2)\), processing \((D_3)\) and input of information \((D_4)\). Since the ILS instrument determines the learning styles according to a numeric scale with values ranging from \(-11\) to \(+11\) on each dimension, we represent the learner model using the developer’s learning style. Therefore, the learner model is represented as a 4-tuple of integer values:

\[
LS = (D_1, D_2, D_3, D_4) \quad \text{with} \quad D_i \in \mathbb{Z} \land D_i \in [−11, +11] \quad (1)
\]

In particular, our approach is based on the processing \((D_3)\) dimension, which is in line with the active/reflective styles. According to the FSLSM, active students prefer doing tasks or talking about concepts,
while *reflective* students are likely to manipulate and examine the information introspectively (Felder & Silverman, 1988).

We rely on the processing dimension because of two main reasons. First, this dimension has shown to be strongly related to the performance of Scrum practices (Scott et al., 2013; Scott, Rodríguez, et al., 2016). Second, we aim at simplifying the learner model to achieve adaptivity straightforwardly since considering all the dimensions may hinder the definition of the instructional strategies. Notice that according to the manual for interpreting the ILS instrument (Felder & Solomon, 1997), the active/reflective styles are determined as:

\[
D_3 = \begin{cases} 
  \text{active} & \text{if } D_3 > 0; \\
  \text{reflective} & \text{otherwise} 
\end{cases}
\]

The FSLSM presents a teaching model that defines general teaching strategies suitable for each learning style. This teaching style model classifies instructional methods according to how well the learning styles are addressed. Although these instructional methods pretend to be general and suitable for any domain of knowledge, they require teachers to tailor the concepts according to their field. For this reason, it is crucial to adequate the set of Scrum topics using the guidelines of the teaching model.

### 4.2. The contents and the instructional strategies

The contents and the instructional strategies are relevant for adaptive educational systems since they determine the learning experience achieved by the learner. Moreover, the strategies and contents and learning styles can enhance learning because the students’ differences are considered when learning. In the context of this study, the teaching domain (also known as the domain model) comprises a set of topics that arise from the Scrum framework, which defines a set of software engineering practices.

The set of topics relies on an existing teaching model (Scott et al., 2014b). The authors have defined the model as the result of several years of teaching software engineering practices, including topics and instructional strategies that define the syllabus. The topics are User Story definition, User Story splitting, User Story estimation and planning, Sprint review, Daily Scrum, and Sprint retrospective meetings.

The main goal of the syllabus is that students learn the practices suggested by the Scrum framework. In particular, the students should learn that a correct user story description must have the following elements: role, desired feature, acceptance criteria and an adequate level of granularity. Moreover, it may include non-functional requirements when necessary (Cohn, 2004). In case of complex user stories or epics, students should learn to apply user story splitting to obtain simpler user stories. Regarding user stories estimation, Release and Sprint plans are created by the Team during their respective planning meetings using the estimates assigned to the user stories (Cohn, 2005; Schwaber & Beedle, 2002). As a rule, these estimates are described using story points, representing the complexity of a User Story. Students should learn that the Team has to estimate the number of story points that can be developed during a Sprint at the beginning of each of them.

Once a Sprint has started, the Team further decomposes each User Story into its constituent tasks that must be performed to deliver a required functionality by the end of the Sprint. Students should also understand that the team is responsible for assigning the estimated duration in hours to each task of the Sprint. At this point, students are encouraged to analyse the relationship between story points and working hours. They should become aware that this relationship is not a fixed one (e.g., 1 point does not equal 8.3 hours). Moreover, they should learn the importance of estimating tasks in hours to confirm whether they have assigned an appropriate amount of work to the Sprint. Another thing we mention is that tasks are to be carried out by one student at a time as they emerge throughout the Sprint. Regarding user stories planning, we provide the students with the well-established group estimation technique called Planning Poker, which is recommended when using agile
software development methods for estimating the size of user stories and developing release and iteration plans (Mahnič, 2012b).

The syllabus should be tailored according to the students’ learner model to achieve adaptation. In this case, the learner model is based on the students’ learning styles, in particular, the active/reflective learning style dimension of the FSLSM. The learner model is crucial since it allows the system to select the appropriate way to deliver the content to the learner. Then, we define two instructional methods by following the guidelines of the original teaching style model (Felder & Silverman, 1988): the passive instructional method for delivering content in a suitable way for reflective students and the active instructional method for delivering content in a suitable way for active students.

4.3. The adaptation mechanism

The adaptation mechanism involves the techniques used by the environment to adapt itself according to the learner model and the instructional strategies. Although the two latter ones are closely related to the resulting learning experience, the adaptation mechanism is also responsible for it. In this study, the adaptation mechanism uses the previously defined learner model based on learning styles, the contents related to the Scrum topics, and instructional strategies according to teaching styles.

The approach requires representing the syllabus using a directed graph, where the nodes represent the topics on the syllabus, and the arcs determine the order in which topics must be delivered. Figure 3(a) shows an example of this structure. Since this structure represents a unique way (i.e. a unique instructional method) to deliver the topics to any student, we can say that it represents a traditional content delivery. In contrast, Figure 3(b) depicts an adaptive approach for delivering content, in which different ways of delivering the same topic are taken into account (i.e. two different instructional methods). As shown in the graph, some topics (nodes a, b, and d) could be delivered in the same way since they do not require any adaptation, whereas the topic c is tailored according to the student’s learning style. Thus, the original topic c in the traditional delivery of content is mapped to two alternatives depending on the student’s learning style: node cr for reflective students, and nodes ca1 and ca2 for active students.

Then, for an reflective student, the topic c is delivered through the active instructional method (node cr), requiring performing the actions ii_reflective and iii_reflective. As for an active student, the

Figure 3. Comparison of content delivery approaches. Nodes a, b, c, d represent the contents delivered to the learner whereas the arcs represent the way in which the contents are delivered (i.e. instructional method).
same topic $c$ is delivered through the passive instructional method (node $c_{a1}$ and node $c_{a2}$), requiring performing the actions $ii_{active}$, $iii_{active}$ and $iv_{active}$.

Let us illustrate the adaptive content delivering by the following example. For instance, the nodes $a, b, c, d$ represent a portion of the syllabus, where $a =$ “User Stories definition”, $b =$ “User Story parts”, $c =$ “Writing User Stories”, and $d =$ “User Story Splitting”. Thus, the structure of the graph determines that learners should receive the topic “User Stories definition” before “User Story parts” since the latter could be difficult to understand without the former. Moreover, the topic $c =$ “Writing User Stories” is taught through the passive method when the student’s learning style is reflective. For this topic, this method requires three steps. First, the reflective student has to complete the previous topic (action $i_{reflective}$). Second, the environment has to show the content (node $c_{r} =$ “Think about different roles that could need something from the system. Write the requirements as User Stories.”). Third, the student has to resolve the given assignment (action $iii_{reflective}$). Figure 4 shows an example of how the topics are mapped to dialogues in the User Interface of the adaptive environment.

On the other hand, the same topic $c$ is taught through the active method when the student’s learning style is active. For this topic, the method requires five steps. First, the active student has to complete the previous topic (action $ii_{active}$). Second, the environment has to show the content (node $c_{a1} =$ “In group, identify and write the parts of the following User Story.”). Third, the

![Figure 4](image-url). The key to adaptivity in the environment is the domain model behind the User Interface.
student has to complete the required information by working in a group with others (action $iii_{active}$). Fourth, the environment has to show the following content associated with topic $c$ (node $c_{a1} =$ “In group, talk about different roles that could need something from a given system. Then, elaborate new User Stories.”). Fifth, the student has to resolve the assignment by working in a group (action $iv_{active}$). In this case, the active method introduces an additional node in comparison with the passive method.

This way, the mechanism enables the delivery of the topics in an adaptive way. More formally, we straightforwardly define the mechanism by using a rule-based approach. Although this approach is simple to implement, it has been used for most adaptive virtual learning environments (Scott, Soria, et al., 2016). An adaptation rule is associated with a topic, the source topic, and is used to control the transition to the next topic, the target topic. Such a rule consists of two parts: a condition part and an action part. The condition part is used to specify when the learner can be directed from the current topic (the source topic) to the next topic (the target topic). The condition is, in general, based on two aspects: the student’s learning style and the suitability of the following topic for the learner. The learning style is stored in the learner model, whereas the suitability for the next topic is given by the underlying structure of the domain model. The action part is the consequence of the condition part, representing a predefined change in the user interface that aims at guiding the learner through the topic.

5. Experimental evaluation

In order to evaluate the proposed approach, we conducted a controlled experiment. The objective is to study the impact of using our adaptive virtual learning environment to support the training of developers in Scrum by taking into account their learning styles. We focussed on analysing the learning gains. The data that support the findings of this study are available on request from the corresponding author. The following sections describe the study design, the measures used, and the experimental results.

5.1. Study design

The study follows the experimental design guidelines suggested by Pashler et al. (2008), and it is organised in five main steps. First, we assign students to two groups according to their learning style: active and reflective students. We used a convenience sampling strategy to select the participants since the authors were teaching the subject of interest: Software Engineering. We used the ILS questionnaire to obtain the students’ learning styles, and we only consider the processing dimension of the FSLSM. At this point, we assess the initial students’ understanding of Scrum using a test (pre-test). We use an adapted version of the instrument initially proposed by Scott, Rodriguez, et al. (2016), whose quality has been analysed through item analysis.

Second, students within each learning-style group were randomly assigned to two different instructional methods: active and passive. This way, both groups have the same number of active and reflective students. This setting also allows us to compare the measures of a control group with the experimental group. Thus, the control group is given by the students who receive instruction through a method opposed to their learning style (e.g. a reflective student receives instruction based on the active method). In contrast, the experimental group is given by the set of students who receive instruction through a method that is suitable to their learning style (e.g. an active student receive instruction based on the active method) (Pashler et al., 2008).

Third, we apply Phase I of the approach. Here, the control and experimental groups receive instruction accordingly. The authors delivered the content with the support of the adaptive virtual environment.

Fourth, after students received the training, we apply Phase II. Here, students are randomly assigned to Scrum teams (approx. five team members each), and they are asked to carry out a
The goal of the capstone course was to build a simple toy application by following the practices learnt. The teams were encouraged to use the adaptive virtual environment for managing all the artefacts needed.

Finally, after one iteration of 2 weeks, we analyse the students’ understanding of Scrum topics and their performance during the capstone project. To assess the students’ understanding at the end of the Sprint, we evaluate the students with the test on Scrum (post-test). Then, we calculate the students’ learning gains based on the scores of the pre and post-tests.

The methods used for the data analysis were mainly descriptive statistics due to the small sample size. However, we still studied the statistical significance of the difference in the mean scores of both groups: students who were taught with suitable instructional methods (experimental group) and students who were taught in a way that did not correspond to their learning style (control group). We conducted a t-test for independent samples (Montgomery, 1984) and we also verified its assumptions.

5.2. Learning gains

We measure students’ learning improvement between the beginning and end of the experience by calculating their learning gains. Equation (2) shows the formula used, where the numerator gives the absolute improvement in the student’s score, and the denominator is a correction factor. This factor acknowledges that it is easier for an initially low-scoring student to have a larger absolute improvement than an initially high-scoring student since they have more opportunity to change from incorrect to correct answers.

\[ g_i = \frac{\text{post}_i - \text{pre}_i}{100 - \text{pre}_i} \]  

(2)

5.3. Results

The experiment was conducted in the first semester of 2016. Twenty-six students participated in the experiment, with 22 (85%) identified as male and 4 (15%) as female. Participants’ ages ranged from 22 to 28 years. The participants were enrolled in the Software Engineering Workshop course, which is part of the Software Engineering curriculum at the National University of Central Buenos Aires (UNICEN). At the time of conducting the experiment, students were enrolled in university studies for 4 to 10 years, and most of the students were undergraduate (92%). In the previous years of the curriculum, students learnt about operating systems, networking, software systems’ design, object-oriented programming, and database management. Table 2 shows the demographic profile of the students.

Out of the twenty-six participants, 6 students were reflective, whereas 20 students were active. Within each style group, we divided the students into two groups according to the instructional methods. The first group contained 11 students, 9 of whom were active, and the remaining were reflective; and the second group contained 15 students, 11 of whom were active, and the remaining were reflective. Therefore, the control group (i.e., active-reflective and passive-active) and the experimental group (i.e., active-active and passive-reflective) consisted of 13 students.

A comparison of the results for both the control and experimental groups is shown in Table 3. The table shows the learning gains of the groups in terms of their arithmetic mean (\(M\)), median (\(ME\)), and standard deviation (\(SD\)). The results show that the experimental group obtained slightly higher learning gains. Moreover, there is less variation of the learning gains in the experimental group. Figure 5 shows the histogram of the learning gains of both groups.

Although the number of students who participate in the study represents a typical cohort size at the University of X, the sample becomes small when broken into groups. Nevertheless, we still
studied the statistical significance of the results by conducting a t-test of independent samples. We first verified its assumptions.

The first assumption is that both samples are drawn from independent populations, which is assured by the experimental design. The second assumption is that the populations can be described by a normal distribution, which is verified by the shape of the density curves (KDE) shown in Figure 5 and the results from the Shapiro-Wilk normality test \(W_{\text{Control}} = 0.944, p_{\text{Control}} = 0.455; W_{\text{Experimental}} = 0.895, p_{\text{Experimental}} = 0.114\).

The third assumption refers to the variance homogeneity of both groups, which is supported by the results from the Levene’s test \(F = 4.339, p = 0.048\). Since there are significant differences in the variances of both groups, we conducted the t-test considering different variances. The results of the t-test show that there is no statistically significant difference between both means \(p = 0.780\).

### 6. Limitations

There are several limitations in the evaluation of the proposed approach. Firstly, the notion of learning styles has been controversial for researchers (Pashler et al., 2008) and should generally be treated with caution. Although there is evidence that learners have different learning preferences, there is no general agreement on the best learning style model to be used. In consequence, it might result controversial whether active and passive learning styles exist. However, the environment can be extended to support any other learning model if practitioners align with a different school of thought.

During the definition of the learner model, we have only considered the processing dimension of the FSLSM since other studies supported the relationship between the processing dimension of the FSLSM and the way Scrum practices are performed (Scott, Rodriguez, et al., 2016; Scott & Soria, 2012). However, each student has a learning style that is simultaneously described by the four FSLSM

### Table 2. Demographic profile of participants.

| Variable         | Number | Percentage |
|------------------|--------|------------|
| Gender           |        |            |
| Female           | 4      | 15%        |
| Male             | 22     | 85%        |
| Age              |        |            |
| 22               | 4      | 15%        |
| 23               | 3      | 11%        |
| 24               | 7      | 27%        |
| 25               | 5      | 19%        |
| 26               | 5      | 19%        |
| 27               | 1      | 4%         |
| 28               | 1      | 4%         |
| Years in university |      |            |
| 4                | 7      | 27%        |
| 5                | 5      | 19%        |
| 6                | 7      | 27%        |
| 7                | 5      | 19%        |
| 8                | 1      | 4%         |
| 10               | 1      | 4%         |
| Degree           |        |            |
| Undergraduate    | 24     | 92%        |
| Postgraduate     | 2      | 8%         |

| Variable         | Number | Percentage |
|------------------|--------|------------|
| Table 3. Learning gains of the control and experimental groups.

| Group      | M       | ME      | SD      | n  |
|------------|---------|---------|---------|----|
| Control    | 0.439   | 0.458   | 0.175   | 13 |
| Experimental | 0.454  | 0.450   | 0.067   | 13 |
dimensions (Felder & Silverman, 1988); then, further studies are required to analyse how the dimensions are related to the performance of Scrum.

We have also used Scrum to develop a teaching approach that covers a defined set of practices. Although these practices are general, our approach has only been applied and could only be applied to analyse students’ behaviours when learning Scrum. That is, more research is needed to apply it to another agile methodology.

Regarding the reliability and validity of the instruments used, the ILS (Felder & Silverman, 1988) has been discussed in a previous study (Felder & Spurlin, 2005). The instrument’s validity used to test the students’ understanding has been reviewed by the authors and one external researcher. Although the original authors of the test analysed the test quality using item analysis (Scott, Rodríguez, et al., 2016), further studies are needed to determine the reliability of the test.

Thirdly, although we have used Virtual Scrum as the base tool where we added our adaptive mechanism, the adaptive mechanism can be applied to other tools. For instance, project management tools such as JIRA9 could be used. However, if tools were changed, data associated with the studied variables should allow for comparative analysis. Moreover, security measures should be taken into account regarding data handling and transmission. Since students/developers and project data are considered sensitive information, explicit agreement according to standards and context-dependent policies should be considered between the parties.

Finally, it is worth noting that our approach is sensitive to the characteristics of the academic context, such as the students’ motivation, the Product Owner’s pressure, and the contents of the course. Another limitation is also related to the generalisation of our results; since we used a specific student sample, we can only draw conclusions for this particular population. Thus, we can state that differences in cultural and educational backgrounds could impact on the suitability of instructional activities for the students.

7. Conclusion

We propose an adaptive approach to deal with the problems and effort associated with personalising the training in Scrum. The central thesis of this research was that it is feasible to train software developers in Scrum by introducing an adaptive learning environment that tailors Scrum topics according to their learning style. We validated our ideas by introducing an adaptive virtual environment capable of tailoring and delivering Scrum topics to the learners according to their learning style. In particular, we used the processing dimension of the Felder-Silverman learning style model. To achieve adaptivity in the environment, we proposed an adaptation mechanism responsible for adapting the virtual environment interface taking into account the most suitable
instructional method for the developer. In this sense, the proposed adaptation mechanism is the key to personalising the training process.

We conducted an experiment with Software Engineering students, and our preliminary results indicate no significant increase in the students’ learning gains. However, the results show less variance in the learning gains of students who used the adaptive approach, which suggests that the approach might reduce the dispersion of the learning gains. It is worth mentioning that these results are drawn from a small sample; therefore, more research is required to show significant results when the adaptive virtual environment is used. Our approach shows great potential for providing training to software developers due to the virtual and collaborative features.

These findings provide several insights for future research. In order to generalise the findings, more experiments are needed. In this line, replicated experiments are encouraged to widen the evaluation results by studying larger samples. The application of the approach in professional contexts such as industrial companies is worth exploring. Another future research line is related to studying how variations of the approach affect students’ performance. These variations could be related to using a different learner model (i.e., an alternative learning style model), different learners’ performance metrics (e.g., using Scrum performance metrics), and different teaching strategies. In all cases, our approach serves as a platform to support the design of these studies. We consider that our approach represents a step towards better scrum learning using adaptive learning environments.

Notes
1. https://github.com/vircadia/vircadia
2. https://github.com/Aca1990/VoRtex-School
3. https://github.com/tivolicloud/
4. https://www.sansar.com/
5. https://www.unrealengine.com/en-US/onlinelearning-courses/teaching-with-fortnite-creative
6. https://secondlife.com/
7. JIRA Website – https://www.atlassian.com/software/jira
8. Translated version available as supplementary material.
9. JIRA web site – https://www.atlassian.com/software/jira

Disclosure statement
No potential conflict of interest was reported by the author(s).

Funding
This work was supported by the National Scientific and Technical Research Council (CONICET), Argentina. Ezequiel Scott is currently founded by the Estonian Center of Excellence in ICT research (EXCITE) of the Estonian Research Council.

Notes on contributors
Ezequiel Scott received his Ph.D. in Computer Science degree from Universidad Nacional del Centro de la Provincia de Buenos Aires (UNICEN), Argentina, in 2016. He is currently a Lecturer of Software Engineering at the Institute of Computer Science, University of Tartu, Estonia. His research interests are in the areas of agile software development, software engineering education, and applied machine learning to aid teams to develop software in a better way.

Marcelo Campo received the Computer Engineer degree from Universidad Nacional del Centro de la Provincia de Buenos Aires (UNICEN), Tandil, Argentina, in 1988 and the Ph.D. degree in Computer Science from Instituto de Informática de la Universidad Federal de Rio Grande do Sul (UFRGS), Brazil, in 1997. He is currently a Professor at the Faculty of Exact Sciences, UNICEN, and principal researcher at the National Scientific and Technical Research Council (CONICET). He has been the director of the ISISTAN Research Institute (CONICET – UNICEN) and currently the director of NICE research group. His research interests include intelligent aided software engineering, software architecture and frameworks, agent technology, and software visualisation.
Ezequiel Scott http://orcid.org/0000-0001-5760-8226

References

Aldrich, C. (2009). Virtual worlds, simulations, and games for education: A unifying view. Innovate: Journal of Online Education, 5(5), s5.

Anderson, T. (2008). The theory and practice of online learning. Athabasca University Press.

Baumann, A. (2020). Teaching software engineering methods with agile games. In 2020 IEEE global engineering education conference (EDUCON) (pp. 1550–1553). IEEE. https://doi.org/10.1109/EDUCON45650.2020.9125129

Begosso, L. R., Franco, L. H. B., da Cunha, D. S., & Begosso, L. C. (2019). SimScrumF: A game for supporting the process of teaching Scrum. In Proceedings of the 9th international conference on information communication and management (pp. 27–31). ACM. https://doi.org/10.1145/3357419.3357426

Brusilovsky, P. (2001). Adaptive hypermedia. User Modeling and User-Adapted Interaction, 11(1–2), 87–110. https://doi.org/10.1023/A:101143116306

Callaghan, M., McCusker, K., Losada, J. L., Harkin, J., & Wilson, S. (2009). Teaching engineering education using virtual worlds and virtual learning environments. In International conference on advances in computing, control, and telecommunications technologies (pp. 295–299). IEEE. https://doi.org/10.1109/ACT.2009.80

Caserman, P., & Göbel, S. (2020). Become a scrum master: Immersive virtual reality training to learn scrum framework. In Joint international conference on serious games (pp. 34–48). Springer. https://link.springer.com/chapter/10.1007%2F978-3-030-61814-8_3

Cohn, M. (2004). User stories applied for agile software development. A-Wesley.

Cohn, M. (2005). Agile estimating and planning. Pearson Education.

Cristea, A., & Garzotto, F. (2004). Designing patterns for adaptive or adaptable educational hypermedia: A taxonomy. In World conference on educational multimedia, hypermedia and telecommunications (Vol. 2004, pp. 808–813). Association for the Advancement of Computing in Education (AACE).

Devedzic, V., & Milenkovic, S. (2011, May). Teaching agile software development: A case study. IEEE Transactions on Education, 54(2), 273–278. https://doi.org/10.1109/TE.2010.2052104

Duncan, I., Miller, A., & Jiang, S. (2012). A taxonomy of virtual worlds usage in education. British Journal of Educational Technology, 43(6), 949–964. https://doi.org/10.1111/bjet.2012.43.issue-6

Felder, R. M., & Silverman, L. K. (1988). Learning and teaching styles in engineering education. Engineering Education, 78(7), 674–681.

Felder, R. M., & Soloman, B. (1997). Index of learning styles questionnaire. Retrieved February 15, 2013, from http://www.ncsu.edu/felder-public/ILSpage.html

Felder, R. M., & Spurlin, J. (2005). Applications, reliability, and validity of the index of learning styles. International Journal of Continuing Engineering Education and Life-Long Learning, 27(1), 103–112.

Feldman, J., Monteserin, A., & Amandi, A. (2015). Automatic detection of learning styles: State of the art. Artificial Intelligence Review, 44, 157–186. https://doi.org/10.1007/s10462-014-9422-6

Fernandes, J. M., & Sousa, S. M. (2010). Playscrum-a card game to learn the scrum agile method. In 2010 Second international conference on Games and virtual worlds for serious applications (vs-games) (pp. 52–59). IEEE. https://doi.org/10.1109/VS-GAMES.2010.24

Girvan, C., & Savage, T. (2012). Ethical considerations for educational research in a virtual world. Interactive Learning Environments, 20(3), 239–251. https://doi.org/10.1080/10494820.2011.641678

Graf, S., Liu, T. C., Kinshuk, D., Chen, N. S., & Yang, S. J. (2009). Learning styles and cognitive traits – their relationship and its benefits in web-based educational systems. Computers in Human Behavior, 25(6), 1280–1289. https://doi.org/10.1016/j.chb.2009.06.005

Guzmán, J. C., & López, G. (2019). Teaching scrum using gamification. In Multidisciplinary digital publishing institute proceedings (Vol. 31, p. 7). MDPI. https://doi.org/10.3390/proceedings2019031007m

Hawk, T. F., & Shah, A. J. (2007). Using learning style instruments to enhance student learning. Decision Sciences Journal of Innovative Education, 5(1), 1–19. https://doi.org/10.1111/dsj.2007.5.issue-1

Herbsleb, J. D. (2007). Global software engineering: The future of socio-technical coordination. In 2007 future of software engineering (pp. 188–198). IEEE. https://doi.org/10.1109/FOSE.2007.11

Jovanović, Ć., & Milosavljević, A. (2019). VoRtex enterprise: Decentralized virtual reality blockchain-based platform. Facta Universitatisis, Series: Automatic Control and Robotics, 18(1), 57–77. https://doi.org/10.22190/FACR.2017.11

Kavanagh, S., Luxton-Reilly, A., Wueniche, B., & Plimmer, B. (2017). A systematic review of virtual reality in education. Themes in Science and Technology Education, 10(2), 85–119.

Kavic, A. (2000). The role of user models in adaptive hypermedia systems. In 2000 10th Mediterranean electrotechnical conference, Mелеcon 2000 (Vol. 1, pp. 119–122). IEEE. https://doi.org/10.1109/MELCON.2000.880382

Keefe, J. W. (1988). Profiling & utilizing learning style. ERIC.
Warburton, S. (2009). Second life in higher education: Assessing the potential for and the barriers to deploying virtual worlds in learning and teaching. British Journal of Educational Technology, 40(3), 414–426. https://doi.org/10.1111/j.1467-8535.2009.00952.x

Wenger, E. (2014). Artificial intelligence and tutoring systems: Computational and cognitive approaches to the communication of knowledge. Morgan Kaufmann.

Whitehead, J. (2007). Collaboration in software engineering: A roadmap. In Future of software engineering, 2007. FOSE’07 (pp. 214–225). IEEE. https://doi.org/10.1109/FOSE.2007.4

Wisnu Wirawan, P., & Adhy, S. (2014, November). Open source tools to support teaching agile software development. In International conference on education and technology and and sciences.

Yu, L., Surma, D. R., & Hakimzadeh, H. (2014). Incorporating free/open-source data and tools in software engineering education. In Open source technology: Concepts, methodologies, tools, and applications (pp. 381–391). IGI-Global.