International Journal of Software Engineering and Knowledge Engineering

World Scientific
www.worldscientific.com

APPLING SCRUM TO KNOWLEDGE TRANSFER AMONG SOFTWARE DEVELOPERS

Fernando Ibarra-Torres

Universidad Nacional de La Plata, UNLP,
La Plata, Buenos Aires,1900, Argentina
oscar.ibarrat@info.unlp.edu.ar

Matias Urbieta

Universidad Nacional de La Plata, UNLP,
La Plata, Buenos Aires,1900, Argentina
matias.urbieta@lifia.info.unlp.edu.ar

Nuria Medina-Medina

University of Granada, ETSIIT, CITIC-UGR,
Granada,18071, Spain
nmedina@ugr.es

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

In the teaching-learning processes, research and continuous innovation are encouraged. Now, when talking about innovation, the concept of adapting method-ologies that have been successfully applied to speed up and increase the quality of software projects begins to emerge, such is the case of agile software development methodologies.
Scrum is one of the most widely used methodologies in the software development process, it increases productivity and deliverable quality of software development teams, but there is not much evidence of how to use it to structure learning content and its use to transfer new knowledge in training or software development contexts. To fill this gap, the present research analyzes the applicability of Scrum for transferring new knowledge among software developers is developed.
In this article we describe the details of how to start transferring knowledge using Scrum, guiding the reader through its standards, processes, phases, and objectives. We also report an experiment for improving students' performance to support the approach's benefits in the academy context and a case study per-formed inside the company context.

Keywords: Knowledge transfer; Agile methodologies; Scrum; Software developers.

1. Introduction

Agile methods for software development emerged as a way to improve efficiency in production and quality of the final product, through teams of high-performance software developers [1]. Experience has shown that the adoption of agile methods improves the management of the development process and customer engagement [2], and decreases the amount of effort to produce deliverables [3]. If we consider these benefits, the application of agile methodologies in other domains that also im-ply an important volume of

knowledge transmission would allow taking advantage of the benefits reported by this type of methodologies.

According to the study [2], agile methodologies teaching is becoming an important part of the Software Engineering curriculum, because it helps and guides software developers to adapt to changes and to seek success in the software industry. Since students know the philosophy and dynamics of agile methodologies, there is an opportunity to apply an agile methodology to improve knowledge transmission process-es in academic environments and, after their graduation, in future organizations where they will work as software developers or with other software engineering pro-files.

The transfer of knowledge between people in the same organization (industrial or academic) faces major problems [4]. For example, the high staff turnover is stressing the company adaptation as good and skilled developers move to other companies. This implies that their knowledge is no longer an asset [5]. With this precedent, the need to innovate in the ways to transfer new knowledge arises, and the reuse of agile methodologies within the software industry to structure content and seek to transmit new knowledge among software developers. This will be an important point to help training the employees effectively, without harming productivity, or affecting the work dynamics among them, [6]. In educational contexts, these methodologies can be used to improve the knowledge transfer between the teacher and the students or between the students themselves when working in groups or teams, where the volume of knowledge to be transmitted is particularly high. [10][29].

To date, there are attempts to introduce agile methods to transfer new knowledge focused on courses and small projects in academic institutions [10], where the most important element to guide students is the teacher and their expert knowledge. Different studies [6],[7] identify that there is a gap in the use of different agile methodologies for knowledge transfer and whether the results are the same in different contexts in addition several of these investigations are more theoretical than practical However, in most of these experiences, the time spent to learn certain content has been long-er than initially contemplated [2]. In the software industry, the role of the professor is no longer present, and it is the software developers themselves who, based on factors such as experience and practice, begin to take on importance at the moment of sharing knowledge. Unlike in the educational field, the time factor in the labor field is much more restrictive and critical.

We can expect that the use of agile methodologies in real problems for acquiring new knowledge in the software industry is a non-trivial task that will involve considering the labor factors such as deadlines and economic resources, as well as the fellow-ship and work environment in the team of programmers. However, its use can be expected to maximize the benefits of knowledge transfer between the team and the client, with the use of Scrum's roles and artifacts combined with the teaching and learning process. For this reason, it is necessary to achieve an agile knowledge transfer approach that is well adapted to both academic and development contexts and al-lows in both cases to reduce learning times.

This paper presents a novel approach to perform knowledge transfer relying on Scrum methodology in both academic and Software industry contexts. We propose this

methodology to be carried out in two scenarios, the first to be carried out at the same time as the daily activities of software developers to minimize losing developer's productivity, and the second when there is a need to learn the use of a new tool that is normally necessary in the software industry to innovate. Within the industry just as it happens in a classroom, there are roles to be filled according to the scenario, taking into account that even individuals may have special capabilities to transmit knowledge [12]. To fulfill the process of knowledge transmission within the industry, the use of additional tools such as training courses can be raised to transmit knowledge effectively thus helping software developers. In fact, in the educational field a similar situation occurs, some teachers also have difficulties in communicating or transmitting their knowledge and need to correct this lack of ability with the use of various additional tools [9]. This research proposes that the person who transmits knowledge also has an important role in the formulation and structure of the topics to be transmitted, creating the content map, giving weight and time to each item on the map, assigning evaluations to analyze what is transmitted and what is learned.

Therefore, the contribution of this work proposes the mapping of the topics, sub-jects or any data item defined during the structuration of the knowledge to transfer to Scrum phases to carry out the knowledge transfer mimicking a software project by establishing roles, ceremonies, and deliverables among others. In this way, a software engineering methodology (Scrum) can be applied to a higher level of abstraction (me-ta) to teach software engineering concepts in a structured and incremental way. The applicability of the proposal contemplates several scenarios so that it is possible to benefit from the proposed methodology. Additionally, the result of the knowledge transfer process in the industry will allow an improvement of the products generated, since, by updating their knowledge, the efficiency of the software developers will be favored with the creation of new and better products, and the industry will obtain a direct economic benefit since it will be the software developers themselves who carry out the process and not the need to hire third parties [29]. In this sense, the structuring of the transfer process and of the knowledge itself, promoted by the Scrum-based proposal, will be fundamental.

This research proposes to take advantage of the benefits of Scrum method, providing a higher level of abstraction and a framework to integrate with other methodologies such as those that a teacher can normally carry (inverted class, problem-solving, etc.). In addition, it promotes a collaborative culture among software developers, which improves the work environment and facilitates the recruitment and adaptation of new talent. There are some topics that are not included in this approach such as learning methodologies or teachers' skills

In this work we explore the use of the best practices and guidelines of the SCRUM agile software development methodology and to benefit the transfer of new knowledge among software developers. A study case about a lecture content in academia is used to illustrate how the content can be adjusted to teaching activities using Scrum approach. Concretely, an experiment in the Object-Oriented Programming language lecture of the National University of La Plata, Argentina has been conducted. The experiment outcome shows that the Scrum application has a positive impact on the learning process. In addition, a case

study industry is described to illustrate the approach application in this other context. The Public Company UTAE in Ecuador, through the department of information technology has provided the examples of real-life projects to use as input for our approach and provide a planning to structure the knowledge leveling of new team members.
The structure of the rest of the article is the following: In Section 2, we present some background of this work. Then, in Section 3, the main contribution is presented. In Section 4 we conducted an evaluation in academy to assess the advantages of our approach. In Section 5 we present a preliminary plan for an industry knowledge transfer case. Finally, we present the conclusion and further work in Section 6.

2. Background

2.1. *Knowledge transfer*

Knowledge transfer commonly seeks to disseminate knowledge, experience, and skills to facilitate the use, application, and exploitation of knowledge and capabilities [10][11]. In this sense, organizations that can effectively transfer knowledge from one unit to another are more productive and more likely to succeed over time than those that are less skilled in knowledge transfer [10].

Although organizations are being able to achieve significant increases in performance through knowledge transfer, they still find it difficult to achieve complete and effective knowledge transfer. Among the main problems are staff turnover, lack of use of agile techniques to capture and transfer knowledge in the software industry, and lack of incentives for team or group work in educational contexts [12].

Formal knowledge transfer occurs when there are written or verbal instruction manuals or procedures that are passed from one person to another. This could hap-pen through formal training or workshops. In comparison, informal knowledge trans-fer is when there are no written manuals or procedures that are passed from one per-son to another. This could be through verbal instructions, methodologies, emails, or even presentations [12].

Knowledge transfer is essential in the academy but it also plays an important role in the management process within an organization. It can facilitate a structured communication process where employees share experiences and information so that they can learn from it and build upon it. Having a knowledge transfer process gives employees access to relevant information to make better decisions and generate new ideas. This creates a space for innovation and encourages employees to work together and openly discuss ways to improve [13][29].

2.2. *Scrum*

Scrum is a widely well-known agile methodology based on the idea of creating short cycles for development [3], so-called sprints. During the Sprint, the team agrees to undertake a set of tasks that will be delivered at the end of the Sprint. To detect roadblocks or risks on time, the team has daily sync meetings to follow up on those tasks, which in turn becomes one of the fundamental elements of this methodology [13].

In terms of Scrum elements, we have the Product Backlog, which is a key part of the Scrum framework, as it consists of an ordered and prioritized list of the customer's needs or requirements, necessary for the execution of the project. As a second element, we find the Sprint Backlog, where the team defines a list of tasks to be per-formed in a Sprint, which they believe can be completed and demonstrated to the client at the end of the iteration. And finally, we have the increment as a part of the product generated in a Sprint, which has been added or developed and is considered finished and fully operational, being ready to be delivered or tested by the client. In [22], the traditional stages of Scrum are detailed, which in turn address several specific activities and describe the flow of a Scrum project.

2.3. *Related Works: Using Scrum for Knowledge Transmission*

To work on this section the researchers have proceeded to use the fundamental aspects of systematic review, defining in the first instance a research question "Can Scrum be used to transmit knowledge?". The Scopus library has been used to execute the following search string: "(knowledge AND transfer AND Scrum) OR (Scrum AND teaching)", providing a result of 323 research work.

As a first filter, we considered papers published between 2013 and 2023, giving us 323 works as a result. When reviewing the titles and abstracts of the articles, to filter on the topic we need, the researchers selected 49 articles for review.

Once the final list of articles is analyzed, it can be highlighted that authors such as [14][15], already identify that there is a lack of studies that directly involve agile methodologies with knowledge transfer, given that the work done has been focused on the management of software projects.

Additionally, Tenório et al [4] identify that Scrum and its various activities have not been exploited to find ways to share knowledge as a tool of transmission, but that the related studies have been targeted to the attempt to transfer knowledge but man-aging software projects, i.e. once the project has finished, they tried to abstract the knowledge that has been used in it, so these authors claim for future research to analyze how to use agile methodologies as a tool for knowledge transmission.

With this objective, in the study of [16], they have already begun to identify four options for attempting to transfer tacit knowledge with Scrum: a) tutorials and training, b) providing more trained personnel to teams of developers, c) participation of developers in theoretical exams resulting from software projects, d) participation in retrospectives of several projects.

Finally, it should be noted that all the studies analyzed on the use of Scrum to try to transfer knowledge have been in the field of education, but several of them [9][16][17][18][19][19][20][21][30] identify as an important opportunity to better map the use of agile methodologies and put it into practice, leaving aside the theoretical.

These studies focus on the conceptual explanation of Scrum, experiments focused on improving task delivery times, and use of Scrum in a domain other than software development, but lack of role definition, lack of contrasting results, and mapping of Scrum phases and activities to relate them to the new knowledge transfer process. In this work,

we introduce a complete mapping from learning content to SCRUM arti-facts so that knowledge transfer can take place using the agile approach. In addition, we have reported preliminary results of the approach application in a university lecture and compared its performance with a group using the traditional methodology

3. Proposal of using Scrum for knowledge transfer

3.1. *Process*

The adaptation of Scrum proposed in this work to transfer knowledge defines a framework that allows structuring the knowledge transfer process in a more organized and effective way. It is therefore a higher level of abstraction, which is placed on the teaching methodology, pedagogical strategy or didactic model that the expert wants to use when explaining each topic or concept. The proposed framework guides the knowledge transfer process, through a series of phases, meetings, roles, etc. to ensure that knowledge transfer takes place incrementally. Scrum offers an effective structuring mechanism and enables active dynamics, where the participation and motivation of the learners is supported by the benefits of this agile methodology. This is especially critical in contexts where the acquisition of this new knowledge is vital for the company, it is a knowledge inherited from other projects, or a particularly broad or complicated knowledge, also when you want to have a quick partial knowledge and increase it gradually.
Figure 1 presents a brief graphic description of how to carry out the proposed process for transferring new knowledge between software developers, using the stages of the Scrum methodology described above as a guide.

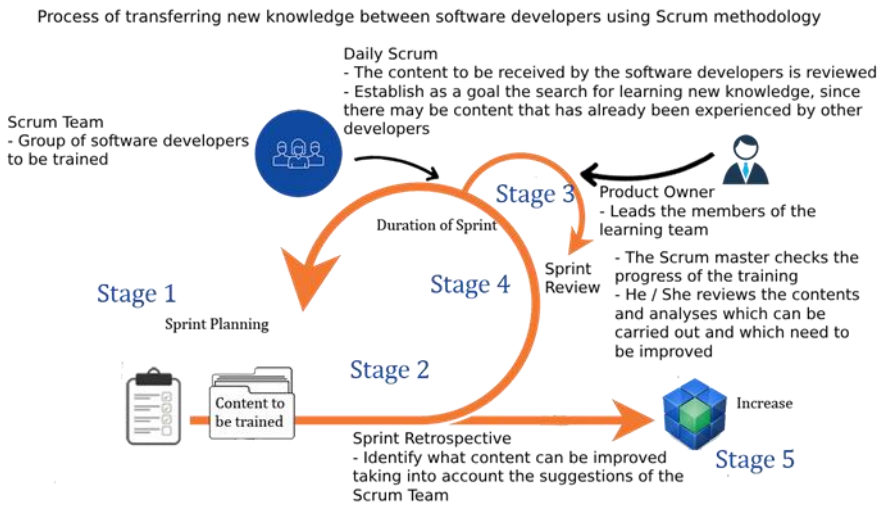

Fig. 1. Process of transferring new knowledge between software developers.

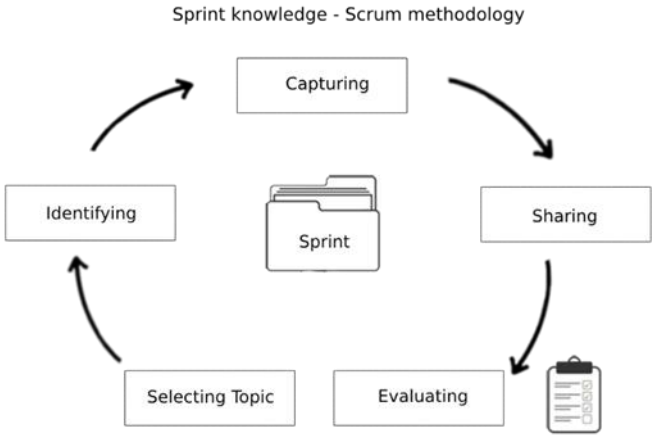

Fig. 2. Sprint knowledge - Scrum

Based on Figure 1, the Scrum process is composed of activities, sprints, and the definition of a list of requirements (Backlog) to be developed in each Sprint.

To start, each topic, concept, and task is incorporated into the Sprint backlog as an assignment or activity that team members need to address. The sprint backlog will be later prioritized and a subset of its items will be selected for a Sprint scope.

In Figure 2, we describe the process to be carried out in each Sprint for knowledge transfer, taking as a reference the model proposed by [28] to identify, capture, share, and evaluate each of the backlog topics. In the Identifying phase, each topic is bro-ken down into concepts or subtopics, if necessary. In the Capturing phase, each team reviews the topic and analyzes it. In the Sharing phase, if a team member requires feedback, the team collaborates to fulfill the process. Finally, in the Evaluating phase, the product owner requests to carry out a project to measure the level of assimilation of the topic.

Once the Sprint is over, we analyze whether the topic has been understood and start preparing the activities for the next Sprint. This is one of the reasons why this methodology makes so much sense to use in the software sector [10] because it helps software development teams evolve and advance with improvements for the final project. The same is extrapolated to the use of the Scrum methodology to transmit knowledge since knowledge acquisition is an incremental process. Knowledge ac-quired little by little, integrated little by little, is more persistent in the long run. That is another advantage of this proposal.

3.2. *Contexts*

There are many different scenarios where the proposed approach can be applied. In Table 1, we show three contexts that we consider to be the main ones. The case of students collaborating within an academy project has not been included, because these works usually have a small scope in terms of the volume of knowledge that needs to be

transferred, and in such a case it would not be necessary to apply the proposed framework (it could be sufficient to directly apply some collaborative methodology).

Table 1. Examples application contexts

Knowledge transfer	Context examples	Product Owner	Scrum Master	Daily meeting	Sprint meeting
New skill from external resource	Academia lecture	Teacher	Teacher	Q&A to Teacher	Progress assessment
	New technology training	Director/Manager	External Professional	Q&A	Progress assessment
Intra-team leveling	Running project training	Director/Manager	Team Leader/Manager	Q&A inter developers	Progress assessment

Firstly, the approach can be applied in academia where a teacher is the knowledge owner to be transferred like a programming language or technology. In this case teacher plays the product owner and scrum master roles because he is following up on the students' learning progress and leading the meetings respectively. During the daily meeting he/she is the main point of contact addressing all the students' concerns that could block them in their tasks. The teacher also reviews the sprint deliverables.

In the industry, a director or Manager could find beneficial for the business to count with a team trained on a technology, tool, or methodology by the team. He/she plays the Product owner role measuring the team deliverables quality during the spring meetings. As the team needs to gain a new knowledge, the knowledge owner (Scrum Master) is external from the team. Either in academia, and this case of industry, the topics, subjects, and items of information that a person should know at the end of the process must be part of the backlog and timely be packed in the sprint as tasks, challenges, or exercises. This plan must be designed by the knowledge owner to apply a problem-based learning. Problem-based learning allows those who use it to acquire relevant and meaningful knowledge, learn to work collaboratively, which is one of the premises of Scrum, and if necessary, it can be combined with other learning methods. It improves learning through continuous feedback from experts and strengthens ethical behavior since software developers feel more involved with the organization's projects [29].

Finally, new people could join a team because of previous drops, in this case there could be knowledgeable or expert people in the team who can support new team members. In this case, the team leader can set as goal the knowledge leveling so that any team member could develop any new requirement or maintenance task. The backlog will be composed of heterogeneous tasks (e.g., frontend, backed, or DevOps tasks) representing subjects, topics, and items that should be knowledgeable by any team member. The Scrum Master could be the leader or the Manager.

In all cases, daily meetings are an open space where blocking issues are discussed and debated by all so that the blocked developers (e.g., missing information, unable to resolve a problem, lack of details, etc.) can get assistance form experts but the remaining team members can learn from the experience in this cooperative context. The team members estimate the backlog tasks during the Sprint planning. All of this will be explained in more detail in the following sections

3.3. *Roles*

In this section, we introduce how the roles proposed by Scrum are put into practice in the specific case of knowledge transmission (Table 2). This is done from a dual perspective, that of industry and that of academia, to integrate both in our proposal.

Table 2**.** Identification of roles in Scrum and knowledge transfer processes.

Role	Definition in Scrum	Role description for transferring new knowledge
Product Owner	According to [16], the product owner should be the person responsible for maximizing the processes and identifying and prioritizing product features to obtain the best result for the business and the lowest cost.	For the case study in the software industry, we suggest that the Product Owner be the manager or director of the software development area of the organization, who in turn should have a clear vision of what needs to be taught or conveyed. He should take into account the risks and the level of feasibility of carrying out the training. It needs to have prior knowledge of what is being developed in the industry and what is intended to be conveyed to serve as a guide in the process [15]. One of his / her most important tasks is to guide the elaboration of the user stories and to look for the constant commitment of the software developers to be trained to fulfill the process of transference of new knowledge. In an educational context, the Product Owner will be the teacher.
Software Developers	According to the systematic review of [23], the ideal number of team members should be 7 people, with a margin of tolerance of plus or minus 2 people.	The software developers of this group are the ones who need to increase their knowledge in the use of some tool or methodology. In an educational context, they would be the students; in an industrial context, they would be the new developers included in the team.
Scrum Master	Guiding us in the study of [16], where it is stated that the Scrum Master must be the person who helps and guides the group until the business value is	We can suggest that, for the task of knowledge transmission, the Scrum Master should be a person with previous knowledge of the tool to be trained. It is recommended that he should be a senior

achieved, that is, the objectives set by the organization.	developer since the Scrum Master, being in permanent contact with the team, require that it is a person who first has the necessary knowledge to address any concern and, secondly, he has experienced the difficulties that software developers might encounter on their way to acquire new knowledge. This will help to make the work environment more conducive and productive for all software developers. In the software industry, the Scrum Master will be covered by a developer expert in the methodology and terms of knowledge. If there is no one available in the company to play this role, an external expert could join the team. While in the academic context, according to [17][20], the Scrum Master can be covered by a professor who helps the team in the problems encountered or if the case may be a member of the team with solid knowledge.

Once the roles have been defined, the planning and estimation phase arrives. The Scrum Master and the Product Owner proceed to define the topics and the team as-signs their respective estimation, and then the Scrum Master begins to delegate the corresponding tasks to each software developer. In contexts where the team members do not know the knowledge to be transferred, the knowledge owner (for example a professor or any external professional) will have to get involved, not being able to delegate completely. In addition, the content planning must be reviewed to identify prerequisites that could block the activity of the knowledge transferor. In the case of very extensive contents, would be necessary to analyze if it is feasible to split the subject matter or if the estimated time will increase in that case.

To approve, to estimate, and to confirm user stories. In this process, the Product Owner approves the user stories for a knowledge Sprint. The product owner will pro-vide a roadmap in terms of skills, abilities or topics that must be gained but will not focus on the how (I.e., technical details). In contexts where the product owner is not an expert in the knowledge being transmitted, he/she may rely on the knowledge owner (Scrum Master) or rely on what is to be learned and not how. The Scrum Master and the software development team will estimate the effort required to learn everything defined in the user story. The team of developers commits to analyze, process, and acquire new knowledge, so that we finally get the user stories, approved, estimated, and committed.

3.4. *Activities*

In this section, we describe each one of the scrum activities linked to the process of knowledge transmission.

Activity 1. Identification of objectives: The backlog is defined, which in turn will become the learning map, i.e., it will contain a structured and weighted list of con-tents, assigning a weight of importance to each content, always depending on the content of the training course. For example, if the contents to be transmitted are about object-oriented programming, class creation, inheritance, encapsulation, etc.
Activity 2. Review of objectives: In the second activity, an estimate should be made to fine-tune the list of pending objectives, since it is crucial to calculate the learning effort it will take. For this, we will rely on the "Expert Judgment", which will consist of the review of a certain objective by an expert developer of the organization where it will be identified how complex is to acquire certain knowledge.
Activity 3. Sprint planning: The third activity consists of planning sprints, and defining an appropriate time, according to the objectives. The duration of the Sprint is planned between 1 or 2 weeks and takes into account the software development team's backlog to date.
Activity 4. Generating the scrum board: The fourth activity consists of generating a scrum board with three columns: Pending, In Process, and Done, the team of developers will update it according to the needs. Our study, focusing on the software industry, suggests that Kanban boards be generated with the help of a digital tool, for ex-ample, Trello is free and available for both mobile phones and computers.
Activity 5. Daily Scrum meeting: During the meeting, students or developers will seek to answer: ¿What did you learn yesterday? ¿What we will learn tomorrow? and ¿what topics are the most complex?
Activity 6. Sprint demonstration: During this meeting, developer team demoes what they have learned during the sprints. The Product Owner with the software developer will define if the meeting can be face-to-face or remote, everything will depend on the time availability.
Activity 7. Sprint retrospective: Finally, in the seventh activity, we will run a Sprint retrospective, it will be a brief meeting where the development team suggests that you can improve the content to learn and give suggestions on how to improve the learning in the next Sprint.

3.5. *Ceremonies*

Within the Scrum ceremonies or meetings, we have the 4 main ceremonies that we describe in the following paragraphs.
Planning Ceremony. This is the first meeting to be held at the beginning of each Sprint. During this ceremony, it is recommended that the Product Owner, the Scrum Master, and the team of inexperienced software developers who have been selected for the process of transmitting new knowledge participate.
The main goal of these ceremonies is that the Product Owner (supported by the Scrum Master when dealing with an external teacher) can communicate the prioritized user stories, included in the Product Backlog with the team of inexperienced developers who are going to receive the training. In this way, the inexperienced soft-ware developers understand the scope of the topics of the course and with mutual agreement negotiate which one can be

developed in the Sprint that is planned. At this point, it is recommended to consider the time the developers have been designated to participate in the knowledge transfer process, as it is important not to overload the team with activities if we consider that they could be part of a different ongoing project. For this purpose, in industrial contexts, for example, pending deadlines and scheduled meetings should be considered. While in educational contexts, the schedule of exams and deadlines for classwork should be considered.

Once the scope of the knowledge Sprint has been defined, the team of software developers will divide each user story into tasks (with the help of the expert on the subject), which will be necessary to structure the planned table of contents, which in turn will be transformed into the structured learning map. Each content should be followed in a hierarchical order to avoid gaps in the transmission of knowledge.

Tasks will have an estimation of effort (Planning Poker could be used) where each learning increment has to be designed in such a way that it is feasible and adjustable to an estimated time, in the educational environment that is the role covered by the teacher because he plans how much he/she can teach in each class, in the industry environment Scrum Master must review the spring feasibility by taking into account the capabilities and previous knowledge of each software developer. The assignments will be show on the whiteboard once the software developer has engaged with the subject matter that needs to be learned. For our study, the expert developer and project leader (Scrum Master) will communicate the prioritized learning contents to the team of software developers, who analyzes each item of the course and plan the duration and activities of the Sprint. Each task will be enhanced with any documentation, example of a similar problem with its solution from internal or public projects based on the expert's suggestions. The new team member will use this material for a flipped class approach where any doubt will be addressed during daily meetings.

In academic context, the teacher will specify and plan the user stories relying on his experience on previous lectures. With the help of some board or digital tool (e.g. Trello), each task that has been identified as important will be placed on the board.

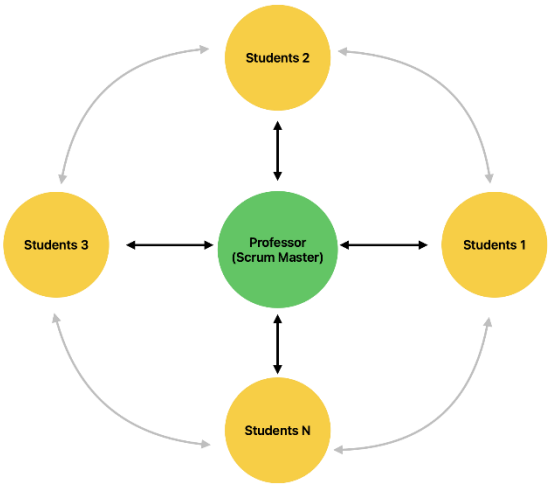

Fig. 3.a Academic star schema for communication.

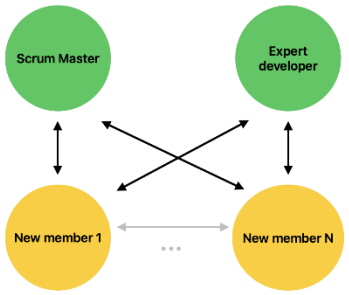

Fig. 3.b Balanced communication schema in industry

In Figure 3, a we show the communication scheme in Academia, where the main ac-tor is the professor who plays a key role in the knowledge transfer process. During daily meetings, a collaborative atmosphere should be promoted to allow not only student-professor integration, but also student-to-student learning experience sharing. On the other hand, Fig. 3.b, shows a more balanced experience between experts and new team members in a team. In the same way as in an academic context, experience sharing is encouraged, but there could be more than one technical reference such as expert developers, team leader (Scrum Master), etc.

Daily Meeting Ceremony. These meetings are estimated to last between 5 to 15 minutes. However, with a large group like a university lecture, it could take longer. In these meetings, the Scrum Master will be able to catch up on what each member of the team of inexperienced developers has managed to learn, but above all to know the difficulties that have arisen in learning or transmitting knowledge, analyze them, and help solve them to continue with the learning process in a normal way.

Revision Ceremony. During the revision ceremony in Scrum, the team of software developers who are involved in the learning process will present to the Product Owner what is being learned and how the process of acquiring the new knowledge has been. Inexperienced developers will explain with a practical example how they will use the knowledge gained in daily work activities. The Product Owner can suggest improvements to the assimilated learning and approve or not the process. This review ceremony will take place on the last day of the Sprint, with sufficient time for both the Product Owner and the team of inexperienced developers.

Retrospective. It will be based on the search for perfection. In this last ceremony of the Sprint, a retrospective should be carried out together with the Scrum Master and the Product Owner to make an analysis of what has been done to transmit the knowledge and the results obtained and to decide and document which concrete measures can be used to better transmit knowledge among software developers, that is to say, to learn from the successes and to improve everything feasible. As for the learning map, we will assess which contents were the most useful, and if this is the case, those that did not contribute to restructuring them to improve them

3.6. *Project closing*

The Owner Product, Scrum Master, and the now-trained software developers will analyze the contents, define the difficulties encountered in the process of transmitting new knowledge to learn from the most conflicting aspects, and improve them to speed up the process of new learning.

A feedback session will be planned with the Scrum Master and the software developers to clarify any doubts or queries that have been left pending in the training, so that these doubts do not become a limitation when applying new knowledge at work.

The Scrum Master will follow up on the development of the trained software developer in the first projects assigned, to evaluate the effectiveness or not of the implementation of the process of transmission of new knowledge. The result will be reported to the Owner product so that it can adjust the teaching process if necessary.

3.7. *How is the trainee evaluated after finishing the course?*

To evaluate the level of learning of a given software developer in the trained knowledge, we propose to use the Kirkpatrick method, which aims to measure the impact on training programs [24][32]. This method was selected because it will pro-vide us with an evaluation of the advantages that software developers obtained in training, and additionally focuses on giving an insight into the benefits that the organization obtains by using a well-structured measurement system. However, the knowledge transfer methodology proposed in this article could be integrated with other different evaluation mechanisms.

Using the Kirkpatrick method, three stages are performed. In the first stage, the applicability of the contents learned, and the mechanisms used to transmit the knowledge shall be assessed. With this purpose, a satisfaction questionnaire will be developed at the end of the knowledge transfer process. With this, we seek to have a first idea of whether what has been transmitted can be sufficient or not to apply this new knowledge to solve with success future problems.

In the second stage of the evaluation of acquired knowledge, a technical evaluation will be taken from the trained software developers. This questionnaire should be posed with questions focused on technical issues addressed in the training so that the software developer remains familiar with the terms and elements of the acquired knowledge.

In the third stage of practical application, a real case will be developed using the software tool, programming language, methodology, etc. that was the subject of the training.

Therefore, at the end of the training Sprint, the deliverable must also be in the process of development. This will allow the real impact of the contents or topics trained to the software developer to be felt.
Finally, the fourth stage of the evaluation is the measurement of results, for which we will base ourselves on the evaluations or on the solution of a case study, which will allow us to analyze which of the topics learned were better grasped or assimilated by the software developers.
To illustrate our approach, let's consider a common scenario in a software development company or in an academic context that must train its development team or students in new technologies: the case of real knowledge transfer in object-oriented programming (case study 1) and an example of the leveling of team members in a company (case study 2).

4. Case Study 1: Transmission of knowledge about OOP in Academy

Object-oriented programming is based on the concept of creating a model of the target problem in your programs, where objects encapsulating data and methods collaborate to achieve an objective [25]. In addition, object-oriented programming reduces errors and promotes code reuse [26][27][31][33]. The following sections summarize the object-oriented programming knowledge transfer experience that has been carried out.

4.1. *Objectives and Roles*

The objective of the training is to provide conceptual tools and resources for developers to be able to develop basic applications and perform corrective maintenance. The profile of the subject who will receive the training is a software developer with minimum project experience to understand the concept of deliverables and the im-portance of dates.
We consider it important for the software developer to maintain it with partial tutoring or feedback from the expert developer, to polish certain details of knowledge that could be left in the air. In addition, it is essential to carry out a collection of in-formation about possible errors or problems in the transference process, learn new knowledge, and carry out a Sprint retrospective to reuse all the observations and suggestions to improve the transference method.
To define the roles for the case study, it is very important to take into account the characteristics set out in the previous section. We proceed to designate the following roles: Product Owner (person responsible for the IT area of the organization, or for the academic case, a chairperson will be designated), Scrum Master (an expert developer will be selected in the case of industry, and in the academic field will be the professor), team of developers (group of software developers who require training, in the academic case the team will be formed by a group of students who receive the subject),

4.2. *Ceremonies*

Planning Ceremony. In the planning ceremony, a subset of activities is defined based on the product backlog following the learning map. Among the main user stories defined in

the learning map, we have object-oriented programming, definition of requirements to use Java, inheritance, polymorphism and others. For example, we have a user story called "Introduction to object-oriented programming" since it was considered necessary to first evaluate the concepts and definitions of object-oriented programming. In Table 3 we describe how the tasks of this first story would be structured.

Table 3. Task list for user story "Introduction to object-oriented programming"

Id Task	Task	N. hours
1	Introduction to the object-oriented programming paradigm	1
2	Pillars of object-oriented programming	2

Meeting Ceremony. The duration of the meeting ceremony or (Daily Scrum) will be established in advance. Although the Scrum daily meeting is set to 10 minutes, in this context daily meetings will depend on the number of students. For example, in our experience each participant requires about 2 minutes to update his status. In any case, this meeting should not to interfere with other lectures or any ongoing-project the developer team is assigned to. The team members will point out to the project leader (Scrum Master), the object-oriented programming learning content they have managed to learn and what topics they consider more complex. For example, in the case of the user story "Introduction to object-oriented programming", they will briefly analyze how the tasks were carried out and the estimated time used to proceed with the learning. In addition to defining whether any of the tasks have been left incomplete or need to be rescheduled.

Revision Ceremony. During the review activity, the team presents to the Product Owner the tasks progress according to the topic under review. For example, the user story "Inheritance" is reviewed on the last Friday of the Sprint (the Product Owner will coordinate the day when this ceremony is executed); the status of the tasks defined to the developers is analyzed and the Product Backlog is updated. If no task is completed, it is understood that the topic was not learned and therefore no progress has been made.

Retrospective Ceremony. The Scrum Master and the Product Owner jointly analyze how the knowledge transmission process has been developed in the case study and verify the level of learning acquired through the execution of practical exercises to the developers trained in object-oriented programming. In the academic case, questions and survey exercises are asked of the students to measure their level of learning. Therefore, at the end of the process of new knowledge transfer, the objective of the study will focus on analyzing the topics or training content, which serve as a guide to object-oriented programming by a software developer, and how it increases their capabilities to develop this type of applications.

4.3. *Evaluation*

Based on the previous, we ran an experiment to transfer knowledge of Object Oriented Programming inside the Universidad Nacional de La Plata, in Argentina. We have

conducted this evaluation to assess the benefits of using Scrum as a knowledge transfer framework. Based on the Basili template, the experiment goal is: to analyze the application of the Scrum approach in teaching to improve the learning experience of novice software developers.

Consequently, this paper addresses the following research questions:

RQ: Does the use of Scrum as a teaching framework improve subject passing rate? This research question aims to analyze whether or not novice software developers (students) who were exposed to the framework have a better pass rate; that is, whether this Scrum-based methodology to transfer knowledge is more effective than the traditional methodology of the master class.

To answer the research questions, we designed a completely randomized experiment in which subjects were asked to participate in the class in one of two options:

- Regular conference. The contents were delivered in the same way as the last 5 years (2015-2020) where every two weeks the expert developers presented the theoretical/practical work biweekly introducing the topics for the following two weeks and answering questions related to the task. At the end of the master class, the novice developer must pass the exams in one of the three attempts.
- Scrum class. As described above, novice developers attended biweekly meetings in which the expert developers reviewed with the novices the theory and practice for the next two weeks. Daily, there was a 30-minute meeting in which doubts and questions related to the task were discussed. Finally, a Sprint task was requested as a deliverable to track the performance of the novices.

The subjects were 48 novice software developers divided into two groups of 23 and 25 novices for the face-to-face class and the practical Scrum class respectively. Although both groups started with the same number of software developers, there was a decline in the hands-on class.

To evaluate the approaches, we decided to use the pass rate which is defined as the number of novice software developers who passed the exams over the number of novices assigned to the approach. The evaluation process was the same for novice software developers framed in this approach and those who were not framed to avoid any bias.

Therefore, in this experiment, the factor is the approach which is the provoked variation controlled by the researchers to measure its consequences. The possible values of the factor are the normal class and the scrum class; also known as treatments. In this case, the experiment was a Complete Randomized Design. Subjects were randomly divided into two groups for the two alternatives of the approach (the experiment factor): regular lecture stories (Control) and Scrum feature (Treatment).

The meetings and master classes were online due to the context of the COVID-19 pandemic, and we used the Moodle e-learning platform to host the material, ex-change messages, and receive deliverables from the students. It is necessary to mentioned that the meetings held with the use of the proposed methodology differ from the meetings held in a normal classroom, starting with the fact that they are systematic, short meetings, aimed at delivering milestones and even focused on providing suggestions to improve the process

of knowledge transfer, while the meetings of a classroom in the educational environment are longer and where the teacher leads them in his way of working, which makes them more rigid.

Both groups of new software developers used the same material for the theoretical and practical classes. During the design of the experiment, we decided not to add a new material-related factor. During the theoretical classes, the expert developer (teacher) used presentations and provided references to books and technical documents. On the other hand, in the practical classes, a practical document was provided with a list of tasks that the new software developer had to carry out.

During the class, the professors collected data on the activities of novice software developers and the results of the tests that were used for their analysis.

Table 4 shows the results of the normal and Scrum classes. 11 novices out of 23 passed the normal class, which means a pass rate of 47.83%. On the other hand, 14 out of 25 newcomers who took the Scrum class passed the course, representing 56%.

Table 4. Experiment result

	# New software developer	# Approved	Percentage
Regular Lecture	23	11	47,83%
Scrum Lecture	25	14	56,00%

Novice software developers taking the Scrum lecture attended daily meetings where different topics related to the lecture were reviewed. To analyze whether or not those meetings conditioned the approval rate we conducted a Pearson correlation test. The test provides a coefficient (r) that represents the relationship between the assistance to the meetings and the test approval. There is an agreement on possible coefficient r values Pearson correlation coefficient (r) value: greater than .5 is strong, between .3 and .5, and between 0 and .3 is weak. The Pearson correlation result points out that there is a significant large positive relationship between the daily meetings and the exam result with a strong relationship ($r(23) = .56$, $p = .004$).

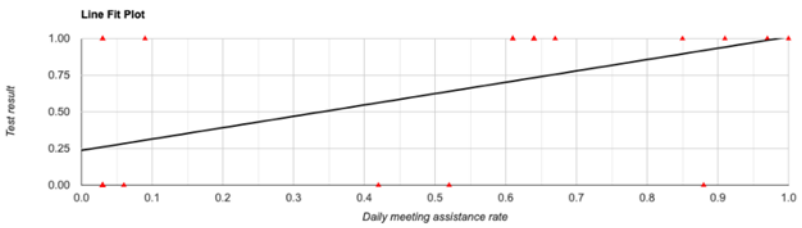

Fig. 4. Daily meeting assistance rate

The use of Scrum approach in lectures showed to improve the approval rate and we found evidence that daily meetings (30 min duration) affect the approval rate as shown in Fig. 4. This represents a 17% improvement in the approval rate. Unfortunately, there are not enough samples to perform a non-parametric test.

It should be noted that when using Kirkpatrick for the evaluation, each of the four levels has been passed through. At the first level, the training action was not required as all students share a knowledge base acquired from previous lectures. At the second level, the transmission of knowledge itself was evaluated through exams in both: control and treatment groups. At the third level, during daily meetings students were asked about any task road blocker that could affect their performance. At the fourth level, it was possible to analyze that the percentage of software developers who worked under the agile approach performed better than those who worked with a regular approach, as evidenced by the number of passes and fails.

5. Case Study 2: New staff leveling in Industry

New team members may join a development team because of many reasons. The dropout for new challenges could motivate some developers to seek new positions or the need to increase the development capacity to accelerate feature development in a project. To illustrate the approach, we connected with a development department of the Public Company PCUTA in Ecuador to get some examples of real user stories to be able to utilize our proposal in Industry. PCUTA is a company that maintains and develops university systems to support lectures enrollments. The department staff consists of 1 director, 3 project managers and 19 developers who are responsible for developing the tools used in at university.

In order to structure how the new staff members are brought up to speed in a team, we will plan (in collaboration with the company) a user story to structure the topics, subjects or any data that needs to be acquired. The planning aim at leveling new staff members in the technology stack. Since this is preliminary the sprints could be run as part of a warm-up period for training where there is no product in development.

The main products developed in PCUTA are Web solutions based on PHP Sym-phony technology and Microsoft SQL server. Therefore, we will classify tasks on the following stack tiers: Frontend, Backend, and Database. As there is a product already developed (or under-development), the user stores will correspond to tasks, new features or maintenance request. Each new member should take one assignment of each type to dive deep into the solution architecture, understand the coding style, testing approaches and its coverage, etc.

As introduced in the previous section, the approach aims at structuring the knowledge transfer. In this case, we are moving in the Intra-team leveling context (file 3 in Table 1). Therefore, the Product Owner is the director of the company, and the Scrum Master is the project manager of the running project. Often the requirements in Sprint 0 are focused on requirements gathering tasks and architecture de-sign. So, it makes sense to have new members review the assets from that sprint. If necessary, the new member can consider these tasks as pending and complete them. Afterwards, the results can be compared with

the results of the original task to stimulate discussion. Based on this, tasks of the project's Sprint 0 were structured as follows:

- To design Entity-Relationship Model or Object-Oriented Model.
- To review the Model.
- To create the database Script based on the model.
- To QA the database setup.
- To generate/create initial Symfony app.
- To code Object classes using design model.
- To generate/code Repository, Controllers and API controllers.
- To setup authentication/authorization mechanism.

In order to structure the content, each of the tasks should be accompanied by ap-propriate documentation/lessons to introduce the novel developer to the new knowledge. In this point is where the knowledge transfer occurs. For example, for the task "To design Entity-Relationship Model or Object-Oriented Model", knowledge about the UML standard and any previous design from the team must be transferred. Another case is the task "Configure Symfony Security System", where links to a GIT repository pointing to a previous system developed by the team using the same auth approach should be provided, as well as Symphony authorization documentation/lessons, and Auth0 documentation examples.

In Table 3, we present the Backlog Listing User Stories (US) definitions that com-bine front-end and back-end tasks. These user stories are part of a Sprint X. For the sake of simplicity, we have excluded from the list two tasks that need to be addressed for each US in this Sprint X: i) To document the new feature in the User Manual and Runbooks, and ii) To document the QA plan for the feature.

Each user story should be completely undertaken by a new developer so that he gets a full understanding of the software architecture avoiding backend/frontend specialty. During daily meetings the new staff members will share their doubts with the consolidated team members who are familiar with the business and technology stack. The daily meeting is a space where the developers can work through the challenges. Because the meeting time is booked on everyone's calendar, it is expected that roadblocks will be resolved with some assistance.

Table 4 shows how the development requirements have associated knowledge transfer elements that enable the new team member to face this development. As indicated above, knowledge transfer is based on expert knowledge of team members and other materials, and is supported by daily meetings.

Table 4. User stories

ID	DESCRIPTION	TASK ID	TASK	TIER	SUPPORTING DOC.
US1	As Administrator, I need to enter and/or modify the information of teachers and students	1	Create controller to create the profile of the teacher/student	Backend	- Symfony's API doc - GIT link to similar solution. - Best- practices on credentials storage and management.
		2	Create view to render teacher/student registration form	Frontend	- Angular forms docs - GIT link to similar solution.
		3	Validate form fields	Frontend	- Angular's validator docs
		4	Create a controller to edit student/teacher registration information	Backend	- Symfony's API doc - GIT link to similar solution.
		5	Create view to see the form to edit the teacher/student record	Frontend	- Idem Task ID #2
US2	As Administrator, I need to see the list of registered teachers/students	6	Create controllers to list the record of active teachers/students, inactive students/teachers and view the details of each one	Backend	- Symfony's API doc - GIT link to similar solution
		7	Create views to display active and inactive teacher/student record	Frontend	- GIT link to similar solution for tables and actions rendering.
		8	Create view to view detailed teacher/student information	Frontend	- GIT link to similar solution.
US3	As Administrator, I need to delete the record of a teacher/student	9	Create controller to delete teacher/student record	Backend	- Link to best practices on logical deletion

US4	As a user, I need to access the application using an authentication form	10	Create the landing page	Frontend	- Angular pages and routing links
		11	Configure Symfony Security System	Backend	- Symphony security documentation - Auh0 documentation for secondary auth strategy - GIT link to similar solution.
		12	Create login page	Frontend	- Idem Task ID #2
		13	Manage access control through routes – Frontend	Frontend	- Angular's security documentation for authorization and resource restriction.
		14	Manage access control through routes – Backend	Backend	- Angular's documentation for restricting content.
US5	As a registered user, I need to reset the users password	15	Create a handler to reset the user's password	Backend	- Email gateway documentation for notifications.
US6	As Administrator, I need to enter and/or modify the lectures information	16	Create a controller to register lecture	Backend	- Idem Task ID #4
		17	Create view containing a form to register lecture	Frontend	- Idem Task ID #4
		18	Create controller to edit lecture information	Backend	- Idem Task ID #4
		19	Create view containing form to edit lecture record	Frontend	- Idem Task ID #4
US7	As administrator, I need to see the list of tutorials	20	Query lectures to list the ones registered by teacher	Frontend	- Documentation using the Query API for Symphony
US8	As Administrator, I need to delete the registration of a tutorial	21	Create handler to delete tutoring record	Backend	- Documentation using the Query API for Symphony

The way we frame the knowledge transfer tasks has the following benefits:

- Methodology and reproducibility: The use of a methodology allows to guide document activities using templates, profit from management best practice and measure method activities. In ad-hoc warmup processes, the new mem-ber is often informally

introduced to the company practices depending on meetings to get some questions answered where the knowledgeable person is not clear how important is his role in the new member training.

- Team engagement: In this way, camaraderie and trust are built through mandatory daily meetings. New team members quickly bond with the team by breaking down barriers of shyness and embarrassment.
- Monitoring: Daily meetings help on following up the members progress on their assignments using standard tools like issues tracker, wiki or any other sophisticated collaborative office tool.

The company found this way of structuring the transfer of knowledge, simple and useful, serving as a formal mechanism that reduces the risk that this late introduction of the new developer usually generates. However, this process of knowledge transfer is in progress and its usefulness cannot be completely evaluated until it is applied in several projects where a new programmer enters once the project is started and these projects under development are successfully closed. The evaluation is planned as further work, when the company close the affected projects and report the experience.

6. Conclusions

In this work, a proposal has been made to use the Scrum methodology in the trans-mission of knowledge. For software developers who are familiar with this agile methodology, its application according to our approach will facilitate the acquisition of new knowledge in a more agile and comprehensive way. Therefore, we consider that the Scrum agile methodology, by providing a constant delivery of results, will help in improving the process of structuring learning content among software developers and its subsequent application for knowledge transfer, because the content is formed with constant structural recommendations, provided by software developers in academic training contexts or to the industry.

In addition, the time management of the software development team will be im-proved since Scrum by providing us with early results, software developers can begin to put what they have learned into practice a little before they have finished reviewing all the new knowledge content.

Another interesting point when using Scrum to transfer new knowledge is content management, software developers, the Product Owner and the Scrum Master at the end of each iteration review what they have learned, therefore, if there is any content that needs a little more effort, the product owner with the Scrum Master will replan the content according to the needs.

On the other hand, with the continuous or daily communication of the team, all the software developers know how the learning level of the new knowledge of each one advances, so if there is some subject that a software developer already under-stands it can become support for the developers who require a little more feedback.

With the proposed approach, iterative and incremental learning is strengthened, since each content or topic must be covered in an iteration, the team of software developers is

committed to performing all the necessary tasks planned for the knowledge transfer. In this research, it was considered important to first define a prior assessment of the level of knowledge and, if necessary, adapt the contents, generate structured contents, the learning map, the meetings for the transmission of knowledge and define the objective of the subject. This initial adaptation phase can be critical for success. In addition, by properly using the proposed Scrum methodology and its practices, we set up an evaluation to verify what is transmitted and what is learned.

Finally, it can be expected that the motivation of the team of developers is in-creased, since they are involved in continuous learning, and the content of new learning is structured with observations of the software developers themselves so that the topics to be dealt with are of interest to them.

With this intention, this article proposes and details a specific methodology to ap-ply Scrum for knowledge transfer among software developers and describes two experiences carried out within the academic context and the industry context to analyze its usefulness.

7. Discussion

Scrum has become a methodology in which people can tackle complex adaptive problems while delivering products with the highest possible productivity. It has be-come a full-fledged framework that is applied to build different types of products. Based on the above, with our proposal, we intend to support the knowledge transfer in an agile way to expert and inexperienced software developers, since we consider that the use of the Scrum methodology presents several benefits that we will detail below.

The first benefit we seek to obtain is to implement training without breaking the work dynamics of a team used to agile projects. In this way, the training participant will not have to submit to a new dynamic imposed by the course. For example, traditional courses require attendance at some classes per week and a final evaluation presenting a very different work rhythm than a project. This advantage in a software company is especially significant. Although, of course, in an educational context, students also benefit from agility, as time is money for everyone.

The second benefit is to take advantage of Scrum's feature of flexibility and adaptation, for which replanning will be used at the beginning of each iteration. Assuming that changes are a natural part of the learning process, the Scrum Master will review and complete the contents according to the value they bring to the course, trying as much as possible to avoid major changes in the course of the training. Traditional courses have two or three instances of evaluation where the teacher can identify problems in the student group limiting the ability to rectify any undesirable situation. In addition, the team of software developers will benefit from the feedback since if there is any doubt about a topic, in the daily meetings they can ask for clarification or guidance to solve the problem.

Another advantage is the adaptability and transparency of Scrum, it creates an environment of continuous improvement among software developers through the characteristic of synchronization and daily adaptation of the methodology. That is, the contents improve in

each Sprint due to changes and refinement (according to the learning progress of the participants), creating an innovative and creative work environment.
Finally, the structure of the learning map becomes an important point within the process of knowledge transmission using Scrum, in view that it will guide through the whole process, the order and importance of contents to be treated. Of course, bearing in mind that to define the contents, the time designated to the software developers daily for the process of knowledge transmission must be taken into account. Thus, the formative learning map is a resource and at the same time a mechanism to im-prove the training process, which is another strong point of the proposal. Another advantage of the approach is that it is flexible enough to be applied with different teaching methodologies. Moreover, the content framing and the incremental approach supports the knowledge transfer experience for team member who are not used to teach (which is often the case with members of a team of developers in industry).

8. Future work

The present research has emphasized the generation of content structures and processes to create training courses oriented to software developers in the industry and academy, using the agile methodology Scrum. Therefore, it would be interesting and planned as future work to apply the proposed methodology to transfer new knowledge among software developers in large or medium software development companies, where the benefits of the methodology are expected to have the greatest impact.
In addition, in the software industry, the XP model is also used for software development. In this sense, it would also be interesting to analyze how this methodology could be used for the transfer of new learning content to software developers with the help of management and collaboration tools.
Currently, the researchers are working on three additional experiments to test the SCRUM and XP approach, preliminary very good results have been obtained and will be shown according to the progress.

Acknowledgments

This work has been supported by the PLEISAR-SOCIAL project (Spanish State Plan for Scientific, Technical and Innovation Research 2021-2023).

References

[1] A. M. Turcios-Esquivel, E. G. Avilés-Rabanales, and G. Sayeg-Sánchez, "Use of technology and Scrum as an agile methodology to favor the development of balanced teamwork enrichment skills in higher education subjects," in 2023 IEEE Global Engineering Education Conference (EDUCON), 2023, pp. 1–3.

[2] A. K. Mbiada, B. Isong, F. Lugayizi, and A. Abu-Mahfouz, "Towards integrated framework for efficient educational software development," in 2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA), 2023, pp. 53–60.

[3] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," arXiv preprint arXiv:1709.08439, 2017.
[4] B. Crawford and C. L. de La Barra, "eXtreme programming meets knowledge creation and creativity," in 13th International Business Information Management Association Conference, IBIMA 2009, 2009, pp. 929–943
[5] Y. Dubinsky and O. Hazzan, "eXtreme programming as a framework for student-project coaching in computer science capstone courses," in Proceedings 2003 Symposium on Security and Privacy, 2003, pp. 53–59.
[6] N. Tenório, D. Pinto, M. J. Silva, I. C. de Almeida, and F. Bortolozzi, "Knowledge management in the software industry: how Scrum activities support a knowledge management cycle," Navus-Revista de Gestão e Tecnologia, vol. 10, pp. 1–13, 2020.
[7] L. Argote, P. Ingram, J. M. Levine, and R. L. Moreland, "Knowledge transfer in organizations: Learning from the experience of others," Organ Behav Hum Decis Process, vol. 82, no. 1, pp. 1–8, 2000.
[8] H. Takeshita, S. Okuaki, M. Nakamura, and H. Yamaguchi, "Analysis of the teamwork formation process in manufacturing problem-based learning (PBL)," Japanese Journal of Educational Psychology, vol. 64, no. 3, pp. 423–436, 2016.
[9] J. Sutherland and K. Schwaber, "The scrum guide," The definitive guide to scrum: The rules of the game. Scrum. org, vol. 268, 2013.
[10] M. J. D'Souza and P. Rodrigues, "Extreme pedagogy: An agile teaching-learning methodology for engineering education," Indian J Sci Technol, vol. 8, no. 9, p. 828, 2015.
[11] M. Kropp and A. Meier, "Teaching agile software development at university level: Values, management, and craftsmanship," in 2013 26th International Conference on Software Engineering Education and Training (CSEE&T), 2013, pp. 179–188.
[12] W. Montalvo, F. Ibarra-Torres, M. V Garcia, and V. Barona-Pico, "Evaluation of WhatsApp to Promote Collaborative Learning in the Use of Software in University Professionals," in Applied Technologies, 2020, pp. 3–12.
[13] G. Lang, "Agile learning: Sprinting through the semester," Information Systems Education Journal, vol. 15, no. 3, p. 14, 2017.
[14] D. Paulin and K. Suneson, "Knowledge transfer, knowledge sharing and knowledge barriers--three blurry terms in KM," Leading Issues in Knowledge Management, vol. 2, no. 2, p. 73, 2015.
[15] D. Babik, "Teaching Tip: Scrum Boot Camp: Introducing Students to Agile System Development," Journal of Information Systems Education, vol. 33, no. 3, pp. 195–208, 2022.
[16] L. Argote and E. Fahrenkopf, "Knowledge transfer in organizations: The roles of members, tasks, tools, and networks," Organ Behav Hum Decis Process, vol. 136, pp. 146–159, 2016.
[17] A. Srivastava, S. Bhardwaj, and S. Saraswat, "SCRUM model for agile methodology," in 2017 International Conference on Computing, Communication and Automation (ICCCA), 2017, pp. 864–869.
[18] V. Potkonjak, K. Jovanović, O. Holland, and J. Uhomoibhi, "Distance learning and skill acquisition in engineering sciences: Present state and prospects," Multicultural Education and Technology Journal, 2013, doi: 10.1108/17504971311312627.
[19] P. A. P. Aragão and R. C. G. Souza, "Scrum XPerience: A New Approach for Agile Teaching," in Proceedings of the XXXVI Brazilian Symposium on Software Engineering, 2022, pp. 134–142.
[20] R. E. Landaeta, S. Viscardi, and A. Tolk, "Strategic management of scrum projects: An organizational learning perspective," in First International Technology Management Conference, 2011, pp. 651–656.
[21] C. Baham, "Implementing scrum wholesale in the classroom," Journal of Information Systems Education, vol. 30, no. 3, p. 1, 2019.

[22] S. Al-Ratrout, "Practical Implementation of Agile Approaches in Teaching process," International Journal of Engineering and Advanced Technology (IJEAT), vol. 8, no. 4, pp. 278–284, 2019.
[23] J. Vogelzang, W. F. Admiraal, and J. H. van Driel, "Scrum methodology as an effective scaffold to promote students' learning and motivation in context-based secondary chemistry education," EURASIA Journal of Mathematics, Science and Technology Education, vol. 15, no. 12, p. em1783, 2019.
[24] J. Metrôlho, F. Ribeiro, P. Graça, A. Mourato, D. Figueiredo, and H. Vilarinho, "Aligning software engineering teaching strategies and practices with industrial needs," Computation, vol. 10, no. 8, p. 129, 2022.
[25] D. F. Rico and H. H. Sayani, "Use of agile methods in software engineering education," in 2009 Agile Conference, 2009, pp. 174–179.
[26] G. Wedemann, "Scrum as a Method of Teaching Software Architecture," in Proceedings of the 3rd European Conference of Software Engineering Education, 2018, pp. 108–112.
[27] K. S. Rubin, Essential Scrum: A practical guide to the most popular Agile process. Addison-Wesley, 2012. .
[28] K. H. Pries and J. M. Quigley, Scrum project management. CRC press, 2010.
[29] S. Ashraf and S. Aftab, "Latest transformations in scrum: a state of the art review," International Journal of Modern Education and Computer Science, vol. 9, no. 7, p. 12, 2017.
[30] R. Bates, "A critical analysis of evaluation practice: the Kirkpatrick model and the principle of beneficence," Eval Program Plann, vol. 27, no. 3, pp. 341–347, 2004.
[31] Kanaki, K., & Kalogiannakis, M. (2018). Introducing fundamental object-oriented programming concepts in preschool education within the context of physical science courses. Education and Information Technologies, 23(6), 2673-2698.
[32] Lutz, M. (2010). Programming python: powerful object-oriented programming. " O'Reilly Media, Inc.".
[33] Tshuma, B., Steyn, H., & Van Waveren, C. (2018). The role played by PMOs in the transfer of knowledge between projects: A conceptual framework. South African Journal of Industrial Engineering, 29(2), 127–140.
[34] J. Saukkonen, "Effects of project-based learning in education-enterprise collaboration to learning experience and student engagement.," Finnish Business Review, 2014.