CLEI ELECTRONIC JOURNAL, VOLUME 17, NUMBER 3, PAPER 7, DECEMBER 2014

Design and Experimentation of Activities for CS1: A Competences Oriented Approach (unpacking the Informed Design Teaching and Learning Matrix)

Alejandro Adorjan
Universidad ORT Uruguay, Facultad de Ingeniería,
Montevideo, Uruguay, 11100
adorian@ort.edu.uy

and

Inés Friss de Kereki
Universidad ORT Uruguay, Facultad de Ingeniería,
Montevideo, Uruguay, 11100
kereki_i@ort.edu.uy

Abstract

In Introductory Computer Science courses, especially Computer Science 1 (CS1), dropout rates are generally high and results are often disappointing. In order to motivate and engage students to achieve better results in CS1, our teaching strategy is based on designing several activities using a competences oriented approach. This paper describes the use of a framework proposed by Crismond and Adams in order to create pedagogical activities for the CS1 course at Universidad ORT Uruguay. We propose to extend that framework with competences oriented activities. We present a detailed description of each activity. Our thesis is that including this kind of activities helps to obtain better results. Experimentation was done in 2012 and 2013. Teachers of the experimental group referred a high level of motivation of the students. Results show that the inclusion of those activities seems to be helpful for students and the proposed pedagogical design appears to produce better final results.

Keywords: Computer Science 1, Programming, Competence, Teaching, Learning.

1 Introduction

Programming is in the heart of computer science, and therefore most Computer Science (CS) programs globally start with an introductory programming course [1]. Programming is certainly a complicated skill to master, and learning to program is correspondingly complex [2,3] and is understandably a key area of education research [4].
In Introductory CS courses, especially CS1, dropout rates are generally high [1,3,5,6,7] and results are often disappointing [1,3]. Several strategies have been adopted by different institutes in the organization and teaching of these courses to diminish these effects [6].

Competences and their development have acquired a key role in many current teaching and training methods [8,9]. Competence means the proven ability to use knowledge, skills and personal, social and/or methodological abilities, in work or study situations and in professional and personal development [10]. Generic and specific competences are been proposed in Computing Engineering profiles through competency-based curricula models [11].
CS1 course at Universidad ORT Uruguay emphasizes teaching problem-solving methodology that uses an Object Oriented Programming approach.

The research question that motivated this study was: What kind of competence-oriented activities can we design to engage students and obtain better results?

We propose unpacking the Informed Design Teaching and Learning Matrix proposed by Crismond and Adams [12], assuming the role of informed design teachers that propose innovative activities (oriented to competences in our case), in order to satisfy learning goals of the course, develop several competences of our students and reduce drop out percentages.

This paper is structured as follows: Section 2 refers to competences and learning outcomes. Section 3 includes several teaching strategies at University level. Section 4 reports the Informed Design Teaching and Learning Matrix concepts. Section 5 deals with CS1 course. Section 6 describes our proposal, details the unpacking of the Matrix and explains each of the activities proposed. Section 7 reports the experimentation. Section 8 presents the conclusion and future work.

2 Competences

Competences and learning outcomes are emerging as a new teaching/learning paradigm where approaches centered on the learner are increasingly important, playing a key role in the teaching and learning process [13].

Competences represent a dynamic combination of knowledge, understanding, skills, and abilities [14]. Most of the taxonomies of competences are organized into general and specific. General competences have acquired a special relevance in the last years [15].

Transversal competences are usually forgotten and neglected, however competences in transversal skills are considered by employers thinking about hiring a university graduate as important as technical knowledge [16].

Higher Education must provide advanced knowledge, skills and competences that students need for their professional life [16].

The higher education sector is faced with several strategic decisions in order to maximize quality, impact, and competitiveness. In this context of "engineering" competences and learning outcomes, one core challenge is the inclusion of curriculum stakeholders in prioritizing subject-specific and generic competences in study programs [17].

Tuning Latin America Project (TLAP) [18] refers to 27 generic competences. From the list of generic competences agreed for Latin America we selected those which apply in the context of our CS1 course: capacity for abstraction, analysis, and synthesis (C1), ability to apply knowledge in practice (C2), ability to organize and plan time (C3), capacity for oral and written communication (C4), ability to use information and communication technology (C5), ability to learn and update learning (C6), ability to identify, pose, and solve problems (C7), ability to work as part of a team (C8), interpersonal skills (C9) and ethical commitment (C10). These competences are skills that software engineering graduates must possess [11].

3 Teaching Strategies

Good teaching is getting most students to use the level of cognitive process needed to achieve the intended outcomes that the more academic students use spontaneously [19]. Traditional teaching methods do not seem adequate for many students for different reasons [3]. Beyond the mastery of core CS material, good CS educators should also be familiar with a significant body of material that will expand their perspectives on the field, and consequently, enhance the quality of their teaching [20].

Several teaching and learning concepts have been proposed in engineering education [21]. Effective teaching requires flexibility, creativity, and responsibility in order to provide an instructional environment able to respond to the learner's individual needs, and one of the ongoing challenges the university teachers are facing is related to matching the teaching strategies with the students' learning styles in order to improve the academic results [22].

Students have different levels of motivation, different attitudes about teaching and learning, and different responses to specific classroom environments and instructional practices, the more thoroughly instructors understand the differences, the better chance they have of meeting the diverse learning needs of all their students [23]. Activities that require students to collaborate, share solutions, review each others' work, or create materials have been shown to be beneficial for the students [24].

A survey of literature related to teaching of introductory programming reported by [25] concludes that there is no canonical answer to the question on how to teach introductory programming courses. Several techniques are identified by [26] in this courses: questionnaires, interviews, observations, videos, inventories, tasks, artifacts, tests and formal course assessments.

In this context, our teaching strategy is based on designing several competences oriented activities that motivate and engage students in order to achieve better results.

4 The Informed Design Teaching Matrix

Crismond and Adams [12] report the Informed Design Teaching and Learning Matrix (IDTLM) for engineering education that describes design strategies, contrasting patterns tiles, statements of how beginning and informed designers do those strategies, relevant goals and instructional approaches that teachers can use.

Design Strategies presented by Crismond and Adams [12] describe nine design strategies: Understanding the Design Challenge (DS1), Building Knowledge (DS2), Generating Ideas (DS3), Representing Ideas for Deep Inquiry (DS4), Weighing Options & Making Decisions (DS5), Conducting Test and Experiments (DS6), Troubleshooting (DS7), Revising /Iterating (DS8) and Reflecting on Process (DS9).

Crismond and Adams [12] argues that IDTLM acts as a framework for teaching and learning, including teaching strategies that instructors need to know to teach engineering effectively. Design activities are the opportunity to naturally weave together skills, processes, and knowledge that are typically taught separately in the discrete subjects of traditional curricula [27].

The following examples of teaching strategies (TS) are presented by Crismond and Adams [12] supporting design strategies:

- TS1: comprehending the problem statement, problem framing, and scoping.
- TS2: focus information searches, study prior art, writing a product history report, research users, product dissections and reverse engineering.
- TS3: divergent thinking, brainstorming, constraint relaxation, generative database searches, starter vs. final project challenges.
- TS4: thinking with given model, building before sketching, virtual drawing and computational modeling, descriptions and structures reviews of design ideas, artifacts, and gestures as stand-ins for drawings.
- TS5: explanation-based designing, decision diagrams, design values and guidelines, emotions and their role in design decision-makings.
- TS6: experiment-base design advice, investigate-and-redesign task and product comparisons.
- TS7: diagnostic troubleshooting, cognitive training in troubleshooting, troubleshooting stations.
- TS8: design storyboards, project and time management, instruction and scaffolding for systematic design, risk taking and iteration.
- TS9: design diaries and portfolios, compare and contrast design cases, computer-supported structures reflections.

All the above teaching strategies could be used in teaching activities and support the design strategies presented in the IDTLM. Nine design strategies (DS1- DS9) are presented in the IDTLM, also contrasting patterns tiles (Informed Design Patterns) and several instructional approaches (TS1- TS9) that teachers can use.

Design knowledge and skills is a core-learning objective in combination with reinforcing fundamental engineering competences [28]. Informed design is a pedagogical approach to design, and as a pedagogical strategy, design activities have great potential to: engage students, encourage pluralism thinking, reflect upon, revise and extend internal models [29].

Crismond and Adams [12] focus on teaching and learning design, arguing that IDTLM contains design strategies that help teachers do informed teaching, helping teachers design tasks while developing their own design pedagogical content knowledge.

Crismond [29] argues that The "Informed Design Teaching and Learning Matrix" provides more in-depth descriptions of the design practices, research on misconceptions, and teaching strategies. In this context, unpacking the matrix and extending the IDTML in a pedagogical approach help us to become "informed design teachers".

5 CS1 Course

Introductory programming courses develop several learning activities and programming projects in an attempt to change negative outcomes, high failure and drop-out rates [1, 6, 30, 31]. Various problems experienced by novices were identified relating to basic program design and algorithm complexity. Programming courses suffer from a wide range of difficulties and deficits [30].

Gomez and Mendez [32] argues that programming is a complex subject that requires effort and a special approach in the way it is learned and taught. To become good programmers, students must acquire several abilities and traditional teaching methods do not seem to be adequate for all students' needs. Also, they describe different reasons why learning programming is inherently difficult: teaching is not personalized, teaching strategies do not support all students' learning styles, teachers are more concentrated on teaching programming languages instead of promoting problem solving, programming demands a high level of abstraction and programming languages have complex syntax.

Literature of learning programming contains many research studies that have been proposed to face these problems with different approaches; for example: games [33,34], robots[35], pair programming [36,37].

Our CS1 course at Universidad ORT Uruguay emphasizes teaching problem-solving methodology using an Object Oriented Programming (OOP) approach. The course prepares the learner for constructing simple programs using the OOP paradigm. By the end of the semester, the student will be ready to analyze simple situations, to design possible solutions and to implement them with an OOP approach. Our teaching strategy is based on designing activities that motivate and engage students in order to achieve better results.

The duration of the course is 15 weeks, 4 hours of lectures and 2 hours for lab session per week. The programming language used is Java for all these assignments. A brief description of the 15-week course is shown in Table 1.

The main topics are: pseudo code, variables, and control structures, objects and classes, association, inheritance, aggregation and collections, enumeration, sorting and searching, and advanced use of collections.

The course includes two compulsory programming assignments (done in pairs) and a final evaluation. The first programming assignment has 20 points; the second 30 and the final evaluation 50. If the student achieves 86 points or more passes the course and does not have to take a final exam (AP1). Between 70 and 85 the course is approved but the student must take a final exam (AP2). With less than 70 points the student fail and must retake the course (FL).

Table 1: CS1 Course Description

Week	Topics
1-3	Variables, pseudo code, control structures.
4	Classes and Objects.
5-7	Relations between classes: Association.
8	Relations between classes: Inheritance.
9-10	Relations between classes: Aggregation. Collections.
11	Enumerations.
12	Sorting and Searching.
13-15	Advanced use of Collections.

6 Proposal: Unpacking the Matrix

IDTLM proposed by Crismond and Adams [12], suggests pedagogical strategies to help teachers design pedagogical content. IDTLM shows key concepts of what instructors needed to know to use design activities effectively in the classroom.

In this context, we unpack the IDTLM, generating informed design activities based on competences in order to satisfy learning goals and develop selected competences of TLAP.

Our proposal is to design pedagogical activities for CS1 using the IDTLM as a framework. We use several teaching strategies (TS1- TS9) proposed by Crismond and Adams [12] and we integrate a variety types of activities, each one designed with the IDTLM design strategy (DS1-DS9).

In particular, we propose to extend the IDTLM with concrete activities (Table 2, Column 3) incorporating a list of selected competences that students should develop for each one of the proposed activities (Table 2, Column 4).

Learning objectives and goals of each activity have been defined in this context and we ensure that each one of the selected TLAP competences were covered (completeness of the selected TLAP).

Table 2: Unpacking the Matrix

Design Strategy [12]	**Unpacking the Matrix**		
	Informed Designer Pattern [12]	***Activity Proposed***	***Competence***
(DS1) Understand the Challenge (Problem Framing)	Delay making design decisions in order to explore, comprehend and frame the problem better.	***Scratch (week 1)***	C1,C7
(DS2) Build Knowledge (Doing Research)	Do investigations and research to learn about the problem, how the system works,relevant cases, and prior solutions.	***Infographics (week 2)***	C1,C2,C3, C4,C5
(DS3) Generate Ideas (Idea Fluency)	Practice idea fluency in order to work with lots of ideas by doing divergent thinking, brainstorming, etc.	***Wordle (week 5)***	C1,C2,C3,C4,C5
(DS4) Represent Ideas (Deep Drawing & Modeling)	Use multiple representation to explore and investigate design ideas and support deeper inquiry into how the system works.	***Modeling Clay (week 4), Tools (week 15)***	C1,C3,C4,C6,C7, C9
(DS5) Weigh Options & Make Decisions (Balance Benefits & Tradeoffs)	Use words and graphics to display both benefits and tradeoffs of all ideas before picking a design.	***Rubric (week 7)***	C4,C10
(DS6) Conduct Experiments (Valid Test & Experiments)	Conduct valid experiments to learn about materials, key design variables and the system work.	***Puzzle Algorithm (week 3)***	C1,C6,C7,C8,C9

Design Strategy [12]	**Unpacking the Matrix**		
	Informed Designer Pattern [12]	***Activity Proposed***	***Competence***
(DS7) Troubleshoot (Diagnostic Troubleshooting)	Focus attention on problematic areas and subsystems when troubleshooting devices and proposing ways to fix them.	***Concept Test (week 10), Minute Test (week 4)***	C1,C2,C4,C6
(DS8) Revise / Iterate (Iterative Designing)	Do design in a managed way, where ideas are improved iteratively via feedback, and strategies are used multiple times as needed, in any order.	***Video (week 5)***	C2, C3, C5,C8, C9, C10
(DS9) Reflect on Process (Reflective Design Thinking)	Practice reflective thinking by keeping tabs on design strategies and thinking.	***UML Modeling Game (week 6)***	C1,C2,C8,C9

In order to design pedagogical content we define learning outcomes in terms of competences and we align teaching activities with strategies (TS1-TS9) proposed by Crismond and Adams [12]. In Table 3 we map each activity with the selected teaching strategies.

Table 3: Activities Vs Teaching Strategies

Activity	**Teaching Strategies**
1-Scratch	TS1, TS2, TS8
2-Infographics	TS2, TS3
3-Wordle	TS2, TS3
4- Modeling Clay ***5-Tools***	TS4, TS6
6-Rubric	TS5, TS6, TS9
7-Puzzle Algorithm	TS3, TS6, TS9
8-Concept Test, ***9-Minute Test***	TS7
10-Video	TS3, TS8
11-UML Modeling Game	TS4, TS9

A detailed description of each activity proposed is given below:

6.1 Scratch

We used Scratch [38] in the very first weeks with the purpose of improving students' programming experiences and motivation. Scratch can be taken as an auxiliary tool of students' learning of programming, stimulating the students' learning motivation and cultivating their ability of solving practical problems with the computational thinking approach [39]. A list of exercices is given to students. The list includes exercises to develop in Scratch (like to draw a particular grid, see (Fig. 1)), sample codes with simple mistakes to correct and examples to complete.

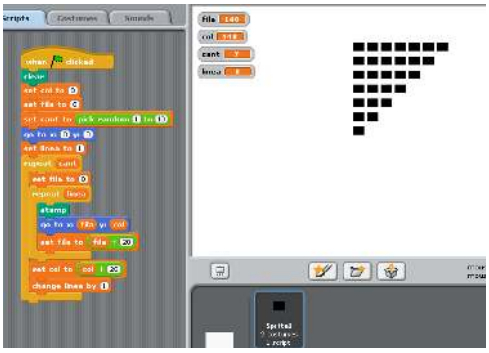

Figure 1: Scratch

6.2 Infographics

Infographics are graphic visual representations of information, data and knowledge that includes text and images, narratives, descriptions, maps, etc. This activity aims to generate computer graphics containing answers to: “What is Java?,” “Where?”, “Who developed it?”, “When?”. As a homework, each student searchs the required information and develops the infographic. In (Fig. 2) there is a sample infographic developed by one student. Students must bring to class the picture. In class, all infographics are discussed and the main topics are referred.

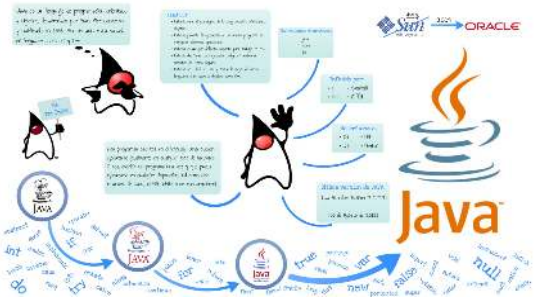

Figure 2: Infographics (In Spanish)

6.3 Wordle

Students are asked to create at home a "word cloud" from Java code using the Wordle tool [40]. The class "Truck" is the first complete design example of a Java class that is presented in the course. The model consists of a truck, with attributes color and plate number. This includes instance variables and methods of access and modification. Each student creates his or her own "cloud" and in the next class all must provide an explanation of the words that appear more prominently (Fig. 3).

Figure 3: Wordle (In Spanish)

6.4 Modeling Clay

To become familiar with the concept of identifying objects, aliasing and message passing we use a kinesthetic learning activity modeling clay promoting comprehension of object oriented concepts as proposed in [41] . This activity focused on improving the comprehension on the difference between object and class, memory, creation of objects, garbage collector, aliasing and message passing (Fig 4). This activity takes about 20 minutes.

Figure 4: Modeling Clay

6.5 Tools

The teacher brings to class various objects (eg disposable razor, kitchen grater, sandpaper drill, soap, bucket, cotton swabs, etc.). Each group of 4 students choose an object. On a sheet they must write: a) name of the group, b) drawing the chosen object, c) analysis of the object‘s characteristics: functionality, materials, costs, etc.).

Students must answer: what would you like to know more about the subject? What new questions arise? (eg Cost?, Who makes it? Is it recyclable?, etc.). Students must describe how they would answer these questions. From the responses on the characteristics and properties of specific tangible objects, they must find an analogy to desirable properties and characteristics of the software (such as usability, efficiency, reliability, etc.). The activity takes about 50 minutes.

6.6 Rubric

In class we study two tasks from a previous course: one developed properly and one with errors. They are given to each group with the rubric to assess the work. The rubric contains the following areas: organization of documentation, writing and spelling, class diagram, test data, listing, coding style and execution. Work is classified in each area as excellent, good or poor. Each group must assess and justify the chosen category. The activity takes about 30 minutes.

6.7 Puzzle Algorithm

Students should solve the problem of finding the maximum of a list of numbers. We provide students with an algorithm puzzle. Each piece of it is a line of code and there are some extra pieces (different valid solutions could be constructed). Students must select the lines, and recompose the algorithm. In (Fig 5) is presented an initial (and partially incorrect) solution proposed by one student. Students present their own solutions putting it on the blackboard, and in groups discuss and select the most appropiate. This activity takes about 45 minutes.

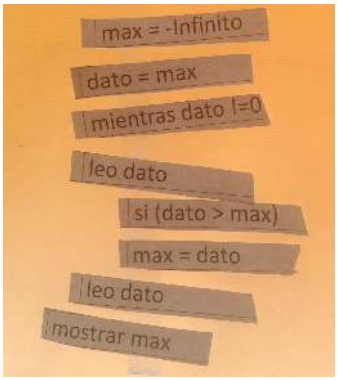

Figure 5: Puzzle Algorithm

6.8 Concept Test

Peer Instruction (PI) [42] engages students during class through activities that require each student to apply the core concepts being presented, and then to explain those concepts to their fellow students. A class taught with PI is divided into a series of short presentations of certain concepts, each focused on a central point and followed by a related conceptual question. We use this activity to develop concepts of “Array”. The activity takes about 50 minutes.

6.9 Minute Test

Angelo and Cross [43] suggest assigning minute papers at the end of class. In one minute, students answer the following questions: (1) What is the most significant thing you learned today?, and (2) What relevant question is in your mind at the end of today's session?. The answers are discussed in groups and the most remarkable points are selected.

6.10 Video

Pair programming consists of two programmers sharing a single workstation (one screen, keyboard and mouse among the pair). The programmer at the keyboard is usually called the "driver", the other, also actively involved in the programming task but focusing more on overall direction is the "navigator"; it is expected that the programmers swap roles every few minutes or so [44]. A short video of pair programming [45] is presented to discuss the concepts of the practice. Students discuss what is the true spirit of pair programming and the advantages and disadvantages of such methodology. The main purpose of this activity is to take into account the required skills and the problems of working in groups. This activity takes about 15 minutes.

6.11 UML Modeling Game

The teacher brings to class several toys in their original packages (e.g. race car, deck of cards, puzzle), see (Fig 6). Students are asked to model the game in UML notation. Information of attributes is implicit in the packages (identification, recommended age, pieces, etc.). Different models are sketched on the blackboard and several valid options are analyzed. This activity takes about 20 minutes.

Figure 6: UML Modeling Game

7 Experimentation

The research question that motivated this study was: What kind of competence-oriented activities can we design to engage students and obtain better results?.

In terms of GQM [46] (Goal, Quality, Metrics): Our study aims to generate informed design activities in order to enhance the development of several TLAP competences, providing better results in CS1 course from the educational perspective in the context of introductory programming (CS1) course at Universidad ORT Uruguay.

The independent variable chosen in the experimental design was the teaching method, which incorporates IDTLM design strategy and selected TLAP competences. The dependent variable is the course result.

An initial experiment was conducted in the second semester of 2012. One random group of 20 freshmen in the experimental group (EG) was selected to participate in the activities and 16 students were in the control group (CG). Students in the EG receive instruction though IDTLM design strategy and selected TLAP competences and students in CG received instruction through the use of traditional lecture. Common syllabus, notes, assignments, teachers experience and classroom were used in (CG) and (EG). Both groups included a high number of students who fail in previous course. A limitation of the study was the low number of freshmen. Considering only the students who take the course for first time, in the selected group, 44% (4/9) of them had completed the course and in the control group, 22% (2/9).

Based on that experience, we designed some improvements in the activities and replicated the study. In the first semester of 2013, two groups of students corresponding to CS1 courses were randomly selected to participate in the experiment. In the EG were 24 students and in the CG were 22 students.

An initial test on Java, object oriented concepts and programming was designed and applied in both groups. The scores showed that both groups were similar in terms of CS knowledge and the students were essentially novices.

As we exposed, final results of assessments establish a classification into 3 groups according to final scores in a 0-100 scale: students that must retake the course (1-69) (FL), students that must take a final exam (70-85) (AP2) and

finally the category of students that approve the course (86-100) (AP1). Fig. 7 illustrates final results of assessments for EG and CG.

The dot plot shows a distribution of final result of each student. In the EG, most of the students are in the AP1 area (86-100 points) and in the CG they are distributed all over the values.

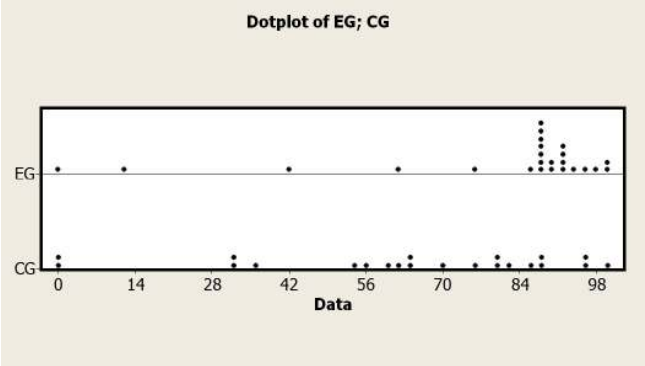

Figure 7: DotPlot (EG) and (CG)

Table 4 illustrates final results for both (EG) and (CG).

Table 4: CS1 Final Results

	Final Results	
Categories Scale (0-100)	Experimental Group (EG)	Control Group (CG)
FL (0-69)	4 (16,7%)	11 (50%)
AP2 (70-85)	1	5
AP1 (86-100)	19	6
AP1+AP2	20 (83,3%)	11 (50%)
Total	24	22

Table 5 illustrates descriptive statistics outputted by Minitab [47].

Table 5: CS1 Descriptive Statistics

	Descriptive Statistics	
	Experimental Group (EG)	Control Group (CG)
Mean	79.75	63.32
StDev	26.0	28.49
Maximum	100	100
Minimum	0	0
Median	88	66.50
N	24	22

In order to evaluate the proposed pedagogical design, we tested the following hypotheses:

Null Hypothesis: H0: There is no difference in applying informed design activities with a competencies oriented approach in the final results of our students.

Alternative Hypothesis: H1: There is a difference in applying informed design activities with a competencies oriented approach in the final results of our students.

Different non-parametric methods can be applied in our experiment. We select Mann-Whitney U [48] (MWU) as the non-parametric context of the experiment. MWU is the non-parametric equivalent of Student's test or the t-test for two samples. Table 6 illustrates the significance level (p-value) of the Mann Whitney U Test, obtained by SPSS [49].

Table 6: Mann Whitney U Test

	Mann Whitney U Test	
	N	**Mean Rank**
Experimental Group (EG)	24	29,02
Control Group (CG)	22	17,48
Output	Mann Whitney U – 131.500 Z= -3.252 Asymp. Sig (2-tailed) , 0,001	

In terms of statistical significance (measured by alpha (α)) a high level of significance ($\alpha < 0.01$) was found. As the value of p-value shown in Table 7 is 0,001 (less than 0.01) we reject H0 and accept H1. In this context, there is statistical evidence to conclude that the performance of the experimental group is significantly different. Considering the detailed final results of the students and the MWUT, we could infer that the proposed pedagogical design produce significantly better final results.

Teachers of the experimental group referred a high level of enjoyment. In class, informally talking with students they show high levels of motivation.

Related to the activities, a survey was conducted among students in the experimental group (EG). We asked them to order the activities according to their preference. Each student selected their three preferred activities. Integrating the answers, the three activities preferred by all the students were: Concept Test, Puzzle Algorithm, and Modeling Clay. A brief description of the results of student's survey preference of proposed activities is shown in Table 7.

Table 7: Activities Survey

Ranking	**Activity**
1	Concept Test
2	Puzzle Algorithm
3	Modeling Clay
4	Infographics
5	Scratch
6	UML Modeling Game
7	Wordle
8	Video
9	Minute Test
10	Rubric
11	Tools

8 Conclusion and Future Work

Unpacking the IDTLM allowed out teachers to generate informed design activities based on competences in order to satisfy learning goals. Our findings suggest that this pedagogical design approach seems to be more effective in engaging students in order to achieve better results.

As mentioned in the experimentation section, in the experimental group 83.3% of the students approved the course and in the control 50%. Also, teachers referred to high motivated students in the experimental group. The main internal threat to validity that could be identified is the size of the groups.

This set of didactic units and activities are the outcome of the work and didactic units are available to all CS1 teachers at Universidad ORT Uruguay. A subset of these exercises obtained one of the first prizes in the PRECITYE Program [50] (Regional Program in Engineering Entrepreneurship and Innovation).

In the future we will incorporate the Informed Design Rubric [51] proposed by Crismond to assess student learning over one or more design activities proposed. Also, we will incorporate other approaches and monitoring the performance of students in the following courses.

References

[1] Kinnunen, P., Malmi, L., "Why Students Drop Out CS1 Course?", *ICER' 06,* United Kingdom, pp 97-108, 2006.

[2] Jenkins, T., "On the difficulty of learning to program.", *Proceedings of the 3rd Annual Conf. of the LTSN Centre for Information and Computer Sciences.* Loughborough: Univ. United Kindom, pp 53-58, 2002.

[3] Gomes, A., Mendes, A.J., "Learning to program- difficulties and solutions*", International Conf. on Engineering Education ICEE,* 2007 .

[4] Sheard, J., Simon, S., Hamilton, M. and Lonnberg, J, "Analysis of research into the teaching and learning of programming", *Proceedings of the fifth International Workshop on Computing Education Research WorkShop ICER 09*, pp.93-104, 2009.

[5] Soh, L. K., Samal, A. and Nugent, G., An integrated framework for improved Computer Science Education: Strategies, implementations, and results". *Computer Science Education*, 17, pp 59-83, 2007.

[6] Ambrosio, A.P., Costa, F.M., Almeida, L.,Franco, A., Macedo, J., "Identifying cognitive abilities to improve CS1 outcome", *Proc. of 41th ASEE/IEE Frontiers in Education Conference*, pp. F3G-1 F3G-7, 2011.

[7] Lahtinen E., Ala-Mutka K., and Järvinen H. M., "A study of the difficulties of novice programmers". *SIGCSE Bull. 37,* pp 14-18, 2005.

[8] Tovar, E. and Soto, O., "Are new coming computer engineering students well prepared to begin future studies programs based on competences in the European Higher Education Area?", *39th IEEE Frontiers in Education Conference, 2009.* FIE '09, 2009.

[9] Ramirez, C.; Sanchez, E., "Competences memory map: A model for the representation of competences applied in education," *2012 IEEE 11th Int. Conf. on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, pp.363,371, 2012.

[10] Commission of the European Communities, "Recommendation of the European Parliament and of the Concil on the establishment of the European Qualifications Framework for lifelong learning", Available at http://ec.europa.eu/education/policies/educ/eqf/rec08_en.pdf , last accessed October 15 2013.

[11] "Software Engineering 2004" Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, The joint Task Force on computing Curricula, IEEE Computer Society, Association for Computing Machinery, August 2004.

[12] Crismond, D. P. and Adams, R. S., "The Informed Design Teaching and Learning Matrix". *Journal of Engineering Education,* Vol 101, No. 4, 738–797, 2012.

[13] Edwards, M., Tovar, E. and Soto, O., "Embedding a core competence curriculum in computing engineering," *Frontiers in Education Conference, 2008. FIE 2008*, pp.S2E-15,S2E-20, 22-25 Oct. 2008.

[14] González, J., Wagenaar, R., "Universities contribution to the Bologna Process. An Introduction (2nd edition)". Publicaciones de la Universidad de Deusto, 2008 [On-line].Available:

http://www.unideusto.org/tuningeu/images/stories/Publications/ENGLISH_BROCHURE_FOR_WEBSITE.pdf , last accessed October 15 2013.

[15] Tovar, E.; Soto, O., "The use of competences assessment to predict the performance of first year students," *Frontiers in Education Conference 2010*, pp.F3J-1,F3J-4, Oct. 2010.

[16] Sanchez, J.L.; Gonzalez, C.S.; Alayon, S., "Evaluation of transversal competences in the final year project in engineering", *2011 Proc. of the 22nd EAEEIE Annual Conference (EAEEIE),* pp.1,5, June 2011.

[17] Kabicher, S.; Motschnig-Pitrik, R.; Figl, K., "What competences do employers, staff and students expect from a Computer Science graduate?", 39th Frontiers in Education Conf., pp.1,6, 18-21 Oct. 2009.

[18] Tuning America Latina, http://tuning.unideusto.org/tuningal/index.php?option=com_docman&Itemid=191&task=view_category&catid=22&order=dmdate_published&ascdesc=DESC, last accessed October 15 2013.

[19] Biggs, J., & Tang, C. , "Teaching for quality learning at university", Open University Press., 2011.

[20] Gal-Ezer J.and Harel D., "What (Else) should CS Educators know?", *Comm. ACM, vol 41, no. 9*, pp 77-84, 1998.

[21] Felder R. M.,Brent R, The ABC's of Engineering Education: Abet, Bloom's Taxonomy, Cooperative Learning, and So on, *Proc. of the 2004 American Society of Engineering Education annual conference and exposition*, Session 1375, 2004.

[22] Tulbure, C. , "Learning styles, teaching strategies and academic achievement in higher education: A cross-sectional investigation". Procedia-Social and Behavioral Sciences, 33, 398-402, 2012.

[23] Felder, R.M. and Brent, R. , "Understanding Student Differences", *Journal of Engineering Education*, 94(1), 57-72, 2005.

[24] Hamer, J., Luxton-Reilly, A., Purchase, H. C., & Sheard, J. , "Tools for contributing student learning". *ACM Inroads*, 2(2), 78-91, 2011 .

[25] Pears A., Seidman S., Malmi L., Mannila L., Adams E, Bennedsen J, Devlin M., and Paterson J., "A survey of literature on the teaching of introductory programming". *SIGCSE 2007,* 2007.

[26] Sheard J., Simon S.,Hamilton M and Lönnberg J.. Analysis of research into the teaching and learning of programming. *In Proceedings of the fifth international workshop on Computing education research workshop (ICER '09)*, ACM, New York, NY, USA, 93-104, 2009.

[27] Davis, M., Hawley, P., McMullan, B., & Spilka, G., *"Design as a catalyst for learning".* Alexandria, VA: Association for Supervision and Curriculum Development, 1997.

[28] Adams, R.S.; Fralick, B., "Work in progress — A conceptions of design instrument as an assessment tool," *Frontiers in Education Conference (FIE) 2010 IEEE*, pp.F2G-1,F2G-2, Oct. 2010.

[29] Crismond, D., Desing Practices and Misconceptions, *The Science Teacher*, 2013.

[30] Robins, A., Rountree, J. and Rountree, N., "Learning and teaching programming: a review and discussion", *Computer Science Education*, Vol 13, No 2, pp. 137-172, 2003.

[31] Bennedsen, J., Caspersen, M., "Failure Rates in Introductory Programming", *Inroads SIGCSE Bull.,* Vol 39:2, , pp 32-36, 2007.

[32] Gomes A.J., Santos A. N. and Mendes A.J.. A study on students'behaviours and attitudes towards learning to program. *Proceedings of the 17th ACM annual conference on Innovation and technology incomputer science education (ITiCSE '12)*, 2012.

[33] Cliburn D.C., Miller S.M. and Bowring E. , "Student preferencesbetween open-ended and structured game assignments in CS1",*Frontiers in Education Conference (FIE),* pp.F2H-1-F2H-5, 2010.

[34] Smallwood G.R. and Black D.V. "Observations on designing a computer science Curriculum focusing on game programming using testimonials from industry leaders", *2011 IEEE International Games Innovation Conference (IGIC),* 2011, pp.126-129, 2011.

[35] De Giusti A., Frati F. E., Sanchez M., De Giusti L. "LIDI Multi RobotEnvironment: Support software for concurrency learning in CS1", *International Conference Collaboration Technologies and Systems (CTS),* pp.294-298, 21-25 May 2012.

[36] Radermacher A. and Walia G. "Investigating student-instructor interactions when using pair programming: An empirical study" , *24th IEEE-CS Conference Software Engineering Education and Training (CSEE&T),* pp.41-50, 2011.

[37] Shaw A. "Extending the Pair Programming Pedagogy to Support Remote Collaborations in CS Education," *ITNG '09. Sixth International Conference Information Technology: New Generations*, pp.1095-1099, 2009.

[38] Scratch, http://scratch.mit.edu , last accessed April 10, 2013.

[39] Xiaoxia Wang; Zhurong Zhou, "The research of situational teaching mode of programming in high school with Scratch," *6th IEEE Joint International Information Technology and Artificial Intelligence Conf.(ITAIC)*, vol.2, no., pp.488,492, 20-22 Aug. 2011.

[40] Wordle, http://www.wordle.com , last accessed October 15 2013.

[41] de Kereki, I. F. "Incorporation of Kinesthetic Learning Activities to Computer Science 1 course: Use and Results". *CLEI Electronic Journal.* V. 13, N.2, Paper 1, 2010.

[42] Crouch C.H. and Mazur E. "Peer Instruction: Ten years of experience and results" , *American Journal of Physics*, vol. 69, no. 9, p. 970, 2001.

[43] Angelo, T.A., and Cross, K.P. *"Classroom Assessment Techniques",* 2nd ed., Jossey-Bass, San Francisco, pp. 148-153, 1993.

[44] Pair Programming. Agile Alliance. http://guide.agilealliance.org/guide/pairing.html, last accessed October 15 2013.

[45] Pair Programming Video, http://www.youtube.com/watch?v=oINaUu_wnqo, last accessed October 15 2013.

[46] Basili Victor, Software Modeling and Measurement: The Goal/Question/metric Paradigm. Technical Report. University of Maryland at College Park, College Park, MD, USA, 1992.

[47] Minitab, http://www.minitab.com , last accessed October 15 2013.

[48] SPSS, Mann Whitney- U Test , http://pic.dhe.ibm.com/infocenter/spssstat/v22r0m0/index.jsp?topic=%2Fcom.ibm.spss.statistics.help%2Fspss%2Fbase%2Ftwo_independent_samples_test_types.htm , last accessed October 15 2013.

[49] SPSS, http://www-01.ibm.com/software/analytics/spss , last accessed October 15 2013.

[50] IngEmprendedores, http://www.ingemprendedores.org/tag/precitye , last accessed October 15 2013.

[51] Informed Design Rubric, www.nsta.org/highschool/connections.aspx , last accessed October 15 2013