Informatics in Education, 2017, Vol. 16, No. 2, 141–164

DOI: 10.15388/infedu.2017.08

Design and Large-scale Evaluation of Educational Games for Teaching Sorting Algorithms

Paulo Eduardo BATTISTELLA,
Christiane Gresse von WANGENHEIM,
Aldo von WANGENHEIM, Jean Everson MARTINA
Graduate Program in Computer Science, Federal University of Santa Catarina
Florianópolis/SC, Brazil
e-mail: paulo@incod.ufsc.br, {c.wangenheim,aldo.vw,everson}@inf.ufsc.br

Received: December 2016

Abstract. The teaching of sorting algorithms is an essential topic in undergraduate computing courses. Typically the courses are taught through traditional lectures and exercises involving the implementation of the algorithms. As an alternative, this article presents the design and evaluation of three educational games for teaching Quicksort and Heapsort. The games have been evaluated in a series of case studies, including 23 applications of the games in data structures courses at the Federal University of Santa Catarina with the participation of a total of 371 students. The results provide a first indication that such educational games can contribute positively to the learning outcome on teaching sorting algorithms, supporting the students to achieve learning on higher levels as well as to increase the students' motivation on this topic. The social interaction the games promote allows the students to cooperate or compete while playing, making learning more fun.

Keywords: evaluation, educational game, data structures, teaching, computing.

1. Introduction

Sorting algorithms, such as *Quicksort* and *Heapsort,* are essential topics in undergraduate computing courses (ACM/IEEE-CS, 2013; Poulova and Klimova, 2015; Chen and Fu, 2010). Following curriculum guidelines, it is expected that students learn competences related to these algorithms in data structures courses on the cognitive levels of comprehension and application (ACM/IEEE-CS, 2013; Bloom, 1956). Typically these competences are taught through traditional lectures (ACM/IEEE-CS, 2013; Denton *et al.*, 2005). This method is appropriate to present abstract concepts and factual information to a large group of students (Choi and Hannafin, 1995; Freeman *et al.*, 2014). Often this method is complemented by exercises focusing on the implementation of these algorithms (Ernst *et al.*, 2009). However, the use of other strategies such as Game-based

Learning (GBL), can bring benefits that improve the learning by the application of this knowledge in a motivating and fun way (Poulova and Klimova, 2015; Hakulinen, 2012; Hakulinen, 2011; Freeman *et al.*, 2014).

Game-based learning focuses on the application of educational games that are designed to balance learning to a particular competence with gameplay (Qian and Clark, 2016). They are currently adopted in computing education in diverse areas, such as software engineering, programming or security (Battistella and Gresse von Wangenheim, 2016; Boyle *et al.*, 2016). Yet, so far they are not much used for teaching data structures knowledge. The few games that exist in this context focus mainly on teaching concepts related to the analysis of the complexity and/or other characteristics of the algorithms (e.g. *Sorting Game* and *Sorting Casino* (Hakulinen, 2011), *DSAsketch* (Hakulinen, 2012), *Critical Mass* (Lawrence, 2004), *Active Learning* (Budd, 2006), *GBL for Data Structures* (Tan and Seng, 2010)). Another shortcoming is the lack of sound empirical evidence of the benefits of these games in data structures courses. As currently most of these educational games were evaluated only in an *ad-hoc* manner and/or with very small samples (typically limited to the size of one class), the validity and generalizability of the results remain questionable (Calderón and Ruiz, 2015; Battistella and Gresse von Wangenheim, 2016; Boyle *et al.*, 2016).

In this context, this paper proposes and evaluates three educational games to complement the learning of the algorithms *Quicksort* and *Heapsort*. The results of this research are expected to provide effective instructional alternatives for teaching sorting algorithms. Furthermore, the results of the large-scale evaluation of the games provide a sound empirical base for the adoption of the games in the classroom as well as identifying strengths and weaknesses of such games as a basis for improvement.

2. Background: Teaching Sorting Algorithms

2.1. *Sorting Algorithms*

The teaching of sorting algorithms is typically part of the core content of data structures courses of undergraduate computing courses (ACM/IEEE-CS, 2013). As part of the data structures course, students need to learn to understand simple numerical algorithms, calculating the average value of data in a list , search the minimum and maximum values in a list, sequential or binary search algorithms, analyze the complexity of algorithms, create *table-hash* and perform basic operations in search-trees such as insertion, deletion, alteration (ACM/IEEE-CS, 2013). Usually, it is required that students know to implement stacks, queues, algorithms of quadratic complexity (*Selectionsort*, *Insertionsort*) and quasilinear algorithms (*Quicksort*, *Heapsort*, *Mergesort*) (ACM/IEEE-CS, 2013; Ernst *et al.*, 2009). At the undergraduate level, it is expected that the students acquire competences on the cognitive levels of comprehension and application, according to the Bloom’s taxonomy (1956). At the comprehension level it is expected that the students comprehend the design principles and the structure of the algorithm. At the application level, students are expected to be able to, step by step, apply the sorting algorithms to a set of random numbers without consultation of the written algorithm. The authors selected the *Quicksort* and *Heapsort* algorithms from the contents of an undergraduate data structures course for the production of educational games. In previous years, these algorithms were taught only through traditional lectures, as depicted in Fig. 2. Thus, the games have been created to allow students instrumentalise the knowledge they acquired during their study of the algorithms.

Quicksort is one of the most popular sorting algorithms in literature, being based on the divide and conquer algorithm design technique (Sane and Deshpande, 2006). There are several versions seeking to improve its operation (Munro and Ji, 1996; Stone, 1978; Prifti, 2015). The algorithm was proposed by Hoare in 1959, but is still taught in computing courses (Reilly, 2003; ACM / IEEE-CS, 2013). *Quicksort* starts by choosing an element of a previously unsorted array, which is called the *pivot*. Then, the algorithm compares the values of the array with the pivot. Values that are lower than the pivot are placed on the left of the pivot and, values that are larger are placed on the right of the pivot. This is the main operation of the Quicksort algorithm, called *partition*, where the array is subdivided into two disjoint subarrays. The algorithm repeats this step recursively with each subarray in order to complete the sorting of the array (Sane and Deshpande 2006). The complexity of the algorithm is quadratic (n^2) in the worst case (when the array is sorted or inverse-sorted) and quasilinear ($n \log n$) in the average case. It has been extensively analyzed (Munro and Ji, 1996; Stone, 1978; Prifti, 2015).

Heapsort is a sorting algorithm that uses a vision of an array as a left-balanced heap-tree to perform the sorting of this array. It was developed by Williams and refined by Floyd in 1964 (Wegner and Teuhola, 1989). In order to sort the elements of the array, the algorithm performs a process called sifting-up. While searching the array, *Heapsort* looks locally at each father/children set, accordingly to the heap-tree vision, and places the highest value in the father and lowest in one of the children (Sane and Deshpande, 2006; Wegner and Teuhola, 1989; Bai *et al.*, 2014). The logic behind this algorithm is to gradually move the larger elements to the beginning of the array (upper levels of the heap-tree), inducing a partial order where, for each subtree, each individual node is always larger as its descendants, without ordering the whole heap. When the largest value of the array is inserted into the root, the value of the first position should be replaced with the value of the last position. Whenever a value is moved to the end of the array, the array size is decremented and sifts-up the array again to ensure that the there is no father larger than its children (Sane and Deshpande 2006; Wegner and Teuhola 1989; Bai *et al.*, 2014). The *Heapsort* algorithm is very stable and shows quasilinear complexity ($n \log n$) in all cases.

2.2. *Educational Games*

A game can be defined as "*any competition (game) between opponents (players) operating under constraints (rules) for a goal (win or profit)*" (Abt, 2002). In this context, educational games are specifically designed to teach a particular competence (Qian and Clark, 2016; Abt, 2002; Dempsey, 1996). Different types of games can be used for education, including digital or non-digital ones. Digital (electronic) games involve interaction with an electronic device (e.g., a computer, console, tablet or smartphone). Non-digital games are manual games such as board games, cards or using simply pens and paper or other props (Connolly *et al.*, 2012; Caulfield, Veal, and Maj, 2011). Games can be played in different modes of interaction, for example as *single-player* or *multiplayer* (Schell, 2008; Fullerton, 2008).

Diverse game genres are adopted also for computing education as shown in Table 1 (Battistella and Gresse von Wangenheim; 2016, Schell, 2008; Fullerton, 2008; Adams and Rollings, 2006; Wolf, 2001; Herz, 1997).

In an educational context, games can be adopted in order to focus on different levels of learning, ranging in undergraduate courses typically from the level of remembering (exhibit memory of previously learned materials by recalling facts and concepts), comprehension (demonstrate understanding of facts and ideas) to the level of application (solve problems to new situations by applying acquired knowledge) based on Bloom's Taxonomy (1956) (ACM/IEEE-CS, 2005).

Table 1

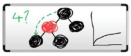
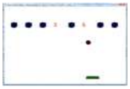
Educational game genres

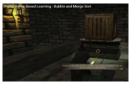
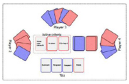
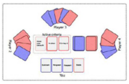
Genre	Description
Action	A game that requires players to use quick reflexes, accuracy, and timing to overcome obstacles, to solve challenges or to answer questions.
Adventure	A game in which the player assumes the role of the protagonist in an interactive story driven by exploration and the solution of challenges/quests (such as decoding messages, finding and using items) in order for the narrative to progress.
Guessing	A game in which the object is to identify some kind of information, such as a word, from other player's drawings or mimicking.
Puzzle	A game that involves controlling game characters using a set of commands and executing them, to navigate through a maze or to arrange objects in a correct and efficient way to achieve the game goal.
Quiz	A game where the player is presented with trivia questions and must select or give the right answer.
Racing	A game in which the player commands a vehicle or participates in a race trying to move faster than an opponent to reach a specified goal or beat a specified time.
Roll-and-move	Board games in which a player's token or tokens are moved based on results shown on a die or dice.
Role-playing game (RPG)	A game where the player controls the actions of a protagonist as this character lives immersed in a fictional world. A role-playing game can be seen as such when player-controlled characters become stronger ("levels up") because of the player's actions (usually depending on experience points received), rather than being upgraded automatically as dictated by the storyline.
Simulation	A game designed to put the player in control of a certain environment or activity being mimicked as realistic as possible.
Strategy	A game that focuses on gameplay requiring careful and skillful thinking and tactics in order to achieve victory.

2.3. *Teaching Sorting Algorithms with Educational Games*

Considering the current tendency to adopt educational games for computing education, they are still not widely used for teaching of sorting algorithms (Battistella and Gresse von Wangenheim, 2016a; Boyle *et al.*, 2016). Based on a literature review, we only identified 7 educational games focusing on the teaching of sorting algorithms in higher computing education as presented in Table 2.

Table 2
Educational games for teaching sorting algorithms

Name	Image Description	Type	Interaction	Genre	Content taught by the game	Learning objectives	Reference
1	2	3	4	5	6	7	8
DSAsketch	 Game that covers a wide range of data structures and algorithms concepts in order to recap the course material. Players are divided into two teams. The main idea of the game is to draw concepts related to data structures and algorithms to the whiteboard, while other team members are trying to guess the concept in the picture. The winner of the game is the team with the most points after all players have had a turn to draw.	Non-digital	Multiplayer	Guessing	Linear data structures, manipulation of data in tree, and recursive.	Comprehension; Remembering	(Hakulinen, 2012)
Lightest and Heaviest	 Students manually sort small boxes with different weights using a scale. Applying the algorithms students must choose between couples of boxes based on their weight to order them.	Non-digital	*Single-player*	Guessing	Sorting algorithms Bubblesort, Insertionsort and Selectionsort	Remembering	(Computer Science Unplugged, 2005)
SAVG-Engine	 Game engine developed with the Microsoft XNA Game Studio created for teachers to develop sorting algorithm games. Three existing games prototypes focus on binary search algorithm, singly-linked list and binary search tree. To create a new algorithm game, the developer needs to create a new algorithm game project. Then, the developer must set up the game properties, assets, graphics, items, classes and screens using the Properties, Assets, Graphics Items, classes, and Screens Game Editors respectively. After that, the developer needs to implement the required methods of the algorithm game, and adds any needed code. Finally, the developer needs to compile, debug and execute the created algorithm game.	Digital	*Single-player*	Puzzle	Linked list, but also allows teachers to implement games related to other sorting algorithms.	Comprehension; Remembering	(Shabanah *et al.*, 2010)

1	2	3	4	5	6	7	8
Sorting Game	 Students collect boxes and sort them according to their values by using sorting algorithms (Bubblesort and Mergesort). The game features a medieval scenario based on the famous game World of Warcraft (RPG). The game supports single player and multiplayer modes. Both game modes support spectator mode, which can be used by any lecturer to act as a facilitator. To win the game, it is necessary to correctly order the boxes, from lowest to highest.	Digital	*Single-player* and Multiplayer	RPG	Quadratic complexity algorithm (Bubblesort) and the quasilinear complexity algorithm (Mergesort).	Application; Comprehension; Remembering	(Cesare, 2012)
Sorting Casino	 In this game, which resembles the card game casino, the player uses cards for defining characteristics of a sorting algorithm, for example, setting its complexity. On his/her turn, the player can capture one or more cards from the table by using one of her/his hand cards. With an algorithm card, the player can capture all special cards that are valid for that algorithm. A player can put all the captured cards to his victory stack. The game ends when either the algorithm stack or special card stack is empty. The winner is the player with most cards in his/her victory stack.	Non-digital	Multiplayer	Action	Complexity of sorting algorithms	Comprehension; Remembering	(Hakulinen, 2011)
Sorting Game	 This game uses the same cards as used in the game Sporting Casino. The basic difference is that in this game are used special cards on complex algorithms. In the beginning, the dealer deals 3 algorithm cards and 2 special cards to each player. The actual game round consists of two phases: special card phase and algorithm phase. During the special card phase, each player can place one special card to the table. In algorithm phase, each player places one algorithm card to the table. The algorithm should be valid for the active special cards on the table. The winner of the round is the player whose algorithm's asymptotical time complexity is the best. The winner collects all the cards played during the round and adds them to his or her victory stack.	Non-digital	Multiplayer	Action	Complexity of sorting algorithms	Comprehension; Remembering	(Hakulinen, 2011)
Sortko	 Designed using an interactive graphical user interface with an option of choosing a sort (e.g. Bubblesort, Shellsort, Quicksort, Insertionsort) and generates a random sequence of numbers to be sorted by applying a sequence of specific interaction sorting gestures. During the sorting process, the system informs students about their success in performing so called "switches" and provides them with textual help messages which facilitate further progress. As a result of one game, a student gets rewarded with points that are then summarized on a joint public list. Win the game, the player that performs the ordination correctly of values. The game was designed to be played on Android phones.	Digital	*Single-player*	Simula-tion	Sorting algorithms Bubblesort, Shellsort, Quicksort, Insertionsort.	Application; Comprehension; Remembering	(Boticki, Barisic, Martin and Drljevic, 2012)

This shows that there are currently only a few educational games for teaching sorting algorithms available, including digital as well as non-digital games. The non-digital games rather focus on lower learning levels (remembering and comprehension) in contrast to some of the digital games that also aim at learning on the application level. And, although, several of the games aim at teaching the application of the sorting algorithms, their benefits remain questionable due to a lack of (rigorous) evaluations with sample sizes large enough to obtain significant results. We also observed a lack of information on how the games have been developed and their characterization with respect to educational aspects.

3. Research methodology

This article aims at the conduction of an exploratory research related to games to teach *Quicksort* and *Heapsort* sorting algorithms. A case study was performed to understand the phenomena observed by the application of the educational games (Fig. 1).

The case study is performed according to the procedure proposed by Yin (2013) and Wohlin *et al.* (2012):

Study definition. Using the approach *Goal Question Metric* (GQM) (Basili *et al.*, 1994) the purpose of the study is defined and systematically decomposed into analysis questions and metrics based on the MEEGA model (Gresse von Wangenheim *et al.*, 2012; Savi *et al.*, 2011). MEEGA evaluates the quality of educational games with respect to their impact on motivation, user experience and learning. It is currently one of the most commonly used models for the evaluation of educational games (Calderón and Ruiz, 2015; Petri and Gresse von Wangenheim, 2016). Based on the analysis questions the standardized MEEGA questionnaire (Savi *et al.*, 2011) is used.

Study execution. The instructional design of the games was done following the ADDIE model (Branch, 2009). Initially, a context analysis was carried out analyzing the target audience and environment, as well as, identifying learning requirements. According to the context analysis, learning objectives and an instructional strategy were defined, including the content, sequence and teaching methods to be adopted. In accordance with the defined instructional strategy, the games were developed taking into consideration fundamental aspects of game design (Fullerton, 2008; Schell, 2008). The games were

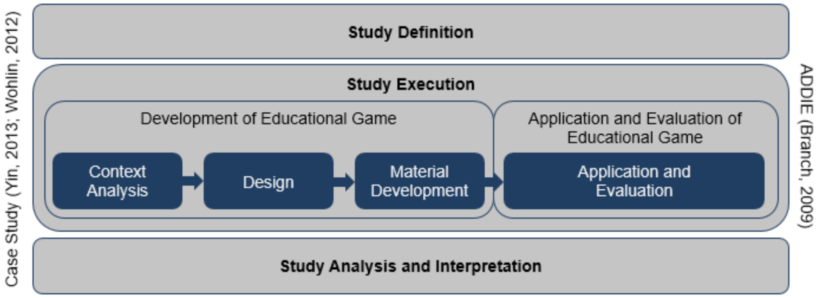

Case Study (Yin, 2013; Wohlin, 2012)
Study Definition
Study Execution
Development of Educational Game
Context Analysis
Design
Material Development
Application and Evaluation of Educational Game
Application and Evaluation
ADDIE (Branch, 2009)
Study Analysis and Interpretation

Fig. 1. Overview on the research method.

applied in several classes in undergraduate computing courses. The evaluation was carried out by a series of case studies with a *one-shot post-test only* design, in which the educational games were applied and afterward data was collected from the students using the MEEGA questionnaire.

Analysis and interpretation. The collected data was analyzed in accordance to the defined analysis questions, using descriptive statistics as indicated by the model MEEGA (Savi *et al.*, 2011). The results were interpreted taking also into consideration further observations made during the applications.

4. Development of the Educational Games for Teaching Sorting Algorithms

4.1. *Context Analysis*

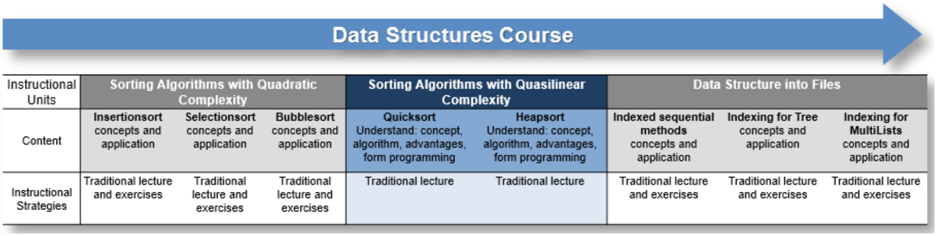
With the objective to improve the teaching of sorting algorithms in undergraduate courses, the target audience is students in the first semesters of computing courses (such as Computer Science and Information Systems). Typically these students already have acquired a basic knowledge of object-oriented programming and repetition and decision structures. Regarding data structures, students usually already have knowledge on sorting algorithms of simple quadratic complexity, such as *Bubblesort*.

In general, computing students are mostly young (19 to 30 years) males. Commonly they play digital games, mainly action and adventure games and RPG (*Role-Playing Games*) for entertainment (Battistella and Gresse von Wangenheim, 2016b). With respect to instructional strategies, the students prefer classes that promote active participation and social interaction with their classmates (Budd, 2006; Qian and Clark, 2016).

In the context of Brazilian public universities, class sizes typical range from 20 to 35 students. Classes are either held in classrooms equipped with a whiteboard and data projector, or computer laboratories. These laboratories typically provide 15-20 computers with Internet access using Windows and/or Linux operating systems (Battistella and Gresse von Wangenheim, 2016b).

Specifically in undergraduate courses at the Department of Informatics and Statistics at the Federal University of Santa Catarina (UFSC), the data structures course is a one-semester course with 108 class hours. On average, 12 hours are spent on teaching sorting algorithms. The teaching of sorting algorithms is divided into two parts in accordance to their complexity. First, quadratic sorting algorithms are taught (such as *Insertionsort*, *Selectionsort, Bubblesort*) and after that, the quasilinear algorithms (typically *Quicksort, Shellsort* and *Heapsort*) are taught. The learning objective of this instructional unit is that at the end of the institutional unit on sorting algorithms with quasilinear complexity, students should be able to understand and apply the sorting algorithms.

Aiming at improving the course, we chose sorting algorithms because this subject was being addressed in a rather abstract way, through traditional lectures based upon structural analyses of the algorithms, where we observed conceptual comprehension difficulties among the students. We specially selected Quicksort, because it is a well-known sorting algorithm that is a standard in data structures course and should be well understood by the



Data Structures Course

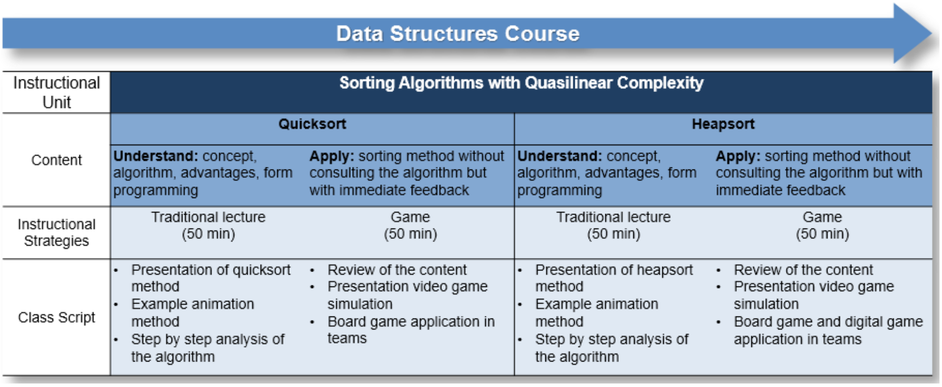
Instructional Units	Sorting Algorithms with Quadratic Complexity			Sorting Algorithms with Quasilinear Complexity		Data Structure into Files		
Content	**Insertionsort** concepts and application	**Selectionsort** concepts and application	**Bubblesort** concepts and application	**Quicksort** Understand: concept, algorithm, advantages, form programming	**Heapsort** Understand: concept, algorithm, advantages, form programming	**Indexed sequential methods** concepts and application	**Indexing for Tree** concepts and application	**Indexing for MultiLists** concepts and application
Instructional Strategies	Traditional lecture and exercises	Traditional lecture and exercises	Traditional lecture and exercises	Traditional lecture	Traditional lecture	Traditional lecture and exercises	Traditional lecture and exercises	Traditional lecture and exercises

Fig. 2. Overview on instructional strategy of the data structures course.

students. Heapsort was selected as its approach differs conceptually from most common sorting algorithms and we observed that it poses a learning challenge to the students.

4.2. *Design of the Educational Games*

Taking into consideration our observations of the major difficulties experienced by students in understanding sorting algorithms with quasilinear complexity and also taking into account the characteristics of the context and the learning objective, we identified the opportunity to improve the instructional strategy. We introduced educational games where the student plays an active role. The student enacts the algorithms through the use of a physical board game that is used, together with poker chips, as a memory model for the manual simulation of the *Quicksort* e *Heapsort* algorithms. This contributes to learning mainly in the levels of understanding and application with respect to the conceptual execution of the algorithm. At this point this does not include any learning objective with respect to the implementation of these algorithms. These contents are presented through traditional lectures, followed by a video that shows a simulation of the usage the board on each game (`http://www.incod.ufsc.br/novaaula/heapsort-simulation/` and `http://www.incod.ufsc.br/novaaula/quicksort-simulation/`) and then the sorting games are applied, as shown in Fig. 3. In addition, students also use the digital game for Heapsort algorithm.



Data Structures Course

Instructional Unit	Sorting Algorithms with Quasilinear Complexity			
	Quicksort		Heapsort	
Content	**Understand:** concept, algorithm, advantages, form programming	**Apply:** sorting method without consulting the algorithm but with immediate feedback	**Understand:** concept, algorithm, advantages, form programming	**Apply:** sorting method without consulting the algorithm but with immediate feedback
Instructional Strategies	Traditional lecture (50 min)	Game (50 min)	Traditional lecture (50 min)	Game (50 min)
Class Script	• Presentation of quicksort method • Example animation method • Step by step analysis of the algorithm	• Review of the content • Presentation video game simulation • Board game application in teams	• Presentation of heapsort method • Example animation method • Step by step analysis of the algorithm	• Review of the content • Presentation video game simulation • Board game and digital game application in teams

Fig. 3. Instructional Unit.

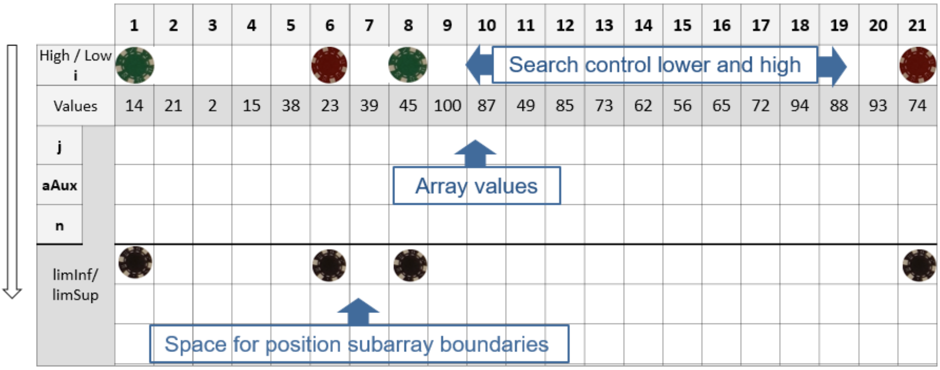
To promote learning on the levels of comprehension and application, simulation games were developed, allowing the students to execute the sorting algorithms step by step. Initially, both board games were created and, then, in order to provide also an alternative beyond the classroom, a digital version of the *heapsort* game was developed.

4.3. *Development of the Games*

4.3.1. *Quicksort Board Game*

The *Quicksort Board Game* is a board game that can be used to represent the memory model of a sequence of recursive calls of the Quicksort algorithm. It allows simulating the execution of the individual steps of the *Quicksort* algorithm (Fig. 4). The game was designed to be played in the classroom in groups of 4 to 6 students with a duration of about 60 minutes. The game uses a board representing the memory area of the values to be sorted, and the contents of the control variables used by the algorithm in the form of an inverted stack containing "photographs" of the subsequent recursive calls to the algorithm.

To play the game, numbered chips are used with values between 0 and 100 arranged in a random sequence on the first row of the array stack of the board. The numbers are inserted on the board randomly, so when starting the game the complexity of the sorting falls in the average case, quasilinear *(n log n)*. Players must use the pointer called "pivot" to sort the values. During each iteration, the array is divided and players must create two new sub-arrays and stack them in a row below, always choosing a pivot for each new sub-array. In the end, the sub-arrays are combined resulting in the sorted array. The group that finishes the sorting first with all values correctly sorted, wins the game. A teaching assistant accompanies the students while they play to ensure that they are correctly following the steps of the *Quicksort* algorithm. A printable model of the board of the game is available (in Brazilian Portuguese) at `http://www.inf.ufsc.br/~aldo.vw/sorting/tabuleiro-sorting-A1.pdf`.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
High / Low i	●					●		●													●
Values	14	21	2	15	38	23	39	45	100	87	49	85	73	62	56	65	72	94	88	93	74
j																					
aAux																					
n																					
limInf/ limSup	●					●		●													●

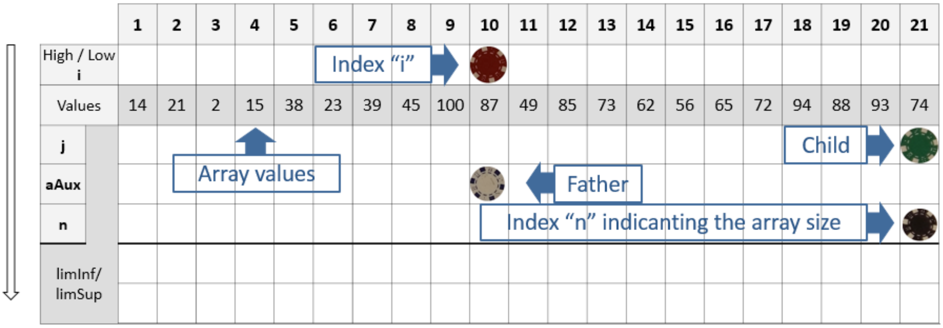
Search control lower and high
Array values
Space for position subarray boundaries

Fig. 4. Board Game for Quicksort Algorithm. Rows labelled *limInf/limSup* represent the stack of recursive calls to *Quicksort* and the corresponding values of the local variables containing each subarray lower and upper boundaries *limInf* and *limSup*.

In the game, student groups should sort numbers by following the steps of the Quicksort algorithm by the students as a group. Within each group the students work in a collaborative way all in the same role as an executor of the algorithm. During the game, the students in each group discuss among themselves the next steps to be taken to sort the data and then execute the step.

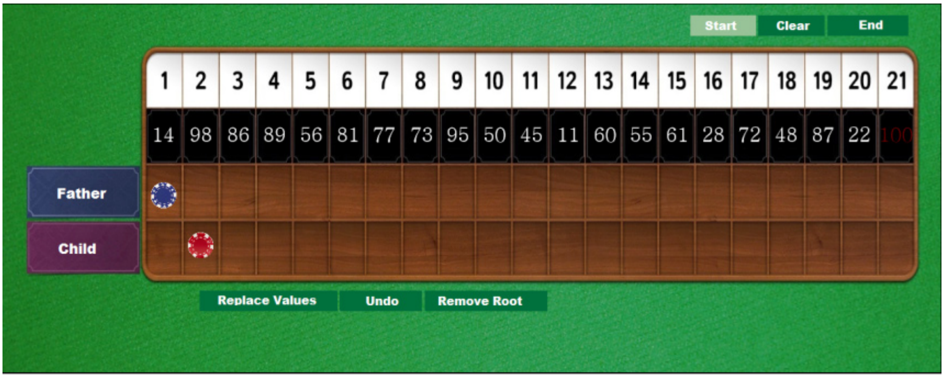
4.3.2. *Heapsort Board and Digital Game*

The *Heapsort Board Game* is a board game simulating the execution of the steps of the *Heapsort* algorithm (Fig. 5). The game is also designed to be played in groups of 4 to 6 students in the classroom with an approximate duration of 60 minutes. In the game, students need to order 21 values (random numbers from 0-100) in an array, accordingly to the *Heapsort* algorithm. The game employs the same game board of our *Quicksort* game, pointers are, however, positioned in different fields and control variables have different meanings. In this case, the memory model represented by the game board is used to simulate the heap tree and the control variables used by the algorithm to traverse this heap during the *sifting-up* process. In order to play, players must place the numbered chips listed in the array and should position special colored poker chips in the positions pointed by the *father*, *child* and *index* variables of the algorithm. Then, the players must decrement the *index* variable until reaching 0, making sure that, at every iteration, every parent node in the heap tree contains a higher value than its children nodes. Children are found from the formula (*index * 2*) and (*index * 2 + 1*), where the *index* is the position of the *father* node in the heap. When a heap is finally built (each parent is larger than its children), the player must move the root (first array element) to the last position. The last unsorted element will be moved to the root, and a sifting-up is performed, to guarantee that a new heap is generated. The game ends when the unsorted array has size 1. The group that finishes the sorting first, wins the game. As in the *Quicksort* the game, a teaching assitant accompanies the students while they play to ensure that they are correctly following the steps of *Heapsort* algorithm. A printable model of the board of the game is also available (in Brazilian Portuguese) at `http://www.inf.ufsc.br/~aldo.vw/sorting/tabuleiro-sorting-A1.pdf`. In the same ways as in the Quicksort game, the students work together in groups, with all students having the same function of sorting the numbers following the steps of the Heapsort algorithm.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
High / Low i																					
Values	14	21	2	15	38	23	39	45	100	87	49	85	73	62	56	65	72	94	88	93	74
j																					
aAux																					
n																					
limInf/ limSup																					

Fig. 5. Board Game for teaching the Heapsort Algorithm.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
14	98	86	89	56	81	77	73	95	50	45	11	60	55	61	28	72	48	87	22	100

Fig. 6. *Heapsort Digital Game.*

The *Heapsort Digital Game* is a digital version inspired by the *Heapsort Board Game* (Fig. 6). As the non-digital version, the game seeks to simulate the execution of the *Heapsort* algorithm, requiring the student to carry out the steps that the algorithm would use to sort the data. The game is played individually so that the student has to sort the values of the array "Array values" using "Father" and "Child" to sort values. To win the game, the student must have ordered all the values of the array, following the steps of *Heapsort*.

To play the game, students should click the "Start" button. Thus, should start ordering the values, using the "Replace values" button to change the value in between the pointers "Father" and "Child". The "Remove root "button is used to switch the first element of the array, with the last element. This button is used when the highest value of the array is at the root. When the player clicks the button, this element is moved to the end of the array. The game also provides a video explaining the operation of the algorithm and the rules of the game. Students learn these rules beforehand during the traditional lectures, but the game also presents rules of the game.

The game has been developed using a client-server architecture to be played online via the browser. The game was developed using the HTML5 language, JavaScript and PHP, and is available online at `http://www.incod.ufsc.br/sortia`.

5. Application of the Educational Games

The games have been applied in data structures courses at undergraduate computing courses (Computer Science and Information Systems) at the Federal University of Santa Catarina (UFSC)/Brazil. Since 2012, the games have been applied in 23 courses with a total participation of 371 students (Table 3). The applications of the games were conducted by the teaching assistant supervised by the lecturer.

According to the defined instructional strategy (section 3) the games have been applied after the respective traditional lectures. In case of the non-digital game, the students were divided into groups of 4 to 6 students. Each group received a board and tokens to

Table 3

Overview on the applications of the games

Educational game	Year / Semester	Course	No. of students
Quicksort Board Game	2012/1	Computer science	27
	2013/2	Computer science	22
	2013/2	Information systems	12
	2013/2	Information systems	5
	2014/2	Information systems	22
	2014/2	Computer science	24
	2015/2	Information systems	4
	2015/1	Computer science	24
	2015/2	Computer science	16
	2016/1	Computer science	27
	2016/1	Information systems	9
Heapsort Board Game	2012/1	Computer science	23
	2014/2	Computer science	11
	2015/1	Computer science	21
	2016/1	Computer science	23
	2016/1	Information systems	10
Heapsort Digital Game	2012/2	Computer science	18
	2014/2	Information systems	17
	2015/2	Information systems	3
	2016/1	Computer science	17
	2016/1	Information systems	8
Heapsort Board and Digital Game	2013/2	Computer science	19
	2013/2	Information systems	9
Total			**371**

mark the pointers of numbered tokens (0 to 100). The mechanics and the rules were explained by the teaching assitant and the students then played the game. Fig. 7 illustrates scenes from the application of the *Quicksort Board Game* and *Heapsort Board Game.* A video showing some student groups playing the game are available here: `http://www.incod.ufsc.br/novaaula/sorting-algorithm-board-game-video/`. In general, students played each game more than once, first to understand the operation of the game and, then again in order to play the game more effectively. The application of both non-digital games took place in classrooms.

The applications of the *Heapsort Digital Game* took place in computer labs. Even though the game was designed to be played individually on a computer, some students played in pairs due to the limited number of computers available, yet also allowing students to help each other. In the beginning, the teaching assitant explained the features and rules of the game.

At the end of each application, students evaluated the games by answering the MEEGA questionnaire (Savi *et al.*, 2011). An exception has taken place in the second semester of 2013, when students played both the *Heapsort Board Game* and *Heapsort Digital Game* and, thus, a single questionnaire has been used to evaluate both games together.

Fig. 7. Scenes of students playing the board games.

6. Evaluation of the Games

6.1. *Definition*

The overall objective is to evaluate the quality of the developed games. Due to the lack of a commonly accepted definition for the quality of educational games, we define a high quality game as one that clearly presents the educational objectives, motivates students and promotes learning curriculum content in a fun, enjoyable and challenging way (Savi and Ulbricht, 2008). Thus, the objective of the evaluation is to analyze the quality (including motivation, user experience and their contribution to learning) of the games *Quicksort Board Game*, *Heapsort Board Game* and *Heapsort Digital Game,* in the context of data structures courses of undergraduate computing courses from the student's point of view.

Based on this objective, we define the following analysis questions:

AQ1. Do the games motivate students?

AQ2. Do the games provide a positive user experience?

AQ3. Do the games contribute to learning?

AQ4. What are differences in the quality of the Quicksort Board Game, Heapsort Board Game and Heapsort Digital Game in relation to motivation, user experience, and learning?

The research hypothesis is that the games contribute to the achievement of the learning objectives of the instructional unit, motivate students and promote a pleasant user experience.

Based on the MEEGA model (Savi *et al.*, 2011) the following dimensions and measurement items are defined as shown in Fig. 8.

For data collection we use the standardized MEEGA questionnaire (Gresse von Wangenheim *et al.*, 2012; Savi *et al.*, 2011) addressing the measurement items as shown in Fig. 8 on a 5-point Likert scale ranging from strongly agree (2) to strongly disagree (–2). In addition, data is collected on a self-assessment of the cognitive level of students before and after the application of the game in relation to the specific learning objectives of every game. The achieved learning level with respect to each learning objective is evaluated by students in relation to their knowledge before and after the game application on an interval scale ranging from 0 to 5.

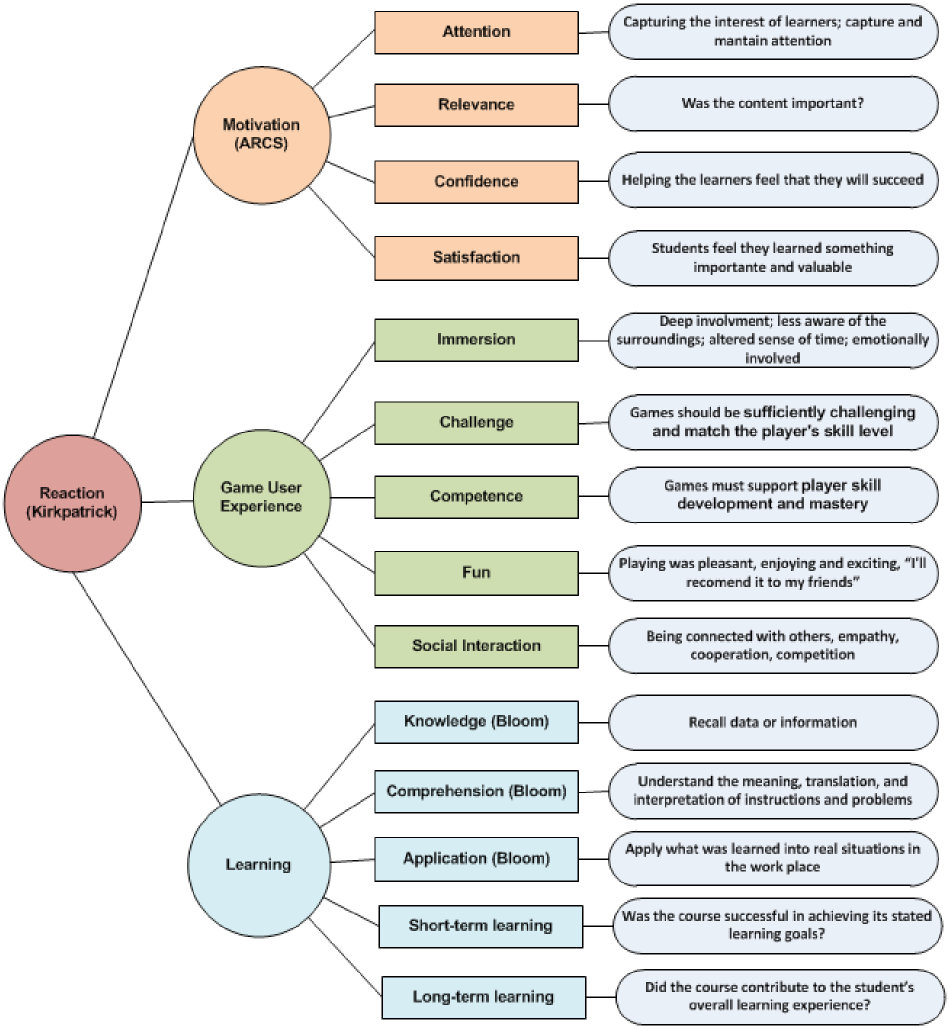

Reaction (Kirkpatrick)
Motivation (ARCS)
Attention
Capturing the interest of learners; capture and mantain attention
Relevance
Was the content important?
Confidence
Helping the learners feel that they will succeed
Satisfaction
Students feel they learned something importante and valuable
Game User Experience
Immersion
Deep involvment; less aware of the surroundings; altered sense of time; emotionally involved
Challenge
Games should be sufficiently challenging and match the player's skill level
Competence
Games must support player skill development and mastery
Fun
Playing was pleasant, enjoying and exciting, "I'll recomend it to my friends"
Social Interaction
Being connected with others, empathy, cooperation, competition
Learning
Knowledge (Bloom)
Recall data or information
Comprehension (Bloom)
Understand the meaning, translation, and interpretation of instructions and problems
Application (Bloom)
Apply what was learned into real situations in the work place
Short-term learning
Was the course successful in achieving its stated learning goals?
Long-term learning
Did the course contribute to the student's overall learning experience?

Fig. 8. Overview of dimensions and measurement items based on the MEEGA model (Savi *et al.*, 2011).

6.2. *Analysis and Results*

In order to obtain greater precision and statistical relevance, increasing the sample size, the data collected from the 23 applications were analyzed cumulatively. The grouping of the data was possible due to the similarity and standardization of the data collected. In this series of case studies, the data are similar in terms of definition (evaluating an educational game in relation to motivation, user experience and learning), research design (case study) and context (undergraduate computing courses). All data collected are standardized with respect to the measures (Fig. 7), the data collection instrument using the MEEGA questionnaire (Savi *et al.*, 2011), and the response format (using a 5-point *Likert* scale and interval scale).

In accordance with the MEEGA model, the collected data has been analyzed using descriptive statistics with respect to each of the defined analysis questions.

AQ1. Do the games motivate students?

In general, students recognized positively the motivation provided by the games, as shown in Fig. 9. The majority of the students was satisfied as the games allowed them to apply the knowledge taught beforehand, providing an alternative way to learn the application of the sorting algorithms. A large majority of the students felt confident that they were learning as they played the games. We also observed that students that had obtained already a better understanding helped their colleagues in a way of peer instruction. Most students considered the contents of the games relevant within the course context. The large majority of the students also considered the design of games attractive, contributing positively to the increase of their interests.

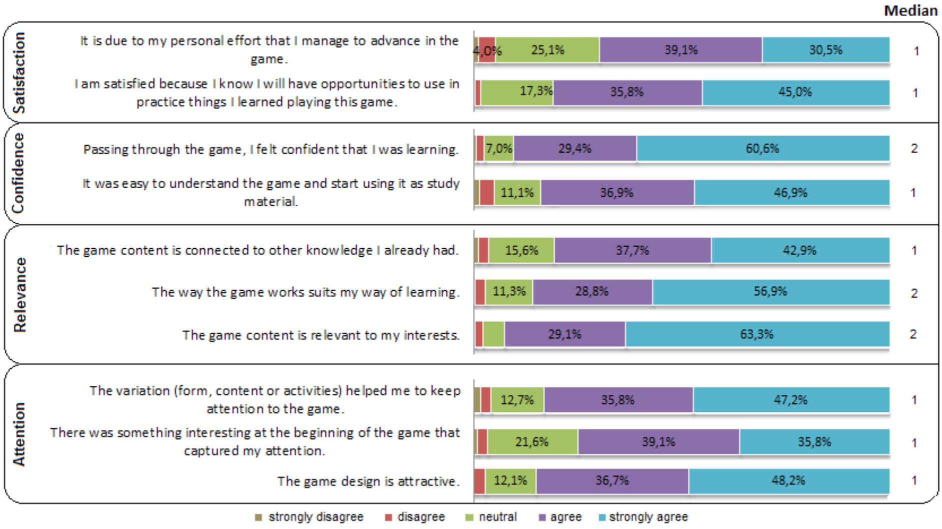


Fig. 9. Evaluation of motivation (total number of responses: 371).

AQ2. Do the games provide a positive user experience?

In general, the students evaluated the dimensions competence, fun and social interaction very positively (Fig. 10). Most students are confident that they are learning in an efficient manner with the games. In general, the students had fun while playing, creating a more relaxed setting during class. One of the most positively evaluated dimensions is the social interaction promoted by the games, whether competitively or collaboratively. Often students helped their colleagues contributing positively to their learning, whereas on the other hand, the competition promoted by the games between groups kept the students focused on the learning task.

On the other side, the games were evaluated moderately with respect to immersion. As immersion is important in order to keep the students focused, it would be important to improve this aspect of the games, for example by stimulating further the competition or create engaging narratives.

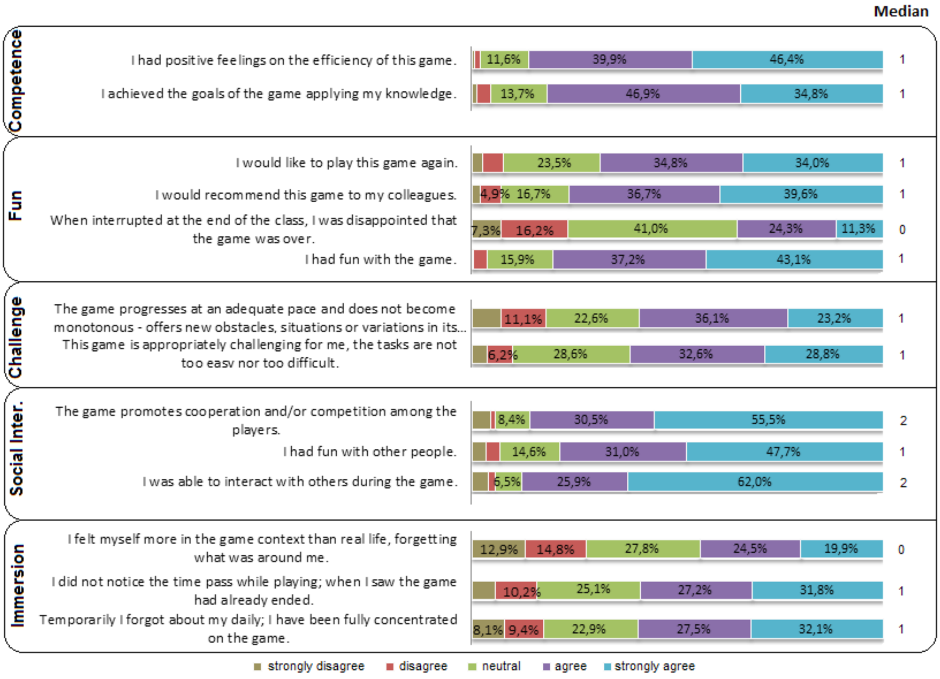


Fig. 10. Evaluation of user experience (total number of responses: 371).

AQ3. Do the games contribute to learning?

In general, the games seem to contribute positively to the student's learning as shown in Fig. 11. The students affirm that the games are effective instructional strategies for learning also in comparison to other instructional strategies used in the course. The stu-

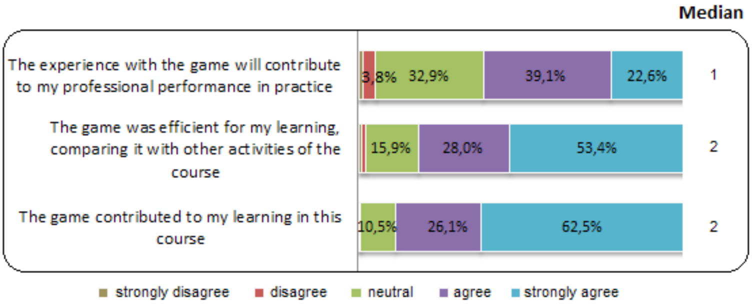


Fig. 11. Evaluation of learning (total number of responses: 371).

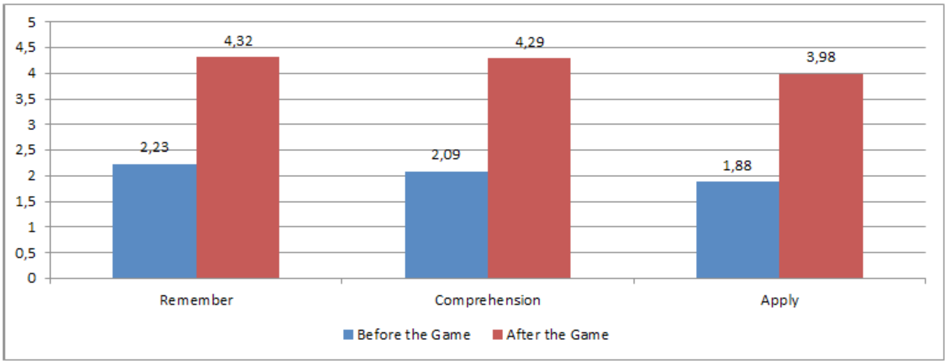


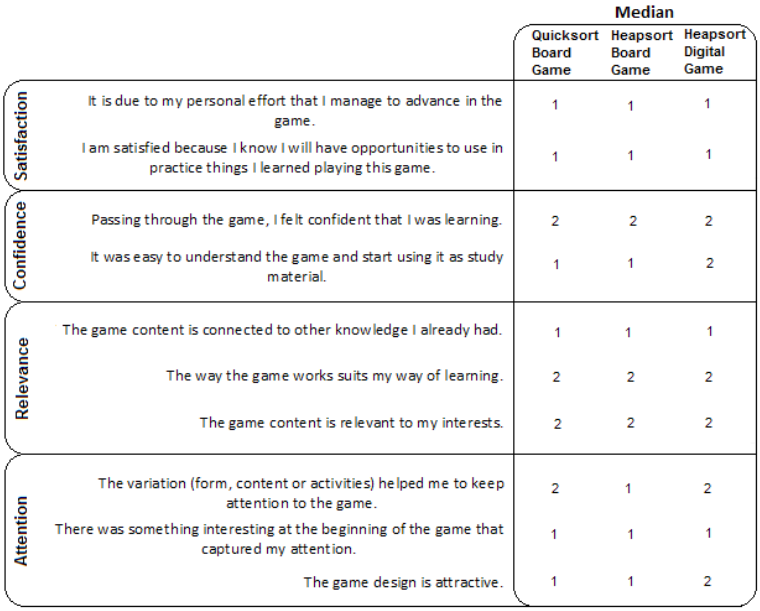
Fig. 12. Auto-assessment of the achievement of learning objectives (total number of responses: 371).

dents assert that the games lead them to learn on cognitive levels of knowledge, understanding, and application, as presented in Fig. 12.

AQ4. What are differences in the quality of the Quicksort Board Game, Heapsort Board Game and Heapsort Digital Game in relation to motivation, user experience and learning?

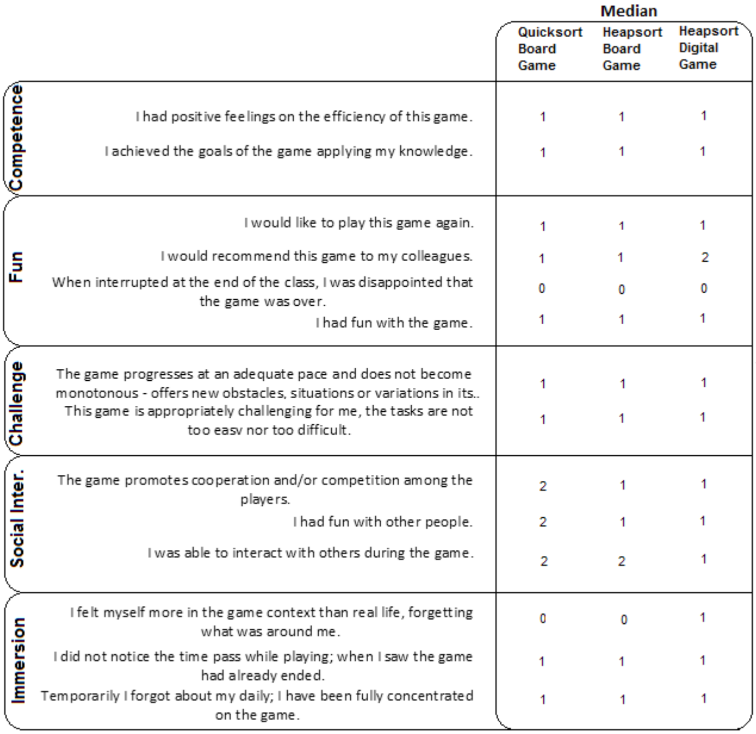
Comparing the evaluation of motivation provided by each game, we can observe that the results were positive in relation to satisfaction, confidence, relevance and attention with respect to all three games (Fig. 13). It is noteworthy, that there have been no evaluations with negative or neutral median for any of the games. In general, the evaluations among the three games are rather consistent. However, the digital game seems to be slightly better evaluated with respect to the ease to play the game as well as the attractiveness of its design. In the digital game, the information the names of pointers ("Father" or "Child") seems to be clearer that board games. Thus, allowing students to understand more easily the features of the game.

Comparing the evaluation of the games with respect to user experience a similar positive result can be observed as presented in Fig. 14. Exceptions are items related to immersion as well as the students' disappointment when the class ended (neutral me-



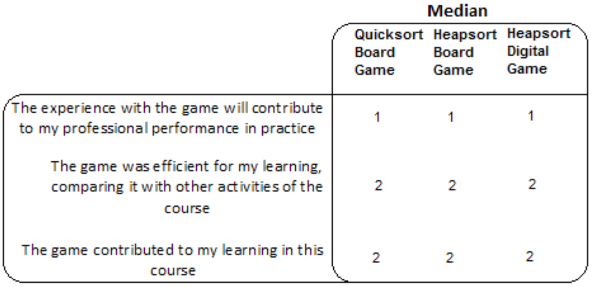
		Median		
		Quicksort Board Game	Heapsort Board Game	Heapsort Digital Game
Satisfaction	It is due to my personal effort that I manage to advance in the game.	1	1	1
	I am satisfied because I know I will have opportunities to use in practice things I learned playing this game.	1	1	1
Confidence	Passing through the game, I felt confident that I was learning.	2	2	2
	It was easy to understand the game and start using it as study material.	1	1	2
Relevance	The game content is connected to other knowledge I already had.	1	1	1
	The way the game works suits my way of learning.	2	2	2
	The game content is relevant to my interests.	2	2	2
Attention	The variation (form, content or activities) helped me to keep attention to the game.	2	1	2
	There was something interesting at the beginning of the game that captured my attention.	1	1	1
	The game design is attractive.	1	1	2

Fig. 13. Comparison of the medians of the evaluation of motivation (total number of responses: 371).



		Median		
		Quicksort Board Game	Heapsort Board Game	Heapsort Digital Game
Competence	I had positive feelings on the efficiency of this game.	1	1	1
	I achieved the goals of the game applying my knowledge.	1	1	1
Fun	I would like to play this game again.	1	1	1
	I would recommend this game to my colleagues.	1	1	2
	When interrupted at the end of the class, I was disappointed that the game was over.	0	0	0
	I had fun with the game.	1	1	1
Challenge	The game progresses at an adequate pace and does not become monotonous - offers new obstacles, situations or variations in its..	1	1	1
	This game is appropriately challenging for me, the tasks are not too easv nor too difficult.	1	1	1
Social Inter.	The game promotes cooperation and/or competition among the players.	2	1	1
	I had fun with other people.	2	1	1
	I was able to interact with others during the game.	2	2	1
Immersion	I felt myself more in the game context than real life, forgetting what was around me.	0	0	1
	I did not notice the time pass while playing; when I saw the game had already ended.	1	1	1
	Temporarily I forgot about my daily; I have been fully concentrated on the game.	1	1	1

Fig. 14. Comparison of the medians with respect to the evaluation of user experience (total number of responses: 371).



	Median		
	Quicksort Board Game	Heapsort Board Game	Heapsort Digital Game
The experience with the game will contribute to my professional performance in practice	1	1	1
The game was efficient for my learning, comparing it with other activities of the course	2	2	2
The game contributed to my learning in this course	2	2	2

Fig. 15. Comparison of the medians with respect to the evaluation of learning (total number of responses: 371).

dians). With respect to user experience, the digital game has only been slightly better evaluated than the non-digital games with respect to being recommended to colleagues. On the other hand, the *Quicksort Board Game* has been highlighted with respect to its very positive evaluation of social interaction.

Comparing the evaluation of the contribution to the learning of each of the games, we can observe a complete consistent result across the three games.

7. Discussion

The results obtained from the evaluation of the games are quite positive. Students noted that the games helped to reinforce the learning of these sorting algorithms in an engaging way, as well as motivating them. Overall, the quality dimensions of the games were highly rated, with medians varying between 1 and 2. In general, students actively participated in the games applications, playing the games with ease and having fun. Especially, the social interaction promoted by the games was commented very positively by students. Even in case of the digital single-player game, social interaction was evaluated positively as students played the game in pairs (two students per computer) and the promotion of competition through the ranking between students.

Students considered the games relevant within the context of the data structures course and being connected to the knowledge they already acquired. They also expressed that they consider the games important to their personal interests. We observed that this instructional strategy kept the attention of the students focused, especially in the beginning, when they concentrated on understanding the application of the sorting algorithm. When compared to other instructional strategies, games seem to promote competition and social interaction in an intrinsic manner contributing to keep the students easily engaged in the learning task. Another aspect highlighted by the students was the fun they had while learning sorting algorithms creating at the same time a more relaxed learning environment.

On the other side, one of the dimensions less positively evaluated was immersion. Thus, an improvement opportunity can be the creation of new challenges that lead students to greater competitiveness and rewards for the winners and/or the integration of an engaging narrative.

Comparing the three games, no significant differences have been observed neither in terms of the sorting algorithm nor the type of game (digital vs. non-digital games).

Based on student evaluations, we can observe that with the games students increased their knowledge on sorting algorithms contributing positively to the achievement of the learning objectives. The games were assessed positively as instructional strategies by the students also in comparison to other strategies used in the discipline.

Threats to Validity. The results obtained in this study need to be interpreted with caution, taking into account potential threats to their validity. One threat may be the research design adopted conducting a series of case studies (*one-shot post-test only*). Due to the lack of a pre-test measurement, learning achievements cannot be compared to the acquired knowledge before the application of the games. This threat is further compounded by the fact that there is no control group to compare the identified effects. However, in this research, we chose to conduct a case study rather than a formal experiment in order to minimize interruptions in the normal course flow. This may reduce the validity of the results obtained, yet due to practical reasons was the only possible way to perform the research.

Another threat to validity is related to the sample size, which may compromise the generalizability of the results. To minimize this problem, we conducted a large-scale series of case studies being conducted over 4.5 years, with the total participation of 371 students representing a much larger sample than other evaluations of similar games encountered in literature.

There may also be threats to construct validity. Due to practical limitations, running the study as part of a class, the results related to learning effects were obtained from self-assessments of the students. This type of assessment may not be enough to measure students' knowledge, however, there is evidence that self-assessments results provide reliable, valid and useful results (Topping, 2003; Falchikov and Boud, 1989). Considering the complexity of an assessment of educational games, the evaluation of the perception of students is a simple, fast and less invasive approach for feedback.

Another possible threat is that the quality aspects of the game are difficult to measure. This fact is mainly due to its high degree of subjectivity. To counteract this threat to validity, a standarized questionnaire has been used based on a commonly evaluation model (MEEGA) that has be evaluated with respect to its validity and reliability (Savi *et al.*, 2011).

8. Conclusion

Motivated by the lack of educational games to teach the sorting algorithms *Quicksort* e *Heapsort*, we systematically developed three games *Quicksort Board Game*, *Heapsort Board Game* and *Heapsort Digital Game* in order to support the teaching of quasilinear sorting algorithms with logarithmic complexity. The games were applied in 23 data structures courses at the undergraduate computing course at the Federal University of Santa Catarina with a total participation of 371 students. Results indicate that the games allow students to achieve the learning objectives of the instructional unit as well as to motivate them and to provide a positive learner experience. Students considered the games relevant within the course context allowing to reinforce the learning of these sorting algorithms on the cognitive levels of comprehension and application. Furthermore, students recognized that they had fun by playing the games enjoying the social interaction through cooperation or competition among colleagues. In general, based on the positive evaluation of the students, the games were integrated permanently in the data structures course as a complementary instructional method to support the teaching of the algorithms *Quicksort* and *Heapsort.*

Acknowledgments

This work is supported by CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico*) and CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*), entities of the Brazilian government focused on scientific and technological development.

We would also like to thank all students for their valuable feedback.

References

Abt, C.C. (2002). *Educational Games.* University Press of America.

ACM/IEEE-CS. (2013). Computer Science Curricula 2013. *ACM e IEEE Computer Society*, 1–514.

ACM/IEEE-CS. (2005). Computing Curricula 2005. *ACM e IEEE Computer Society*. 2005, 1–56.

Adams, E. and Rollings A. (2006). *Fundamentals of Game Design, Prentice Hall.* Prentice Hall.

Bai, Y., Ahmed, S.Z., and Granado, B. (2014). Fast and power efficient heapsort IP for image compression application. In: *Proc. of the IEEE 22nd International Symposium on Field-Programmable Custom Computing Machines*. Boston, USA, 237–237.

Basili, V.R., Caldeira, G., and Rombach, H.D. (1994). Goal Question Metric Paradigm. *Encyclopedia of Software Engineering.*John Wiley and Sons.

Battistella, P., Gresse von Wangenheim, C. (2016a). Games for Teaching Computing in Higher Education – A Systematic Review. *IEEE Technology and Engineering Education*, 1(3), 8–30.

Battistella, P., Gresse von Wangenheim, C. (2016b). Characterization of the Target Audience Educational Games in computer area. In: *Workshop on Computer Education*, Porto Alegre, Brazil.

Bloom, B.S. (1956). Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I. Cognitive Domain, Longmans.

Boticki, I, Barisic, A., Martin, S., Drljevic, N. (2012). Sortko: Learning Sorting Algorithms with Mobile Devices. *IEEE International Conference on Wireless, Mobile and Ubiquitous Technology in Education.* (pp. 49–56). Takamatsu, Japan.

Boyle, E.A., Hainey, T., Conn, T., and Riberio, C. (2016). An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and Educational Games. *Computers & Education*. 94, 178–192.

Branch, R.M. (2009). Instructional Design: The ADDIE Approach. *Springer*, New York, USA.

Budd, T.A. (2006). An Active Learning Approach to Teaching the Data Structures Course. *Proc. of the Technical Symposium on Computer Science Education*. (pp. 143–147). New York, USA.

Calderón, A., Ruiz, M. (2015). A systematic literature review on Educational Games evaluation: An application to software project management. *Computers & Education*, 87, 396–422.

Caulfiled, C., Vael, D. e Maj, S.P. (2011). Teaching Software Engineering Project Management – a novel approach for Software Engineering programs. *Modern Applied Science*, 5(5), 28-43.

Cesare, R.V. (2012). Learn Bubble and Bucket Sorting Algorithms by playing a 2D Digital Game. Sorting Algorithms: Digital Game Based Learning. Available at: `http://sortingalgorithmsgame.webs.com`

Chen, D., Fu, N. (2010). On Computer Science Reform of Curriculum Structure. In: *Proc. of the International Symposium on Electronic Commerce and Security.* (pp. 21–24). Guangzhou, China.

Choi, J., Hannafin, M. (1995). Situated Cognition and Learning Environments: Roles, Structures and Implications for Design. *Educational Technology Research and Development.* 43(2), 53–69.

Computer Science Unplugged. (2005). Lightest and Heaviest – Sorting Algorithms. Available at: `http://computingunplugged.org/sorting-algorithms`

Connolly, T.M., Boyle, E.A., MacArthur, E., Hainey, T., Boyle, J.M. (2012). A systematic literature review of empirical evidence on computer games and serious games. *Computers & Education,* 59(2), 661–686.

Connolly, T.M., Stansfield, M., Hainey, T. (2007). An application of games based learning within software engineering. *British Journal of Educational Technology*, 38(3), 416–428.

Dempsey, J.V., Lucassen, B., Rasmussen, K. (1996). *The Instructional Gaming Literature: Implications and 99 Sources*. Technical Report 96-1. College of Education, University of South Alabama, Molibe, USA.

Denton, L.D., McKinney, D., Doran, M.V. (2005). A melding of educational strategies to enhance the introductory programming course. In: *Conf. ASEE/IEEE Frontiers in Education.* Indianapolis, USA, 7–12.

Ernst, D.J., Stevenson, D.E., Wagner, P. (2009). Hybrid and custom data structures: evolution of the data structures course. In: *Conf. on Innovation and Technology in Computer Science Education*. Paris, France, 213–217.

Falchikov, N., Boud, D. (1989). Student self-assessment in higher education: a meta-analysis. *Review of Educational Research*, 59(4), 395–430.

Freeman, S., Eddy, S.L., McDonough, M., Smith, M.K., Okoroafor, N., Jordt, H., Wenderoth, M.P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proc. of the National Academy of Sciences*, 111(23), 8410-8415.

Fullerton, T. (2008). *Game Design Workshop: A Playcentric Approach to Creating Innovatives Games*. 3rd. ed. Burlington: Morgan Kaufmann.

Gresse von Wangenheim, C., Savi, R., Borgatto, A.F. (2012). DELIVER! – An educational game for teaching Earned Value Management in computing courses. *Information and Software Technology*, 54(3), 286–298.

Hakulinen, L. (2012). DSAsketch: Data structures and algorithms drawing game. In: *Proc. of the Koli Calling Int. Conf. on Computing Education Research*. Tahko, Finland, 139–140.

Hakulinen, L. (2011). Card games for teaching data structures and algorithms. In: *Proc. of the Koli Calling Int. Conf. on Computing Education Research*. Koli, Finland, 120–121.

Hamilton, J., Tee, S. (2010). Smart utilization of tertiary instructional modes. *Computers & Amp, Education*, 54(4), 1036–1053.

Herz, J.C. (1997). *Joystick Nation : How Videogames ate our Quarters, won our Hearts, and Rewired our Minds*. Little, Brown and Company, Boston, USA.

Hurwitz, A., Day, M. (2007). *Children and their Art: Methods for the Elementary School.* Eighth Edition. Belmont: Thomson Wadsworth, USA.

Lawrence, R. (2004). Teaching data structures using competitive games. *IEEE Transactions on Education*. 47(4), 459–466.

Munro, J.I., Ji, X.R. (1996). On the Pivot Strategy of Quicksort. Conference on Electrical and Computer Engineering. Calgary, Alta, Canada.

Qian, M., Clark, K.R. (2016). Game-based learning and 21st century skills: A review of recent research. *Journal Computers in Human Behavior*, 63, 50–58.

Padrón, C., Díaz, P., Aedo, I. (2007). Towards an Effective Evaluation Framework for IMS LD-Based Didactic Materials: Criteria and Measures. Human-Computer Interaction. *HCI Applications and Services.* Beijing, China, 312–321.

Petri, G., Gresse von Wangenheim, C. (2016). How to evaluate educational games: a systematic literature review. *Journal of Universal Computers Science*, 22(7), 992–1021.

Poulova, P., Klimova, B. (2015). Education in Computational Sciences. International Conference on Computational Science. *Procedia Computer Science*, 51, 1996–2005.

Prifti, V., Bala, R., Tafa, I., Saatciu, T., Fejzaj, J. (2015). The time profit obtained by parallelization of quicksort algorithm used for numerical sorting. In: *Conf. of Science and Information*. London, UK, 897–901.

Reilly, E.D. (2003). *Milestones in Computer Science and Information Technology.* 1ª Edition, Greenwood Press, Westport, Connecticut, USA.

Sane, S.S., Deshpande, N.A. (2006). *Data Structures & Algorithms*. 1ª Edition, Technical Publications Pane.

Savi, R., Gresse von Wangenheim, C., Borgatto, A.F. (2011). A model for the evaluation of educational games for teaching software engineering. In: *Conf. of Brazilian Symposium on Software Engineering*. São Paulo, Brazil, 194–203.

Savi, R., Ulbricht, V.R. (2008). Educational games: benefits and challenges. *Magazine New Technologies in Education* – Renote 6. (in Portuguese).

Schell, J. (2008). *The Art of Game Design: A Book of Lenses.* 1ª ed. Morgan Kaufmann: CRC Press.

Shabanah, S.S., Chen, J., Wechsler, H., Carr, D., Wegman, E. (2010). Designing computer games to teach algorithms. In: *International Conference on Information Technology*. Las Vegas, USA, 1119–1126.

Stone, H.S. (1978). Sorting on star. *IEEE Transactions on Software Engineering*, 4(2), 138–146.

Tan, B., Seng, J.L.K. (2010). Game-based learning for data structures: a case study. In: *Int. Conf. on Computer Engineering and Technology*. Chengdy, China, 718–721.

Topping, K.J. (2003). Self and peer assessment in school and university: reliability, validity and utility. In: M.S.R. Segers, F.J.R.C. Dochy, E.C. Cascallar (Eds.), *Optimizing New Modes of Assessment: In Search of Qualities and Standards*, Kluwer Academic Publishers.

Wegner, L.M., Teuhola, J.I. (1989). The external heapsort. *IEEE Transactions on Software Engineering.* 15(7), 917–925.

Wohlin, C., Runeson, P., Host, M, Ohlsson, M.C., Regnell, B., Wesslen, A. (2012). *Experimentation in Software Engineering*. Springer Verlag.

Wolf, M.J. (2001). *The Medium of the Video Game*. University of Texas Press, Austin, USA.

Yin, R.K. (2013). *Case Study Research: Design and Methods*. 5. Edition, SAGE Publications.

P.E. Battistella, PhD in Computer Sciences from the Department of Informatics and Statistics, of the Federal University of Santa Catarina, and researcher of Software Quality Group – GQS, at the National Institute for Research and Technology on Digital Convergence – INCoD, Florianópolis, Santa Catarina, Brazil.

C.G. von Wangenheim, Prof. Dr. rer.nat., professor at the Department of Informatics and Statistics of the Federal University of Santa Catarina, Florianópolis, Santa Catarina, Brazil, where she also coordinates the Software Quality Group – GQS focusing on scientific research, development and transfer of software engineering models, methods and tools and software engineering education in order to support the improvement of software quality and productivity. She also coordinates the initiative *Computação na Escola*, which aims at bringing computing education to schools in Brazil.

A. von Wangenheim, Prof. Dr. rer.nat., professor at the Department of Informatics and Statistics, of the Federal University of Santa Catarina. He coordinates the National Institute for Research and Technology on Digital Convergence – INCoD. His experience is in the areas of Content Production for Interactive Digital TV, Computer Vision/Image Analysis and Processing and Health Informatics, focusing on Telemedicine, Teleradiology, Support Systems for Image Diagnosis and Medical Image Processing, mainly on: Intelligent and High-Performance Image Analysis, DICOM, Content-Based Image Retrieval – CBIR, Computer Vision and PACS.

J.E. Martina, Prof. PhD, professor at the Department of Informatics and Statistics of the Federal University of Santa Catarina, and research in the Computer Security Laboratory – LabSec. He is currently teaching Data Structures for the degrees in Computer Science and Information Systems and working on authentication protocols for telemedicine environments and on the formalization of security ceremonies. He is also a visiting lecturer in Computer Science at University of Hertfordshire working mainly as project supervisor for B.Sc. and M.Sc degrees.