International Journal of
Computer & Software Engineering

Case Study Open Access

Diversity in Subject Content Transfer Styles Improves Software Engineers' Analysis Skills

Maria-Isabel Sanchez-Segura[1*], Fuensanta Medina-Dominguez[1], Ana Sanz[2] and Javier Saldaña[3]

[1]Department of Computes Science, Carlos III Technical University, Madrid, Spain
[2]Softtek, Madrid, Spain
[3]Ayming, Madrid, Spain

Abstract

The learning profile of each student must be taken into account to offer the learning resources that best suits him/her and thus can benefit him/her more. In software engineering field, the learning process can be improved if lecturers use multiple content transfer styles to teach the same concept. The authors provide a case study to demonstrate that diversity in content transfer styles oriented to teaching software analysis techniques allows lecturers to improve students' analysis capabilities. To validate the hypothesis, a case study was developed using a control group which learned software analysis techniques using the traditional method based on slides; for the second group the lecturers provided different subject content transfer styles in order to observe the improvement in the learning process when students have a choice of content transfer styles. Statistic techniques were used to validate the results. Due to the results obtained and analyzed, the authors can affirm that diversity in subject content transfer styles improves software engineers' analysis skills, reflected in the quality improvement of software products developed.

Publication History:

Received: June 25, 2016
Accepted: December 27, 2016
Published: December 29, 2016

Keywords:

Computer science education, Education, Educational technology, Engineering education, Improving classroom teaching

Introduction

Research into the way people learn is becoming more and more advanced. That is, research is showing that everybody has a learning profile which influences how they learn [1-6]. In addition, the advancement in information and communication technology (ICT) has changed the way teaching and learning are conducted, facilitated by the effective combination of different delivery modes, teaching methods and learning styles [7]. So, the combination of knowledge about how people learn and ICTs provide opportunities to create interactive student-centered learning environments, which are easily accessible and flexible [8]. Apart from technology, it is necessary to consider the learning profile of each student to offer the learning resources that best suits him/her and thus can benefit him/her more [9].

Considering that not everybody learns in the same way, it is advisable to use certain pedagogical elements according to each person's learning profile. Training course material should not only focus on one type of educational resource, typically slides and bibliographic texts, but several tools must be provided to meet individual learning styles [10]. Therefore, it is necessary to integrate ICTs and pedagogical elements to design and develop training material [11].

This work focuses on the improvement of software engineers' learning analysis process. For software engineers, analysis of the information system is one of the main processes that constitute the software development lifecycle. As a result of this process, a detailed software product requirement specification is developed [12]. The requirements specification must satisfy the user's needs and forms the basis of designing the system [13]. Failures introduced by software engineers during the analysis phase in the software under development will persist throughout the product lifecycle, increasing the cost and time of rework to correct them when they are detected later [14]. So, the better the system requirements specification skills software engineers have, the fewer the mistakes made during the system analysis phase.

The reason why the authors focused on the learning process of analysis techniques is because, as explained before, it is a critical process in software development and the loss of requirements, which causes a lack of functionalities in the final product, is still a common problem [14]. This loss of requirements can be clearly identified in the relationship between use cases and requirements specifications. When software engineers define use cases, they often forget that each requirement has to be included in at least one use case description or scenario, resulting in a loss of requirements [15]. In an incremental-iterative use-case-driven lifecycle, this is currently more widespread; the final system will lose the functionality that has not been included in use cases [16].

The main goal of this paper is to improve software engineers' capability to perform information system analysis tasks, more precisely, use cases and requirements specifications in order to decrease the percentage of volatilized requirements in software projects by developing a teaching-learning process that suits the profile of every software engineer; in other words, a process that takes into account the two key elements of learning mentioned previously: ICT and pedagogical factors. The degree of improvement in the teaching-learning process where each student can choose the format he/she is more comfortable with or feels that he/she can progress more quickly with will be evaluated.

This paper is organized as follows. Section 2 contextualizes this work describing existing learning theories and styles and several technology systems which support learning. Section 3 describes related works. Section 4 describes the method to improve the teaching-learning

[*]**Corresponding Author:** Dr. Maria-Isabel Sanchez-Segura, Department of Computes Science, Carlos III Technical University, Madrid, Spain, Tel: +34916249421; E-mail: misanche@inf.uc3m.es

Citation: Sanchez-Segura MI, Medina-Dominguez F, Sanz A, Saldaña J (2016) Diversity in Subject Content Transfer Styles Improves Software Engineers' Analysis Skills. Int J Comput Softw Eng 1: 110. doi: https://doi.org/10.15344/2456-4451/2016/110

process and the case study which was carried out to test whether the personnel's training had improved. Also, at the end of this section, the authors present and analyze the results. Finally, section 5 presents the conclusions of this work.

Context of this Work

Learning Theory: Learning theories help to understand and predict human behaviour from the point of view of how knowledge is acquired. Their goal is to define theoretical models that specify the human learning process. These theories, however, do not provide a solution; they only draw lecturers' attention to essential variables. Some learning theories are cognitive information processing [22], cooperative learning [23], objectivism [24], sociocultural Learning [25], constructivism [26] etc. The most recent one, constructivism [26], establishes that students have to build new knowledge based on previously acquired knowledge [18], instead of memorizing everything as proposed in other theories. Constructivism was selected for this work because it best fits the way software engineers think and learn. For the constructivist, knowledge is not the result of a copy of reality; it is an interactive dynamic process through which the individual interprets and reinterprets external information, progressively building more complex and powerful explanatory models [27].

Learning style: Learning style is defined as the way a person perceives, processes, integrates and remembers information; that is, behaviours that indicate how to learn [28]. Students have different levels of motivation, attitudes, and learning styles [29]. Knowledge of learning styles can be used to increase self-confidence and awareness of strengths and weaknesses [30].

Several learning styles have been defined to support students' needs. Table 1 summarizes the most widespread learning styles and their characteristics.

The authors selected the Felder-Silverman learning style model (SPAL Method) for this case study because it is considered the most appropriate for a software engineering training course [40]. It was designed to capture the most important learning style differences among engineering students, and provides a good basis for engineering instructors to formulate a teaching approach that addresses the learning needs of every student [38]. The model classifies students according to the following four dimensions: sensitive or intuitive, visual or verbal, active or reflective, and sequential or global [38]. This model has been already implemented in software engineering courses and it can be tested using different technologies.

Learning management system: Although Blackboard [41] is to date one of the best learning management systems, is it also true that existing open source tools have many of the desired features for a learning management system. Due to budget constraints, open source tools were used to carry out the case study presented in this work, which is more replicable for the reader. In the case of WebCT [41], there were important deficiencies in the user interface, making it complex and unintuitive. Atutor [42], Chamilo [43], Moodle [44] and Sakai [45] were also analyzed for this work. Moodle [44] was selected because it facilitated creating and modifying resources easily in comparison with Atutor [42] or Chamilo [43]. It also imports SCORM (Sharable Content Object Reference Model) contents.

Learning Style	Context
Canfield´s Learning Styles Inventory	
The Canfield Learning Styles Inventory (CLSI) is a 30-item assessment using a 4 point rank order procedure for each item. Students rank these choices in order that best describes their preferences or reactions. A ranking process is used to obtain the raw scores. Thus, the lower the score, the stronger the preferences [31].	Generic Education
Dunn and Dunn Learning Style Inventory	
Learning strategies are included in the methods thorugh which teachers teach and/or learners learn. Methods and strategies which match the different types of learners are, for example, contract activities packages (CAP), program learning sequences (PLS) and multi-sensory instructional packages (MIP) [32].	Generic Education
Kolb´s Experiential Learning Model	
Kolb classified students into four types: having a preference for concrete experience or abstract conceptualization (how they take information in), and active experimentation or effective observation (how they process information) [33]. Preferences on this scale are assessed with the Learning style inventory [34] or the Learning type measurement [35].	Generic Education
Honey and Mumford´s learning style model	
Based on Kolb´s theory, this model identifies four types of students: reflector, theorist, pragmatist and activist [36]. It was developed in an attempt to apply learning style theory to business and management studies measurement [37]	Business and Management studies
Felder-Silverman Learning Style Model (SPAL method)	
It was designed to capture the most important learning style differences among engineering students, and provide a good basis for engineering instructors to formulate a teaching approach that addresses the learning needs of every student [38]. The model classifies students according to the following four dimensions: sensitive or intuitive, visual or verbal, active or reflective, and sequential or global [38].	Engineering students
Keefe´s Learning Style Profile	
The Learning Style Profile (LSP) was designed to give teachers an easy way to determine the learning styles of middle and senior high school students. LSP diagnoses students´ cognitive styles, perceptual response tendencies, and study instructional preferences by means of 23 variables [39]	Middle and senior high school students

Table 1: Characteristics of learning styles.

Related Works

The importance of considering different learning methods has been widely studied [46]. Murrel and Claxton [47] cited several studies in this area that form the basis for all subsequent research on learning styles. Dewey [48] pointed out that students learn best if you include a component of experience in the learning process. Similarly, Lewin [49], found that an active learning environment plays an important role.

Several studies confirm the relationship between learning styles and academic achievement as a result of students' response to different teaching methods [46]. Different researchers have found evidence to show that current information using different approaches lead to more effective instruction [50]. The overview of studies on academic performance and learning styles is very extensive: analysis of academic performance, in general; relation to learning styles; relation among learning styles; teaching strategies, methods and academic performance, etc. Most of the works establish a relationship between the influence of learning styles and student performance [50-57]. After reviewing different studies Alonso et al. [Alonso, 99] concluded that it seems well established that students learn more effectively when learning styles are adapted to their needs. According to Figueroa [58], in engineering classrooms, students are extremely practical, oriented toward facts and procedures and prefer visual presentation of material, so the lecturer should empower these dimensions. However, as Alonso [55] pointed out, there is great difficulty in adapting teaching to learning styles. Not only must students' learning styles be taken into account but also the teaching style of lecturers.

This is why the authors of this paper have defined and validated a method to adapt a learning style, with a set of specific content transfer styles and integrated it into lecturers' way of teaching.

Adapting Learning Material To Student Needs: A Case Study

The goal of this case study is to apply a method that takes into account both pedagogical and technological perspectives to make knowledge transfer more efficient. As mentioned in the previous section, the SPAL [38] learning style was selected to develop this case study. SPAL defines the sequence of activities that should be carried out by students and lecturers to achieve course objectives. This model was used to create learning resources to transfer subject contents, called "content transfer styles", to represent different learning resources being used: slides, electronic process guidance (EPG), wikis, videos, and interactive exercises.

These content transfer styles were selected for two reasons:

- they are mentioned in the work of Caper Jones [56] as one of the major channels for software personnel (people involved in the software development process) to acquire new information, evaluated in terms of cost, learning efficiency, learning effectiveness and currency information.
- they comply with the features of SPAL, which requires that the content transfer styles must be intuitive, reflective, sequential, verbal and visual (Table 2).

Content transfer styles designed and used in this case study	Content transfer style description	Content transfer style example	SPAL method student traits covered
Slides	Slides with information about the topic used by the lecturer in class.	http://seldata.sel.inf.uc3m.es/docs/mdv/CasosDeUso.pdf	Intuitive Reflective Sequential Verbal Visual
EPG	Electronic Process Guidance that describes the software development process to be followed by students.	http://dhip.sel.inf.uc3m.es/PFC-guiaprocesos/WS_AnalisisOrientadoObjetos.htm	Active Intuitive Reflective Sensitive Sequential Verbal
Videos	For each class there was a video which shows the previous class covering the same topic.	http://www.youtube.com/watch?v=OTxi3aMXTVA&list=UUD-BQ8n7IsuihmHbIZK6T6Q&index=10&feature=plcp&noredirect=1	Global Intuitive Reflective Sensitive Sequential Visual
Wiki (product patterns)	A wiki with the description of every product pattern that could be used by students.	http://productpatternswiki.sel.inf.uc3m.es/mediawiki/index.php/Metodo_de_Craig_Larman	Active Global Intuitive Reflective Sensitive Visual
Interactive Exercises	Exercises which allow students to test their knowledge.	http://seldata.sel.inf.uc3m.es/docs/mdv/CasosDeUso.htm	Active Global Intuitive Reflective Sequential Verbal Visual

Table 2: Content transfer styles designed and used in this case study.

The description of student traits in the SPAL learning style appears next:

1. Active: learns by trying things out, enjoys working in groups.
2. Global: holistic thinking process, learns in large leaps.
3. Intuitive: abstract thinker, innovative, oriented toward theories and underlying meanings.
4. Reflective: learns by thinking things through, prefers working alone or with a single familiar partner.
5. Sensitive: concrete thinker, practical, oriented toward facts and procedures.
6. Sequential: linear thinking process, learns in small incremental steps
7. Verbal: prefers written and spoken explanations
8. Visual: prefers visual representations of presented material, such as pictures, diagrams and flow charts.

Definition of the Case Study

The case study was carried out in a software engineering training course in the last year of a large state university in Madrid, Spain. This course is addressed to improving software engineers' abilities to perform software engineering tasks, specifically those developed in the analysis and design phases. The authors focused on the analysis phase because the goal of this case study is to improve the training of future software engineers in the system analysis phase, more precisely, in the use of techniques; use cases and requirements specifications. The achievement of this goal will be demonstrated by a decrease in the percentage of volatilized requirements in software projects and improvement in the quality of the products obtained by the participants. The use cases technique was selected because it is a very widespread technique, common to all development paradigms. Use cases is a key element and if there is a breakdown between system requirements and use cases, then the system will lose a part of its functionality, specifically the one related to the requirements lost when use cases were modelled. Improvement in the software engineers' training was carried out by means of different subject content transfer styles instead of restricting the learning process to just one style.

To develop the case study, two software projects were developed consecutively from September 2013 to February 2015 in order to observe the improvement in software engineers' analysis capability. Each software project developed corresponds to an iteration of the case study (two iterations in total). Table 3 summarizes the main features of each iteration.

The authors considered the analysis phase in the two software projects in Table 3, with similar features and complexity, performing the same set of software product development activities. The same lecturers trained the participants in the two projects. The way in which the set of activities proposed by the authors was developed on each iteration is summarized in Table 7, section 4.2.

It is important to highlight that, as this is a real case study, there were different students on each iteration, but similar sample sizes were maintained and the students on each iteration had the same profile, knowledge level and capabilities. Before starting the subject the lecturers developed a test to determine the students' knowledge of a set of concepts studied in previous years. The distribution of the grades in the samples is always the same, so on both iterations there is a similar distribution of student with grades between 5 to 7, 7.1 to 9, and 9.1 to 10. . This is how the lecturers maintained the variable related to the students' capabilities. Also, although the students were not the same on each iteration, two different practicals were prepared, one for each iteration in order to prevent students from sharing their practicals across iterations. Each practical corresponded to a software project developed on each iteration.

The training phase, where students learn the subject content, was different on each iteration. Therefore, in the first iteration the training was performed in the traditional way (on-site classes, class materials such as PowerPoint slides), making this the control group in this case study. In the second iteration, the content transfer styles used were slides, EPG, videos, wikis and interactive exercises. Each student chose the means that best suited his/her learning characteristics. Although this paper focuses on requirements specification and use cases techniques, it is important to point out that complete software projects were performed and they lasted 6 months on average. Of this time, one month was devoted to defining requirements specifications and use cases; for this purpose two one-and-a-half-hour training sessions were carried out weekly.

The parameters studied, being directly related to the previously stated objectives, improvement of the students' performance of use cases and requirements analysis task were:

- Volatility of requirements: percentage of coverage of requirements regarding use cases. Traceability matrix between requirements and use cases are analyzed in order to identify the percentage of requirements that do not trace with any use case.
- Number of reviews performed to achieve the required quality level: number of reviews necessary to perform the software requirement specification until the required quality level is achieved.

	Subject content transfer styles used	Number of students involved	Project developed
Iteration 1 September 2013 to February 2013	Slides	36 (12 groups, 3 students each)	Project 1: Development of an application that allows the management of the family account.
Iteration 2 September 2014 to February 2015	EPG Exercises Interactive Slides Videos Wiki (product patterns)	30 (10 groups, 3 students each)	Project 2: Development of an application that allows to manage bibliographical resources.

Table 3: Main features of each iteration.

As a secondary aim, the authors wanted feedback on the participants' satisfaction when they were able to choose the way they learnt requirements analysis.

- Participants' satisfaction: Degree of satisfaction of software engineers who took part in the project during the second iteration (2014-2015).

The following section presents a description of activities, defined in The Student Preference Adapted Learning Method, that the authors propose in this work. The activities describing the method followed can be applied to any area, but they have been illustrated for the specific case of software engineering students. First, the activities of the proposed method to be developed on each iteration of the case study are described in section 4.2. After, Table 7 shows the specific way in which each activity was carried out on each iteration.

Description of activities developed

Activity 1: Elaborate learning resources

The first activity consists of designing and elaborating learning resources to be used in the training course to improve the knowledge acquisition process.

Learning resources influence actively the process of student learning. These resources facilitate knowledge transfer, so they should be elaborated with special care and concern. It is important that they motivate students and not have the opposite effect.

In this activity, the material used to implement the course was created when none existed, or the existing material was updated. Table 4 summarizes the actions and results obtained from the execution of activity 1.

2013-2014 Iteration 1	Slides used in the training course were updated and reviewed. After every class they were published. Throughout this process, innovation and creativity features were considered in order to create attractive and friendly material for students to use. They were published after each class. An example of these slides can be accessed at: http://seldata.sel.inf.uc3m.es/docs/LearningExperiment/slides/Requisitos.pdf
2014-2015 Iteration 2	Slides used in the training course were updated and reviewed. They were published after each class. An Electronic Process Guide (EPG) that defined the main software development process as well as techniques to be applied was also available for students. The EPG can be accessed at: http://dhip.sel.inf.uc3m.es/PFC-guiaprocesos/ Videos related to main analysis techniques were provided. An example to illustrate the videos provided can be seen at: http://www.youtube.com/watch?v=w5IvHrf1njY A wiki with information related to analysis and design task was available to facilitate the software project activities. http://productpatterns.sel.inf.uc3m.es

Table 4: Summary of the results from Activity 1.

Activity 2: Select technology

In order to support the knowledge acquisition process it is recommended to use a technological platform that facilitates a collaborative and interactive learning environment. Each time this activity is performed the technology must be reviewed to check that it complies with the established requirements. Table 5 shows a summary of actions and results obtained from the execution of this activity in this case study.

2013-2014 Iteration 1	Moodle was selected as the platform to support the dissemination of the pedagogical material to students as well as a mechanism to support communication and collaboration among students and teacher.
2014-2015 Iteration 2	Technological needs and the coverage of the platform were reviewed. According to the results of this revision, Moodle was maintained as the technological platform. In addition, a wiki was developed; an open source tool called Dekiwiki was used to implement it.

Table 5: Summary of the results from Activity 2.

Activity 3: Establish course goals and schedule

The third activity consisted of establishing the course goals. Before the course began, the lecturer established the course goals, determined the learning resources to be used, planned scheduled contents and established the method to assess students. Table 6 shows the goals identified for this case study in which the method defined in this work was applied as well as an excerpt from the schedule where the kind of tasks and their sequences are shown. Goals are common for the three iterations and the example corresponds to iteration 1.

Goals (common to all iterations) - Acquire knowledge and suitable techniques for the analysis and design of a software product, using an object-oriented software methodology. - Understand the principles of a good software analysis and design. - Develop a software application using the knowledge acquired in the course.	
Excerpt of planning tasks:	
Date	Content
1st October 2013	Introduction OO (Oriented Object), UML (Unified Modelling Language), RUP (Rational Unified Process) (1 h) Requirements (2h)
8th October 2013	Requirements: practical work (3 h)
15th October 2013	Presentation of the case study (45 min) Performing the case study (1 h 15 min)

Table 6: Summary of the results from Activity 3 in iteration 1.

Activity 4: Select teaching-learning strategy

The strategy used for the classes is described below.

Step 4.1: The lecturer explains concepts that software engineers have to know and that integrate a learning unit. For this purpose, the lecturer uses slides prepared or updated previously.

Step 4.2: A practical exercise is done for each unit so that software engineers can learn how to apply the knowledge to solve problems and consolidate concepts learned in the master lecture. For this purpose, several exercises are set out and, subsequently, solved.

Step 4.3: In the session in which software engineers begin to develop the case study, the lecturer presents the project that they have to execute and teaches them how to use the learning resources.

Step 4.4: The software engineers develop a project assigned by the lecturers that covers the development of use cases and requirements specification techniques. This work focuses on analyzing the improvement obtained in the analysis phase. Although the students developed the entire project, this paper only considers products related to software requirements and use cases.

Activity 5: Assess degree of learning

The lecturer assesses the software engineers works developed, bearing in mind several quality criteria previously defined. The assessment is carried out in a continuous way, taking into account products obtained in each phase of the software development lifecycle contemplated in the deployment of the method proposed by the authors. In the context of the case study developed and reported in this paper, the products under consideration are requirements specification and use cases.

Table 7 shows the differences between activities developed in the iterations described in this work.

Analysis of results

The results obtained from the experiment were analyzed from three perspectives: volatility of requirements, reviews needed to achieve the quality level required and, as a collateral effect, the participants' level of satisfaction using the content transfer styles proposed.

Volatility of requirements

The percentage of requirements that are not related to any use case was calculated for every project, considering the average rate each team achieved. Therefore, the traceability matrix between requirements and use cases was performed for these, which allows to detect requirements that do not trace with use cases; these are lost requirements. An T-test and the representation of the interval plot were used to analyze data.

- Data Analysis of Iteration1 versus Iteration 2

H0: number of volatility requirements in both iterations is the same.
Iteration 1: number of volatility requirements in iteration 1.
Iteration 2: number of volatility requirements in iteration 2.

With a p-value of 0.009, lower than 0.05, H0 can be rejected. As Figure 1 shows, the number of volatility requirements was reduced when the content transfer styles proposed were used.

To conclude, it can be said that the use of the content transfer styles provided improved their analysis capability because the number of requirements that are lost when use cases were modeled decreased.

Reviews needed to achieve the level of quality required

The quality of requirements is a significant factor that affects the success of a project. A suitable software requirement specification will contribute to improving the resource and time planning, and the development of the system. Therefore, it is necessary to ensure that requirements are clear, precise, unambiguous and complete. By means of this variable, the number of reviews that were necessary to carry out until the software requirement specification fulfilled the level of quality required was analyzed. The quality criteria used to validate the software requirements specification are accessible at: http://seldata.sel.inf.uc3m.es/docs/LearningExperiment/criteria/Criteria.pdf. A T-test and the representation of the interval plot were used to analyze the data.

ACTIVITIES	Iteration 1: 2013-2014	Iteration 2: 2014-2015
Activity 1	The effort was addressed to develop and improve the slides used in the master classes. Product patterns and the EPG began to be developed.	Pedagogical material was reviewed and updated. Further, interactive exercises were created.
Activity 2	The technology selection process was carried out.	The technology was reviewed.
Activity 3	This activity is performed in the same way in each iteration.	
Activity 4 Step 4.1	This activity is performed in the same way in each iteration.	
Activity 4 Step 4.2	With the objective of reinforcing the knowledge students acquired, several optional exercises were proposed to be solved at home.	With the objective of reinforcing the knowledge acquired by students, several optional exercises were proposed to be solved at home. These exercises were interactive in order to increase students' motivation to perform them.
Activity 4 Step 4.3	This activity is perfomed in the same way in each iteration. The only difference is the explanation lectures gave for the learning resources to be used on each iteration.	
Activity 4 Step 4.4	Students developed a case study in groups, using slides and the recommended library resources.	Students developed a case study in groups. For this purpose all the resources were available: slides, EPG, product patterns, videos, texts and library resources.
Activity 5	The lecturer assesses students. For this task, the lecturer grades the work and the students' presentation.	The lecturer assesses students. For this task, he/she gives a mark to the works and carries out an individual exam in order to identify if students have achieved the learning goals previously defined. To prepare the exam, the resources used during the work (slides, library resources, the EPG, videos, product patterns, interactive exercises and library resources) were available.

Table 7: Summary of activities and how they have been developed on each iteration.

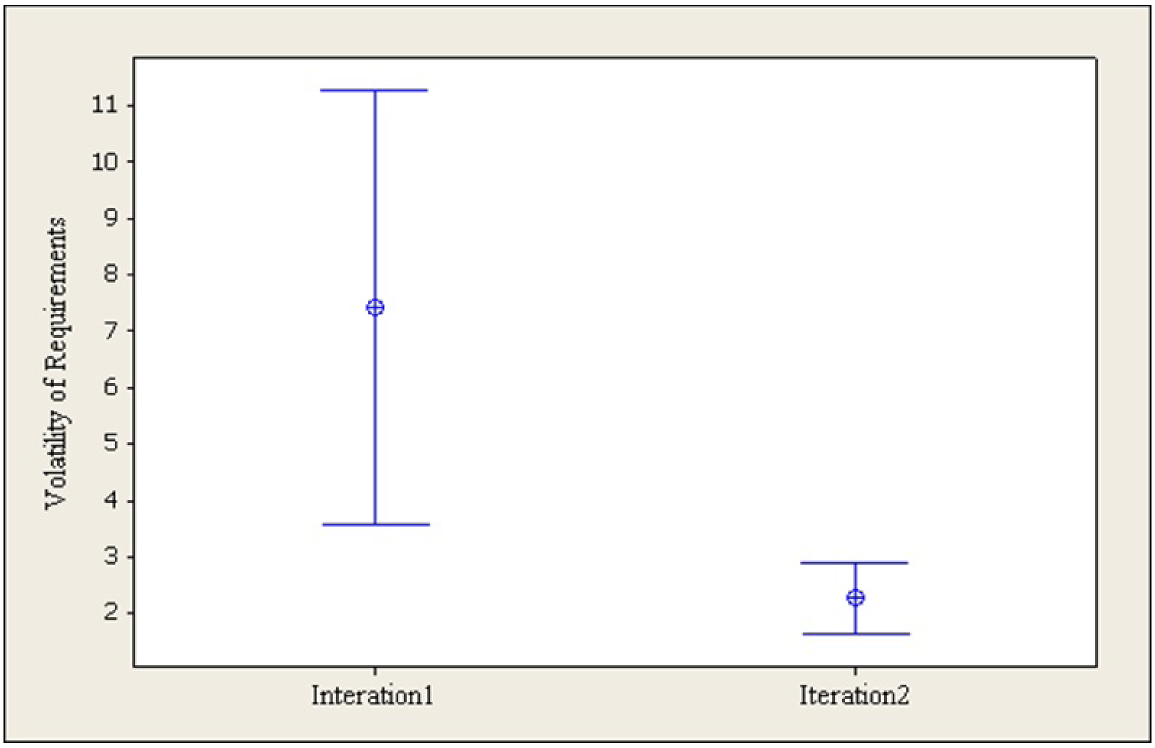

Figure 1: Interval plot: Volatility of requirements for Iterations 1 and 2.

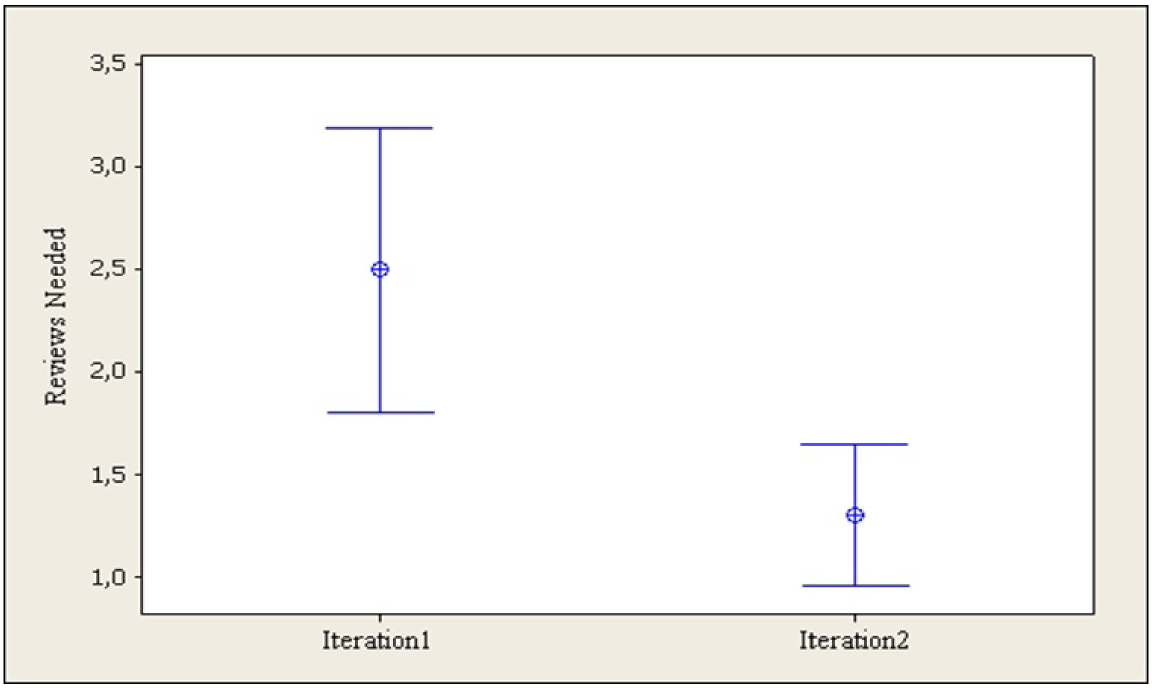

Figure 2: Interval plot: Reviews needed for Iterations 1 and 2.

- Data Analysis of Iteration1 versus Iteration 2

H0: number of reviews needed to achieve the level of quality required in both iterations is the same.

Iteration 1: number of reviews needed in iteration 1.

Iteration 2: number of reviews needed in iteration 2.

With a p-value of 0.004, lower than 0.05, H0 can be rejected. As can be seen in Figure 2, the number of reviews needed to achieve the quality required in systems requirements is lower in iteration 2 when the software engineers used the content transfer styles proposed.

To conclude, it can be said that the use of the proposed content transfer styles improves the quality of the system requirements because the number of reviews needed to achieve the quality required decreased.

In addition, the reduction in the number of reviews needed to reach the required quality of the system requirements using the proposed content transfer learning styles positively affects the verification and validation process because the time and effort and, in consequence, the cost of this process is reduced.

Participants' satisfaction

In the second iteration, a survey was carried out in order to assess participants' satisfaction as regards content transfer styles used in the course. This information was addressed to determine the usability of such resources.

Figure 3 shows that the differences between valuations obtained for resources are not significant. Therefore, no resource was valuated negatively or with a really low mark. This means that the software engineers considered that these tools helped them positively to improve their skills in software requirements and use cases. Moreover, the fact that all the resources got similar marks proves that there are several learning styles and therefore every individual requires the pedagogical resources that best suits his/her learning profile. Thus, the usability of each resource depends on the software engineer and his/her learning style, but there is no evidence that a resource is not usable or better than the rest. It depends on the software engineer's preferences.

The questionnaire used to analyze the participants' satisfaction (http://seldata.sel.inf.uc3m.es/docs/LearningExperiment/questionnaire/CuestionarioSatisfaccion.pdf) includes questions addressed to identify the usability of each content transfer style as well as control questions to identify ambiguous or conflicting responses by the same individual. The results obtained for each resource corroborate the data shown in Figure 3. None of these reveal a significant number of people who fully disagree with it, so every content transfer style is usable for an important group of people. Further, there is no resource which has a significant percentage of people that identify with it, but there is a variety in the choice of content transfer styles used.

Threats to Validity of the study

The major internal validity threats are listed here.

Related to the influence of teachers, in both iterations there were two teachers but they were different in each iteration. The teachers who taught in the iteration 2 did not participate in iteration 1. The only common person in both iterations was the coordinator of the subject, who did not teach class in any of the iterations and was not communicating with students. It is the reason which authors can affirm that there was no influence by teachers in the outcome of the data related to the common material in both iterations, the authors can ensure that the slides, which are the only common material in both iterations, were not improved nor updated; teachers used the same slides in both iterations.

Conclusions and Future Works

Requirements are one of the key elements to develop software systems in planned time and with required quality levels. These must be clear, precise, unambiguous and complete. However, the low quality of requirements and their loss are common problems. One way to solve these problems is to improve software engineers' qualification through the use of multiple content transfer styles. This work proposes to develop training materials which integrate both pedagogical and technical perspectives to improve software engineers' capability to perform information analysis tasks. The authors have developed a case study where the statistical analysis of its results reveal that the percentage of volatilized requirements decreased and their quality increased in the software analysis phase.

Due to the fact that each individual has a different learning style, the authors have developed a method which involves the use of several learning resources in the format of content transfer styles so that each software engineer can select the one which best suits his/her profile. To assess the improvement with this method, a case study was carried out and the results of the different iterations were analyzed.

As a conclusion of the results, the percentage of volatility requirements decreased as well as the number of reviews to achieve

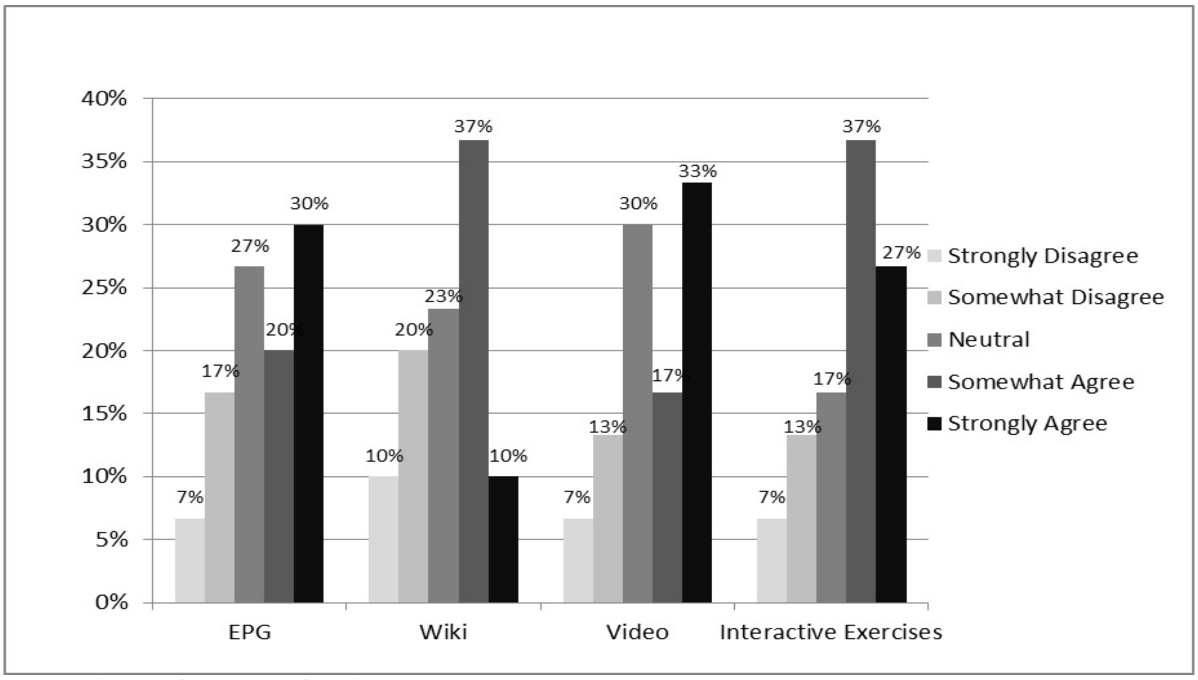

Figure 3: Valuation of content transfer styles.

the level of quality required, which means that the system analysis tasks were performed better, using the different content transfer styles proposed by the authors.

This means that not only has the requirement quality increased but also the verification and validation process has improved due to the time reduction.

It worth's the cost of producing the additional material involved in the development of the case study not only because the results of the case study are positive but also because they can be reused on following academic years.

The students valued content transfer styles according to their learning profile. Therefore, it can be concluded that all the content transfer styles have contributed to the software engineers' learning, one cannot be classified as better than the others because all of them have been useful, depending on the learning style of the student.

This work is focused on Use Cases and requirements specifications in order to improve analysis skills. However, it is possible to replicate this experiment in other development and management processes in the software engineering field and even in engineering areas in general.

Competing Interests

The authors declare that they have no competing interests.

References

1. Crick RD (2007) Learning How to Learn: The Dynamic Assessment of Learning Power. Curriculum Journal 18: 135-153.
2. Ching-Hong L ((2010) The comparison of learning effectiveness between traditional face-to-face learning and e-learning among goal-oriented users, 6th International Conference on Multimedia Technology and its Applications (IDC), 255-260.
3. Bi T (2011) Modern learning theory and u-learning studies, International Conference on e-Education, Entertainment and e-Management (ICEEE), 239-242.
4. Heywood J (2011)A historical overview of recent developments in the search for a philosophy of engineering education"; Frontiers in Education Conference (FIE), PEEE-1-1-PEEE-14.
5. Dihua, X (2012) Personalized Learning Path Recommender Based on User Profile Using Social"; Fifth International Symposium on Computational Intelligence and Design (ISCID), 511- 514.
6. Yarandi M (2012) An adaptive e-learning Decision support system"; 15th International Conference on Interactive Collaborative Learning 1-5.
7. Chia-Wen T, Pei-Di S, Meng-Chuan T (2011) Developing an appropriate design of blended learning with web-enabled self-regulated learning to enhance students' learning and thoughts regarding online learning"; Behaviour & Information Technology 30: 261-271.
8. Kennewell S, Tanner H, Jones S, Beauchamp G (2007) Analysing the use of interactive technology to implement interactive teaching"; Journal of Computer Assisted Learning 24: 61–73.
9. Felder R, Brent R (2011) Understanding student differences. Journal of Engineering Education 94: 57-72.
10. Brusilovsky P, Peyl C (2003) Adaptive and intelligent web-based educational systems, International Journal of Artificial Intelligence in Education, 13: 2-4, Special Issue on Adaptive and Intelligent Web-based Educational Systems, 159-172.
11. Koun-tem S, Yuan-cheng L, Chia-jui Y (2008) A study on learning effect among different learning styles in a Web-based lab of science for elementary school students. Computers & Education 50: 1411-1422.
12. Ahram T, Karwowski W, Amaba B (2011) Collaborative systems engineering and social-networking approach to design and modeling of smarter products. Behaviour & Information Technology 30: 13-16.
13. Ruparelia NB (2010) Software development lifecycle models, ACM SIGSOFT Software Engineering Notes 35: 8-13.
14. Hamill M, Goseva-Popstojanova K (2009) Common Trends in Software Fault and Failure Data"; IEEE Transactions on Software Engineering 35: 484-496.
15. Kulak D, Guiney E (2004) Use cases: requirements in context, Addison-Wesley Professional.
16. Larman C, Basili VR (2003) Iterative and incremental developments. A brief history. Computer 36: 47- 56.
17. Bormida GD, Ponta D, Donzellini G (1997) Methodologies and tools for learning digital electronics"; IEEE Transactions on Education 40: 294-294.
18. Salovaraa H (2005) Achievement goals and cognitive learning strategies in dynamic context of learning"; Acta Universitatis Ouluensis, Scientiae Rerum Socialum E78.
19. Vainionpää J (2006) Erilaiset oppijat ja oppimateriaalit verkko-opiskelussa"; (English translation: Different Kinds of Learners and Learning Materials in Web-based Studying) Acta Tamperensis, 1133, Tampereen yliopisto, Tampere.
20. Akdemir O, Koszalka T (2008) Investigating the relationships among instructional strategies and learning styles in online environments. Computers & Education 50: 1451-146.
21. Nijhuis J, Segers M, Gijselaers W (2008) The extent of variability in learning strategies and students' perceptions of the learning environment"; Learning and Instruction 18: 121-134.
22. Schuell TJ (1986) Cognitive conceptions of learning. Review of Educational Research 56: 411-436.
23. Slavin RE (1990) Cooperative learning: Theory, research, and practice, Englewood Cliffs, NJ: Prentice Hall.
24. Yarusso L (1992) Constructivism vs. Objectivism. Performance and Instruction Journal 31: 7-9.
25. O'Loughlin M (1992) Rethinking science education: Beyond Piagetian constructivism toward a sociocultural model of teaching and learning. Journal of Research in Science Teaching 29: 791-820.
26. Jonassen DH (1993) Thinking technology: Context is everything. Educational Technology 31: 35-37.
27. Gomez C, Coll C (1994) De qué hablamos cuando hablamos de constructivismo. Cuadernos de Pedagogía.
28. Matthews BD, Hamby VJ (1995) A comparison of the learning styles of high school and college/university students"; The Clearing House 68: 257-261.
29. Kettanurak VN, Ramamurthy K, Haseman WD (2001) User attitude as a mediator of learning performance improvement in an interactive multimedia environment: an empirical investigation of the degree of interactivity and learning styles"; International Journal of Human-Computer Studies, 54: 541-583.
30. Coffield F, Moseley D, Hall E, Ecclestone K (2004) Learning styles and pedagogy in post-16 learning, A systematic and critical review"; Learning and Skills Research Center (LSRC), London, England.
31. Canfield AA (1992) Canfield learning styles inventory manual, Western Psychological Services, Los Angeles, CA.
32. Dunn R, Dunn K (1999) The complete guide to the learning styles in service system, Allyn and Bacon.
33. Kolb DA (1984) Experiential learning: Experience as the source of learning and development"; Englewood Cliffs, NJ: Prentice Hall.
34. Atkinson G (1991) Kolb's learning style inventory: A practitioner's perspective"; Measurement and Evaluation in Counseling and Development 23: 149-161.
35. LTM - Learning Type Measurement (2004) Discover your learning styles graphically.
36. Zwanenberg NV, Wilkinson LJ, Anderson A (2000) Felder and Silverman's index of learning styles and Honey and Mumford's learning styles questionnaire: How do they compare and do they predict academic performance?"; Educational Psychology 20 : 365-38.
37. Honey P, Mumford A (1998) The manual of learning styles, Maidenhead, UK: Peter Honey.

38. Felde RM, Silverman LK (1998) Learning and teaching styles in engineering education. Engineering Education 78: 674-681.

39. Keefe JW (1986) Learning style profile; Reston, VA: National Association of Secondary School Principals.

40. Carver CA, Howard RA, Lane WD (1999) Addressing different learning styles through course hypermedia. IEEE Transactions on Education 42: 33-38.

41. Blackboard Inc (2008) Blackboard Platforms.

42. A tutor Group (2008) A tutor Learning Management Tools.

43. Chamilo Asssociation (2008) Chamilo e-learnig and collaborative software.

44. Moodle group (2008) Moodle.

45. Sakai community (2008) Sakai Platform.

46. Martinez E, Gallego A (2003) Estilos de aprendizaje y e-learning. Hacia un mayor rendimiento academico"; RED: Revista de Educación a Distancia 7: 1-10.

47. Murrel P, Claxton C (1987) Experiential Learning Theory as a Guide for Effective Teaching"; Counselor Educational and Supervision 27: 4-14.

48. Dewey J (1986) Experience and Education. The Educational Forum 50: 241-252.

49. Lewin K (1951) Field Theory in Social Sciences"; New York, Harper and Row Publishers.

50. Saarikoski L, Salojärvi S, Corso D, Ovcin E (2001) The 3DE: An Environment for the Development of Learner-Oriented Customised Educational Packages"; ITHET, Kumamoto, pp. FC2:21-FC2-26.

51. Newland JW, Woelf NN (1992) Learning style and academic performance within a group of sophomore Medical students. Academic Medicine 67: 349.

52. Sobral D (1995) Diagnostic ability of medical students in relation to their learning characteristics and pre-clinical background. Medical education 29: 278-282.

53. Davies SM, Rutledge CM, Daviesa TC (1997) The impact of student learning styles on interviewing skills and academic performance. Teaching and Learning in Medicine 9: 131-135.

54. Lynch T, Wowlfl N, Steele D, Hanssen D (1998) Learning Style influences student examination performance"; The American Journal of Surgery 176: 62-66.

55. Alonso CM, Gallego DJ, Honey P (1999) Los estilos de aprendizaje. Procedimientos de Diagnóstico y Mejora" (4rth ed.); Ediciones Mensajero, Bilbao, Spain.

56. Carper J (2009) Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies". MacGraw-Hill.

57. Duran EB, Costaguta RN (2008) Experiencia de Enseñanza Adaptada al Estilo de Aprendizaje de los Estudiantes en un Curso de Simulación. Formación Universitaria 1: 19-28.

58. Figueroa N, Zulma C, Méndez P, Rendón J, Guido F (2005) Los Estilos de Aprendizaje y el Desgranamiento Universitario en carreras de Informática"; Primeras Jornadas de Educación en Informática y TICS en Argentina (JEITICS). Buenos Aires, Argentina.