UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

The impact of tinkering and planning behaviors on learning introductory programming concepts

Fernando Henrique Carvalho Silva

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

Fernando Henrique Carvalho Silva

The impact of tinkering and planning behaviors on learning introductory programming concepts

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Seiji Isotani

USP – São Carlos
February 2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Carvalho Silva, Fernando Henrique
CS586t The impact of tinkering and planning behaviors on learning introductory programming concepts / Fernando Henrique Carvalho Silva; orientador Seiji Isotani. -- São Carlos, 2020.
89 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2020.

1. Ciência da Computação. 2. Informática na educação. 3. Resolução de problemas. 4. Mineração de dados. 5. Tinkering. I. Isotani, Seiji, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Fernando Henrique Carvalho Silva

O impacto de comportamentos tinkering e planejamento no aprendizado de conceitos introdutórios de programação

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Seiji Isotani

USP – São Carlos
Fevereiro de 2020

ACKNOWLEDGEMENTS

To my family that always supported me. To my laboratory colleagues that provided valuable discussions and feedback, with special thank to Armando M. Toda and Kamila T. Lyra for their patience and guidance during this time. To my Advisor, Seiji Isotani, for teaching me a lot about the academy and research. To my closest friend Fernando Neto, that helped me more times that I can count. To my partner, Lara, that is there in the best and worst times, and without her, I wouldn't be where I'm today.

Also, I would like to thank CAPES for the financial support, also thank the University of São Paulo and the Instituto de Ciências Matemáticas e de Computação (ICMC) for their support.

RESUMO

SILVA, F. H. C. **O impacto de comportamentos tinkering e planejamento no aprendizado de conceitos introdutórios de programação**. 2020. 86 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

As habilidades de programação estão se tornando cada vez mais essenciais nas mais diferentes áreas de atuação e disciplinas. Como parte integrante do currículo de formação básica em ciência da computação, essas habilidades são consideradas atividades de solução de problemas. A solução de problemas é um processo cognitivo interno de alta ordem que procura uma solução para um problema ou um caminho para um determinado objetivo. Como um processo interno, as estratégias de solução de problemas não podem ser observadas diretamente e, portanto, devem ser inferidas. Essa necessidade levou vários pesquisadores a identificar e classificar essas estratégias com base no comportamento dos alunos. Entre essas pesquisas, a classificação entre exploradores e planejadores, do inglês *tinkerers* e *planners*, proposta por Turkle e Papert foi utilizada neste trabalho. Nessa classificação, os planejadores são estudantes que apresentam uma estratégia estruturada, *top-down*, para resolver problemas, que é geralmente vista em ambientes de ciência da computação. Por outro lado, os eploradores empregam uma estratégia alternativa, menos estruturada e de *bottom-up*, que usa as informações apresentadas pelo sistemas para atingir seus objetivos. De modo a investigar os comportamentos de planejadores e tinkerer em ambientes online e seu impacto na performance das notas dos estudantes, primeiro foi empregada uma abordagem de mineração de dados educacionais usando agrupamento para identificar os comportamentos dos alunos e os recursos relacionados ao tinkering, em seguida, foi realizada uma avaliação das notas dos alunos e o tempo que estes necessitaram para concluir as atividades online. Como resultado, foram identificados quatro comportamentos relacionados à classificação de Turkle e Papert. A análise estatística dos dados relacionados à estes comportamentos mostraram que os alunos que empregavam uma quantidade maior de tinkering, apresentaram também desempenho superior nos testes finais, além de precisarem de mais tempo para concluir suas atividades on-line. Esses resultados indicam que tinkering é uma estratégia válida de solução de problemas com potencial deve explorado pelos educadores.

Palavras-chave: Resolução de problemas, sciência da computação, mineração de dados, agrupamento, informática na educação, tinkering.

ABSTRACT

SILVA, F. H. C. **The impact of tinkering and planning behaviors on learning introductory programming concepts**. 2020. 86 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

Computer programming skills are becoming essential across different fields and disciplines, and as integrating part of computer science basic formation curriculum, these skills are considered problem-solving activities. Problem-solving is an internal cognitive process of a high order that searches for a solution for a problem or a path to a given goal. As an internal process, problem-solving strategies cannot be observed directly and must be inferred. That necessity led several researchers to identify and classify those strategies based on students' behaviors. Among those researches, the classification between tinkerers and planners proposed by Turkle and Papert was used in this work. In this classification, planners are students who present a structured, top-down strategy to solve problems, which is commonly reinforced in computer science environments. On the other hand, tinkerers employ an alternative, less structured, bottom-up problem-solving strategy that uses the system's feedback to achieve their goals. To Investigate the students' tinkerer and planner behaviors in online environments and its impact on the students' grades performance, firstly an educational data mining approach using clustering was employed to identify the students' behaviors and the features related to tinkering, followed by an assessment of the students' grades and the time that they required to finish the assignments. As a result, four behaviors were identified related to Turkle and Papert classification. Statistical tests showed that students that employed a higher amount of tinkering presented a better performance in final tests and required more time to finish their online assignments. These results indicate that tinkering is a valid problem-solving strategy with potential that worth being explored by educators.

Keywords: problem-solving, computer science, data mining, clustering, computer science in education, tinkering.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

CHAPTER

1

INTRODUCTION

1.1 Contextualization and Motivation

With the ubiquitous presence of technology in almost all aspects of everyday life, professionals with the required computer programming skills are becoming essential across different fields and disciplines (FEDORENKO *et al.*, 2019).

According to the curriculum guidelines from the Brazilian Ministry of Education (MEC) (MEC, 2016), programming subjects are integrating part of the computer science basic formation curriculum. The curriculum comprises, amongst other topics, programming languages, programming paradigms, data structures and algorithms, and are described as a problem-solving activities (SANTOS; COSTA, 2006). The Brazilian Computer Society lists amongst the skills expected of a student graduated in computer science the ability to "employ concepts, techniques and computational tools to identify and analyze problems from every day, social and all areas of knowledge, model and solve them, individually and/or cooperatively, using languages and representations adequate to describe processes (algorithms) and information (data), validating strategies and results" (SBC, 2017)

A problem-solving process can be defined as a high order cognitive process and one of the most complex cognitive activities (GOLDSTEIN; LEVIN, 1987). Researches as ROMEIKE (2008) and MACKIE-MASON; GROTH (2010) defines the problem solving as the core skills of computer science, where all the problems that can be computationally addressed can be considered a problem.

Problem-solving is a cognitive process of the brain at a higher cognitive layer that searches for a solution for a given problem or finds a path to reach a given goal (WANG; CHIEW, 2010). This process occurs internally in the problem solver, and therefore, the strategies employed to solve problems cannot be directly observed and must be inferred (SHARMA *et al.*, 2018).

Even though it has been acknowledged that programming is more than the ability to write and run lines of code and that problem-solving skills plays an essential role in several activities, including programming, in the computer science education the traditional outcome paper-based assessment it's still the predominant model to evaluate students' performance. However, in this outcome-based scenario, the development and application of students' problem-solving skills are usually neglected or limited to the current top-down problem-solving approaches (KIESMÜLLER, 2009), without taking in account the impact that may have on the students' learning. This leads to a divergence between the student's problem-solving skills and their ability to employ those skills to solve real-world problems (SHARMA *et al.*, 2018).

To reduce this divergence in the students' problem-solving skills, it's essential to educators to acknowledge the different problem-solving strategies that students may employ in programming subjects and how they can be applied in computer science education, as well as the needs that students may present to develop new problem-solving skills. Therefore, in addition to the appropriated programming paradigms, environment, and tools, some authors argue that educators should consider teaching and developing students' problem-solving strategies (SHARMA *et al.*, 2018).

The necessity to infer students' strategies and to acknowledge and explore the role of problem-solving skills in computer science education led several researchers to attempt to identify and classify students' employed problem solving by their behaviors expressed in programming assignments (PERKINS *et al.*, 1986; BRUCE *et al.*, 2004; BLIKSTEIN *et al.*, 2014; SHARMA *et al.*, 2018). Amongst these attempts, the most well-received classification of students' by their employed problem-solving strategies is proposed by TURKLE; PAPERT (1992), where students are classified between *bricoleurs* or, as more used in the literature, tinkerers (BLIKSTEIN *et al.*, 2014; PEREZ; RICHARDSON; ROSENBLUM, 2017), and planners, depending on the problem-solving approach employed (BLIKSTEIN *et al.*, 2014). In this classification, planners are students that present a structured, top-down strategy to solve problems, while tinkerers employ an alternative, less structured, bottom-up problem-solving strategy that uses system's feedback to achieve their goals. Even though those are opposites problem-solving strategies, TURKLE; PAPERT (1992) argues that both are equally valid problem-solving strategies and that students that employ those strategies present similar performance in programming tasks.

However, along the same lines of thought followed by other studies (PERKINS *et al.*, 1986; TURKLE; PAPERT, 1992; BRUCE *et al.*, 2004; BLIKSTEIN *et al.*, 2014), to develop the students' problem-solving skills and employed strategies, further research is needed to understand and validate those strategies if used to reduce the divergence between students' problem-solving skills.

In this context, as a means to contribute to the development of students' problem-solving skills and employed strategies, this work seeks to understand the impact of the students' problem-solving behaviors and its impact on the students' grades performance during online assignments.

Section 1.2 describes a list of objectives to help us achieve this goal.

1.2 Objectives

1.2.1 Main Objective

Investigate the students' tinkerer and planner behaviors in online environments and its impact on the students' grades performance.

1.2.2 Specific Objectives

To achieve the main objective of this work, the following specific objectives were established:

1. **Identify students' assignments features related to the tinkering problem-solving strategy**

   This specific objective focuses on identifying a set of features related to Turkle and Papert's definition of tinkerers and planners. It allows the application of educational data mining techniques to identify the occurrence of students' problem-solving behaviors and their characteristics. This objective also contributes to a better definition of planners and tinkerers through the identified features (step 1 of Figure 1).

2. **Identify students' tinkering problem-solving behaviors and its relation to the Turkle and Papert's definition of Tinkerers and Planners**

   Based on the selected educational data mining approach, it will be performed identification of the students' problem-solving behaviors (step 2 of the Figure 1), as well a discussion about its relation with planners and tinkerers definition proposed by Turkle and Papert (step 3 of the Figure 1).

3. **Assess the impact of the students' tinkering problem-solving behaviors in their grades performance**

   Lastly, assess the impact of the students' behaviors identified in this work through the students' grades performance in different assignments during the observed time.

1.3 Dissertation Organization

The remainder of this dissertation is organized as follows: in Chapter 2 discusses the main concepts pertinent to the understanding of the text, such as problem-solving and data mining concepts, along with the literature related to the presented work; in Chapter 3 discusses in detail the research questions, hypothesis formulation and methodology, as well the process of

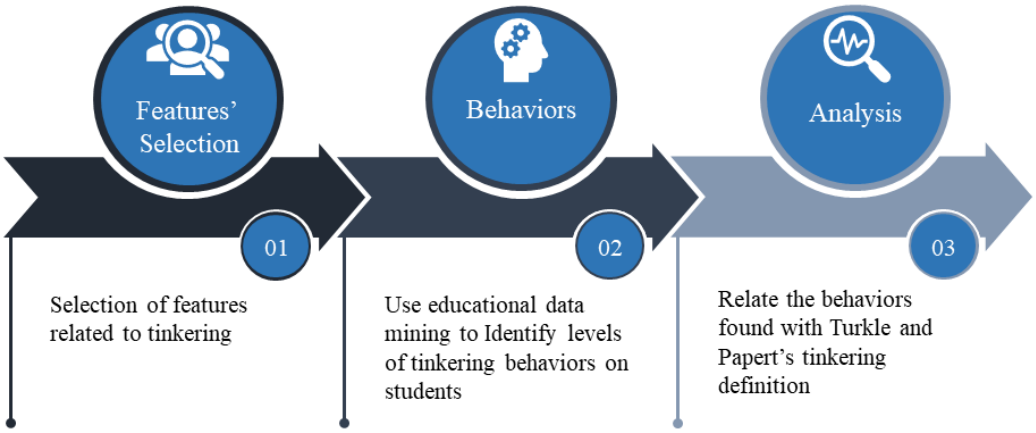


Figure 1 – Process using Specific Objective 1, 2 and 3

Source: Elaborated by the author

data selection and preprocessing techniques, methods and tools used in this work; in Chapter 4 the data mining results, hypothesis tests and discussion of the results obtained are presented; and lastly, in Chapter 5 the final remarks, a summary of the work, its contributions and perspectives for future work are discussed.

CHAPTER

2

BACKGROUND

Based on the literature review conduct during the development of this work, this chapter presents the main concepts of this work that are the base for a better understanding of the presented research. Section 2.1 discuss different learning strategies employed by students during the learning process and where tinkering lies in this context. Section 2.2 presents the basic Data Mining concepts, its use in an educational environment, as well as the algorithms and methods relevant to this work.

2.1 problem-solving Strategies

Problem-solving is one of the main skills among the expected competencies of a computer science graduate (SBC, 2017; MEC, 2016) and is considered by several authors as the core skills of computer science (ROMEIKE, 2008; MACKIE-MASON; GROTH, 2010).

According to POLYA (2004) and ORMROD (2011) a problem is composed by three parts: i) The information available relative to the problem; ii) a final state to the problem, also known as goal or solution; iii) A set of possible actions that can be performed to achieve the problem's goal.

A similar definition is proposed by NEWELL; SIMON *et al.* (1972) that divides a problem into two parts: i) A description of the problem space that contains all the possible states of the problem and the problem solver, and ii) A set of paths through the problem space's states.

The strategies that the problem solver can employ in order to reduce the difference between the current problem state and the goal state are known as problem-solving strategies.

REISBERG; MAYER (2013) defines problem-solving as a *"cognitive processing directed at achieving a goal when the problem solver does not initially know a solution method"*. The following aspects of this process worth to be highlighted: i) since it's a cognitive process it involves representing and manipulating knowledge in the problem solver cognitive system, which

can only be assessed based on the solver's behaviors; and ii) problem-solving is a directed and personal process, which means that the cognitive processing is focused by predefined goals, and its difficulty is affected by problem solver's prior knowledge, skills, and the problem's nature (MAYER; WITTROCK, 2006).

Problem-solving strategies are techniques that, even though it does not guarantee success in finding a solution, may aid the problem-solvers during the solving process (MAYER, 1992; GICK, 1986). Thus, choosing different problem-solving strategies may have different impacts on the students' learning process (SHARMA *et al.*, 2018). Past researches (LISHINSKI *et al.*, 2016) has shown that in a programming learning environment, the choice of different problem-solve strategies presented the highest correlation with students' performance in coding tasks amongst the assessed features.

Various approaches to problem-solving strategies have been investigated in different areas (e.g., psychology, cognitive informatics, and computational intelligence) (WANG; CHIEW, 2010) and the most common strategies are, inter alia, the following:

- **Hill climbing:** When using this strategy, the problem solver goal is to improve their problem's solution step by step. At each step, they try to find the optimal solution for the respective next step of the problem and repeating this process until they reach the problem's final solution (BARNES; FINCHER; THOMPSON, 1997).

- **Trial and error:** This strategy is preferred when the problem solver finds itself facing a complex problem and consists of finding the correct path through the problem space trying different possibilities one by one and evaluating their outcomes until they find a solution (CHI; GLASER, 1985).

- **Top-down:** This strategy uses an identification of all problem's states by the problem solver before finding the correct path through the problem space. This strategy is well known in computer science and software engineering (KIESMÜLLER, 2009).

- **Bottom-up:** This strategy focus on the solving of each problem's state presented to the problem solver before moving to the next state. A problem is considered solved after the last problem's state is solved (KIESMÜLLER, 2009).

As can be seen, some strategies may be similar to each other in some aspects (e.g., both trial and error, and bottom-up are similar unstructured strategies), but might have different impacts the students' problem-solving ability. Those particularities motivated numerous attempts to classify students based on their employed strategies. PERKINS *et al.* (1986) discuss different learning strategies and their importance for the pedagogy of programming. Using as examples the strategies employed by students using BASIC and LOGO programming languages, by structured observation, the authors classified students from elementary and High school in two categories

based on their attitude when facing a problem: *Stoppers*, the ones that once face a difficulty do not try to find a solution, and *movers*, students who try new ideas consistently, modifying and testing their code until they find a solution.

Conducted with thirteen (13) first-year undergraduate students, BRUCE *et al.* (2004) research used a phenomenographic approach to analyze the employed students' learning strategies in an introductory programming course. As a result, they categorized the students' attitudes during the process in five different categories: *Following* - students that experienced the process as a set of tasks to be done in tot their grades; *Coding* - students that presented this attitude were invested in learning the programming language syntax, sometimes showing some tinkering behavior in the process; *Understanding and Integrating* - these students seek to understand how the learned concepts integrate past tasks, building their knowledge at each step. Similar to *coders* - these students also presented some tinkering behavior, "fiddling" with the concepts; *problem-solving* - students focused on solving the problem instead of learning how to code, to these students coding is a tool that would be used to solve the problem. These categories students presented similar planning approach to the *planners* students proposed by TURKLE; PAPERT (1992); *Participating* - in this category, the students are motivated to learn proto program be a part of the programmers' community and interacting with it, seeing the act of coding as culture itself.

BLIKSTEIN *et al.* (2014) conducted a series of exploratory studies over two semesters with three hundred and seventy (370) undergraduate students from a university introductory programming course. Using machine learning methods, such as x-means clustering and dynamic time warp, to analyze and uncover possible hidden patterns in the students' data, the authors defined a set of interest variables based on *tinkerers/planners* behaviors proposed by TURKLE; PAPERT (1992) to represent the amount of tinkering in the students' coding assignments. The results however, presented no significant correlation between students' performance and their categories.

In a more recent work SHARMA *et al.* (2018), conduct a study with six hundred (600) students from a Java introductory programming university course to predict students' performance based on their behaviors during unity testing tasks during the semester. Using Generalized Additive Model, the authors identified three different categories: *Intellects* - students with high programming skills and more confidence in their abilities that run tests less frequently; *Thinkers* - students that seek system's feedback to progress in the tasks, therefore, run tests more frequently; and *Probers* - students that experience difficulties during the tasks, which leads them to run tests most frequently than the others.

Amongst several attempts to classify students by their problem-solving strategies, the most well-received is the framework proposed by TURKLE; PAPERT (1992), which labeled students based on the amounts of tinkering they express. The authors proposed the following categories: *Tinkerers* - students that presented a playful, non-hierarchical approach to the problem-

solving and a high number of interaction with the system (tinkering); and *Planners* - students that presented a structured, well-defined approach to problem-solving and low amount of interactions with the system (tinkering) (BLIKSTEIN *et al.*, 2014).

2.1.1 Tinkerers and Planners

The term *Tinkerer* coined by TURKLE; PAPERT (1992), has its roots in the LÉVI-STRAUSS *et al.* (1962), who used the french verb *"bricoleur"*, which could be translate as *"To tinker"*, to describe the process of creative work with endless possibilities (ROSE, 2016). In the work published by TURKLE; PAPERT (1992), the term tinkerer is used to describe an alternative bottom-up problem-solving approach based on the students' behavior. This approach is opposed to the structured, top-down, rule-driven approach of planners.

While *planners* value hierarchy and abstraction, *tinkerers* prefer to negotiate and rearrangement with the available materials and tools (TURKLE; PAPERT, 1992). Therefore, instead of planning and following steps to solve a problem, tinkerers choose and change their next steps along the problem-solving process as new information emerges (RESNICK; ROSENBAUM, 2013).

Some authors (PEREZ; RICHARDSON; ROSENBLUM, 2017) describe tinkering as a state of mind achieved by the problem solver when "pursuing a form of active learning in which knowledge is often sought as a means to accomplish a particular task in which one is personally invested."

To planners, mistakes are symbols of wrong choices in their logic. For tinkerers, mistakes are signs used to guide them to their answers. These moments of struggle can be considered essential in their iteration and experimentation process (VOSSOUGHI; BEVAN, 2014). According to PETRICH; WILKINSON; BEVAN (2013) "having an artifact to point to, an artifact that may be rickety or lopsided, but yet has resolved the problem that so puzzled the learner" is one of the elements that make tinkering so compelling. In addition, some authors (MARTINEZ; STAGER, 2013; VOSSOUGHI; BEVAN, 2014) argue that employing tinkering helps develop the problem solvers' creativity and confidence.

2.2 Data Mining

Every day large volumes of data are generated from the most diverse activities, from business, science, medicine, and recreation, in almost every aspect of daily life, and this makes the ability to extract useful information and to make sense of these complex amounts of data increasingly important (KANTARDZIC, 2011). However, when conducted manually, besides being expensive and time-consuming, this process may be prone to human error and bias towards the data(HAN; KAMBER; PEI, 2011). The general process of understanding, preparation, and analysis of this data generated is known as data mining.

Data mining is an iterative process that can be defined by the discovery in which humans and computers cooperate to uncover nontrivial and valuable information within large volumes of data. According to KANTARDZIC (2011), this process has two primary goals, predict unknown value based on a set of current information (variables) and finding patterns that can describe the data in a way that humans may interpret. FAYYAD; PIATETSKY-SHAPIRO; SMYTH (1996) defines the data mining process as interactive and iterative, with several decisions made by the human counterpart and loops and interactions between the data mining process's steps. This can be viewed in Figure 2.

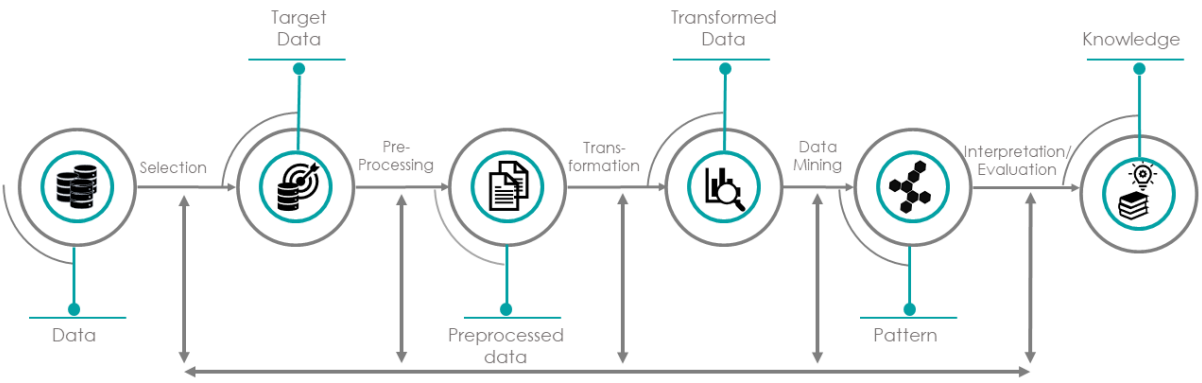


Figure 2 – Data mining process

Source: Elaborated by the author based on FAYYAD; PIATETSKY-SHAPIRO; SMYTH (1996)

A more formal definition is given by the Gartner group that describes data mining as *"...the process of discovering meaningful new correlations, patterns, and trends by sifting through large amounts of data stored in repositories, using pattern recognition technologies as well as statistical and mathematical techniques."*

2.2.1 Educational Data Mining

With educational systems and technologies becoming more present daily, larges amounts of new educational data became available, from students' keystrokes and assessment submissions (BUDIMAN *et al.*, 2017; DUTT; ISMAIL; HERAWAN, 2017) to complete system logs packed with possible new information to be uncovered (BLIKSTEIN *et al.*, 2014).

Educational Data Mining is a field of study where different methods such as data mining and cognitive psychology are implemented in order to analyze educational data as a

way to improve learning, teaching, or institutional effectiveness. According to the International Educational Data Mining Society (EDUCATIONAL. . . ,), educational data mining can be defined as an *" emerging discipline, concerned with developing methods for exploring the unique and increasingly large-scale data that come from educational settings and using those methods to better understand students, and the settings which they learn in".*

Traditionally, researchers have applied several methods to predict students' behaviors and outcomes in several subjects and degrees. In the study conducted by HAMSA; INDIRADEVI; KIZHAKKETHOTTAM (2016), the authors used fuzzy genetic algorithms and decision trees to create a predictive performance model from 120 bachelor students and 48 master degree students during the first semester of a computer science and electronics and communication courses based on their admission scores and grades during the semester. With a similar goal, FENG *et al.* (2017) used the Gini index in combination with a random forest algorithm to predict the performance of three hundred and seventy-nine (379) undergraduate students in an English subject during the course of two semesters. Even though the subjects and the methods applied differ, they share a similar goal to employ educational data mining to construct a predictive model of students' academic performance that can be used in a learning environment to identify students' difficulties earlier in the subject.

Amidst the several methods used in educational data mining, clustering is an unsupervised approach to analyze and collect invaluable information. This method can be used to predict students' performance, associate learning styles, and students' behaviors (DUTT *et al.*, 2015) (2015).In a recent study, DUTT; ISMAIL; HERAWAN (2017) conducted a systematic review over the last three decades of educational data mining works that encompassed one hundred and sixty-six (166). The authors' results show that among the works analyzed, clustering was the most popular data mining method in educational domains, with the partitional method K-means and the hierarchical method Ward's AHC as the most popular algorithms.

JUNIOR (2019), conducts a study to construct predictive models to identify in the first two weeks students that may fail in the subject, as well the use of genetic algorithms to automate the development and optimization of machine learning pipelines. To achieve this goal, the authors extracted 14 features from the automatic correction system's logs generated by the one thousand six hundred and thirty-six (1636) undergraduate students during three years period from introductory programming subject. As a result, the authors achieved an area under the curve (AUC) of 0.87 on the validation set to early predict students' performance. In addition, the authors identify that students' grades in the first two weeks were the most relevant features to predict their performance.

2.2.2 Clustering

Clustering is the process of finding partitions (*clusters*) for data objects in such a way that similar data objects that are somehow related belong in the same group, whereas non-similar

data objects belong to other group (GAN; MA; WU, 2007).

Since the information of the data objects' class label is not present, clustering is considered unsupervised learning, a form of *"learning by observation rather than learning by example"*. Therefore, different clustering methods can lead to different clusters using the same data set. And since the clusters' formation is performed by the unsupervised clustering algorithm, this process can lead to the discovery of new groups and information within the data set analyzed (HAN; KAMBER; PEI, 2011).

According to (HAN; KAMBER; PEI, 2011), clustering algorithms can be divided into five categories: partitioning, hierarchical, grid-based, density-based, and model-based. Partitioning methods find partitions in the data, grouping similar data objects in the same partition, and each group contains at least one object. Hierarchical methods, create hierarchical decomposition of the set of data objects, splitting (divisive) or merging (agglomerative) groups of data objects iteratively until each data object is in one cluster (divisive) or all clusters are merged (agglomerative). The grid-based methods quantize the object space into a finite grid structure. Density-based methods aim to find densely group regions of data objects that can be divided from other groups by a sparse margin. Lastly, model-based methods aim to fit the data objects into a model, such as neural networks or probability density functions. (HAN; KAMBER; PEI, 2011; SILVA; BATISTA; KEOGH, 2018).

Clustering methods have been used in a plethora of applications, such as business intelligence, image pattern recognition, security, web search, and education. It's important to note that given the multitude of options in clustering methods and their characteristics, different methods may be more commonly employed in determined applications.

In this work, our goal is to analyze an educational data set to uncover information about students' problem-solving strategies and the impact that these strategies have on the students' performance through the subject. Therefore, this section is focused on the most popular algorithms used in the literature (DUTT; ISMAIL; HERAWAN, 2017), K-means, and Ward's AHC. In addition, for the sake of readability, the reader may refer to (AGGARWAL, 2016) for a more formal and throughout description of other clustering methods and algorithms.

2.2.2.1 K-means

Partitioning methods, such as K-Means, are the most straightforward clustering algorithms in cluster analysis. Given a data set D that contains n data objects and a previously defined number of partitions (*clusters*), k (where $k \leq n$). The algorithm creates the clusters to optimize a predefined objective function, such as a distance-based dissimilarity function, where the objects that belong to the same cluster are similar to each other and dissimilar to objects in other clusters.

K-Means is a centroid based partitioning method, where a centroid is conceptually the center point of a cluster. These centroids can be defined based on different methods, such as mean

or medoid of data objects that belong to that cluster, and are used to measure the within-cluster variations. At each of the algorithm's iterations, a new data object is assigned to a cluster *Ci*, based on the distance between the data object and the cluster mean, the algorithm then computes a new mean for the cluster based on the data objects assigned, thus minimizing within-cluster variation. This process repeats until there's no change in the clusters' composition (HAN; KAMBER; PEI, 2011). It's worth point that this method does not guarantee convergence to a global optimum, often terminating in a local optimum. Withal, the results may vary due cluster initialization, requiring multiple runs with different initial clusters to produce more reliable results (GAN; MA; WU, 2007; ZAHRA *et al.*, 2015).

The use of K-Means in an educational environment can be seen in the study conducted by KUMAR; SINGH (2017), where, combined with statistical analysis, the algorithm is used to evaluate how external factors such as family background, daily habits, address, sex, and assiduity, may impact on the pass/failure probabilities of four hundred and twelve (412) post-graduate students during the semester. In addition, the authors also assessed the performance of data mining algorithms in that scenario. As a result, the authors achieved a percentage of 61.4% of correct predictions using random forests algorithm and students' pre-admission and academic attributes. In the published study, the authors didn't specify the study's context, impeding the study's replication. Additionally, the authors also pointed out the unreliability of students' self-reported data as one of the limiting factors during the attribute selection phase.

2.2.2.2 Ward's Agglomerative Hierarchical Method

Hierarchical clustering methods group data objects into a hierarchy of clusters and can be divided into agglomerative (bottom-up) or divisive (top-down). An agglomerative hierarchical clustering method works by initially dividing each data object *n* in a separate cluster *Ci* and, based on a predefined similarity measure, merging two closest clusters at each iteration, until all data objects are in a single cluster or a certain condition is meet (GAN; MA; WU, 2007). A Ward's clustering visual representation can be seen in Figure 3 as a dendrogram.

A divisive hierarchical clustering method works on the contrary direction, putting all data objects *n* in the same initial cluster *Ci* and recursively dividing it into several smaller clusters, until each cluster at the lowest level contains only one data object, or the data objects in the same cluster sufficiently similar (HAN; KAMBER; PEI, 2011).

The Ward's Agglomerative Hierarchical Method (Ward's AHC) is an agglomerative method where all data objects are divided into different clusters (*singletons*) and iteratively merge them until all data objects belong to the same cluster. At each iterative merging step, Ward's algorithm selects the two clusters to be merged with minimal variance within-clusters, i.e., the variation between the within-cluster variances before and after the merge occurs is minimized (WARD, 1963).

In the study conducted by Antonenko P., Toy S. and Niederhauser D. (ANTONENKO;

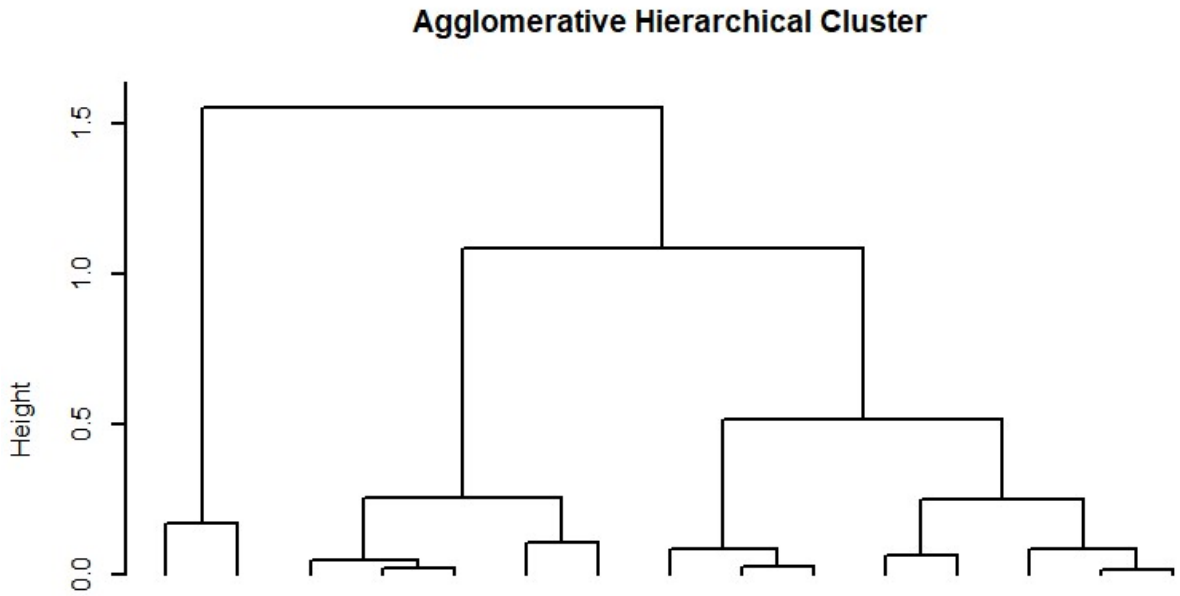


Figure 3 – Hierarchical clustering displayed as a dendrogram.

Source: Elaborated by the author

TOY; NIEDERHAUSER, 2012), Ward's algorithm was used with other partitional algorithms to automatic analyze characteristics of one hundred and eighty-three (183) undergraduate students in an engineering economics class and their learning behavior while engaged in collaborative problem-solving activities in an online learning environment. As a result, the study proposes students' labels (high and low performing students) based on their behaviors while browsing online for information related to their tasks. However, only data from 40 of the 183 students were used in their final analysis (20 students in the high-performance group and 20 students in the low-performance group), while the rest were dropped from the analysis without further discussion. In addition to the student classification, the study discusses the impact of algorithm and clustering measures selection to be used during a clustering analysis. Furthermore, the authors focused on analyzing the students' performance during problem-solving activities instead of the students' problem-solving strategies as the level of tinkerers and planners.

2.2.3 Association Rule Mining

Association rule mining (HAN; KAMBER; PEI, 2011) is a data mining technique used to discover associations and correlations of elements among large amounts of data in transitional or relational data set. Also known as market basket analysis, this technique is famous for its application in customer habits analysis, where it finds associations between the customers' purchased products or their "shopping baskets." This technique can also be used to describe the associations among clustering partitions, obtaining this way, new information about the clusters

formed.

Consider a data set D composed by T non-empty transactions and I a set of items such as $I = I_1, I_2, I_3, ..., I_n$ and $T \subseteq I$. Where a transaction can be seen as an event that occurred involving a set of items. A transaction T is said to contain any set of items X if $X \subseteq T$. The association rules implications follow the form of $X_i \implies X_j$, where $X_i \subset I, X_j \subset I, X_i \neq \emptyset, X_j \neq \emptyset$ and $X_i \cap X_j = \emptyset$, which means that to create an implication rule $X_i \implies X_j$, the item sets X_i and X_j must belong to the same itemset I and not having any items in common.

The three following metrics are commonly used in order to assess the rule's strength, or how interesting the rules are:

- Support is the percentage of transactions T in D that contain both X_i and X_j and it's calculated as $supp(X_i \implies X_j) =$ probability $P(X_i \cup X_j)$;
- Confidence is the percentage of transaction T in D containing X_i that also contain X_j and it's calculated as $conf(X_i \implies X_j) =$ probability $P(X_j|X_i)$;
- Lift is a correlation measure used to boost the support-confidence of a association rule and can be obtained by $lift(X_i, X_j) = \frac{P(X_i \cup X_j)}{P(X_i)P(X_j)}$ which is equivalent to $confidence(X_i \implies X_j)/support(B)$.

Both support and confidence values range from 0% to 100%, and when these values are higher than a predefined lower threshold (minimum support or minimum confidence respectively), those rules are considered strong/interesting. Lift values, however, indicates the correlation between the itemsets. For lift values lesser than one ($lift(X_i, X_j) < 1$), it expresses a negative correlation, which means that the occurrence of one itemset indicates the absence of the other itemset. If the lift values are greater than one ($lift(X_i, X_j) > 1$), it expresses a positive correlation, which means that the occurrence of one itemset indicates the occurrence of the other itemset. In case of lift values equal to one ($lift(X_i, X_j) = 1$), the itemsets $X_i and X_j$ have no correlation, which means that they are independent (HAN; KAMBER; PEI, 2011).

2.3 Final Remarks

In this section were presented the main concepts and methods necessary for a better understanding of this work. A brief contextualization of the main concepts related to students' problem-solving strategies and some of the works that aim to classify students based on these strategies, focusing on the tinkering approach described by Turkle and Papert and how the tinkering approach differs from more structured approaches. Furthermore, Data Mining and Educational data mining were contextualized, and their methods (clustering and Association Rule Mining) and algorithms used in this work (Ward's AHC and K-means) were briefly discussed. In addition, the researches related to this work were presented through the chapter, in particular

the literature pertaining to the use of educational data mining to identify students' problem-solving strategies depending on the concepts that each research approached (VITAL *et al.*, 2019; BLIKSTEIN *et al.*, 2014; ANTONENKO; TOY; NIEDERHAUSER, 2012). A summary of this work and the related works presented in this section can be seen in Table 1 below.

Table 1 – Related works summary

Author	Number of participants	Education Level	Application Domain	Approach	Data collection period	Study objective
PERKINS *et al.* (1986)	not specified	Elementary and High school	Introductory programming (Basic and LOGO)	Structured observation	not specified	Identify students' problem-solving behaviors
BRUCE *et al.* (2004)	13	Undergraduate	Introductory programming	phenomenographic	1 sem	Analyze students' learning approaches
BLIKSTEIN *et al.* (2014)	370	Undergraduate	Introductory programming	Dynamic time warping and Clustering	2 sems	Predict students' performance
SHARMA *et al.* (2018)	600	Undergraduate	Introductory programming (Java)	Generalized additive models	1 sem	Predict students' performance based on unity tests
HAMSA; INDIRADEVI; KIZHAKKETHOTTAM (2016)	168	Undergraduate	Computer Science and electronics and communication	Fuzzy genetic algorithm and decision trees	1 sem	Predict students' performance based on admission scores
FENG *et al.* (2017)	379	Undergraduate	English language	Gini index and random forests	2 sems	Predict students' performance
DUTT; ISMAIL; HERAWAN (2017)	166	not specified	Educatiuonal Data Mining (EDM)	Systematic review	30 yrs	Provide an overview of Educational data mining
JUNIOR (2019)	1363	Undergraduate	Introductory programming	Genetic algorithm	6 sems	Early prediction of Students performance
KUMAR; SINGH (2017)	412	Post-graduate	not specified	Clustering and statistical analysis	1 sem	Assess the influence of external factors in students' performance
ANTONENKO; TOY; NIEDERHAUSER (2012)	183	Undergraduate	Engineering economics	Clustering	not specified	Predict students' performance and assess the impact of different algorithms in the prediction
This work	59	Undergraduate	Introductory programming	Clustering	1 sem	Investigate the students' tinkering behaviors and its impact on the students' grades performance

Source: Elaborated by the author

CHAPTER

3

RESEARCH DESIGN

This section describes the process of data selection and pre-processing, methods and tools used in this work. Section 3.1 presents the research question and hypothesis formulated related to this work's objectives. Section 3.2 describes the data source and materials used by the students during the activities. Section 3.3 describes the methodology process adopted to develop this work.

3.1 Research questions

Based on the main goal of this work, presented in Section 1.2.1, *Investigate the students' tinkerer and planner behaviors in online environments and its impact on the students' grades performance.*, the following research questions were formulated:

In order to assess the impacts of the students' tinkering behavior on the students' grades, first is necessary to identify the students' behaviors in the selected dataset, therefore:

RQ1: Which students' tinkering behaviors can be identified using the selected assignment features?

Once the students' tinkering behaviors are identified, it's necessary to evaluate how representative are the features. Students that present the same behavior may need different amounts of time or number of submissions to complete an assignment, or students that presented different behaviors may have a similar number of submissions during assignments. How those features are related to the identified students' tinkering behaviors may give a deeper understanding of those. Thence:

RQ1.1: How can the differences and similarities in student's tinkering behaviors be characterized based on the selected features?

The main goal of this work is to investigate the students' tinkerer and planner behaviors

in online environments and its impact on the students' grades performance, researches questions RQ 2 and RQ 2.1 refers to evaluate the impact that the students' tinkering behaviors, which are expressions of the students' problem-solving strategies, have on the students' grades performances. In addition, the RQ 2.1 aims to assess the impact of different assignments in the students' performance, which may not provide the necessary conditions for students' to employ their tinkering problem-solving strategies.

RQ2: Is there a grade performance difference between the identified students' tinkering behaviors?

The alternative hypothesis was formulated based on the premise that students' would have different performances based on their tinkering behaviors.

- Null Hypothesis: There are no differences between identified students' tinkering behaviors grades performance.

  H_0**:** $GP(T_i) = GP(T_j)$, for all (i,j) contained in 1,2,...,n

- Alternative Hypothesis: There are differences between identified students' tinkering behaviors grades performance.

  H_0**:** $GP(T_i) = GP(T_j)$, for any i!=j, contained in 1,2,...,n Where $GP()$ is the grade performance of the students' tinkering behavior T_n.

RQ2.1: Is there a significant difference in grades performance among the tinkering behaviors identified depending on the assignment format?

Besides the identification of tinkering behaviors and the impact of those on the student's grades performance, the following research questions evaluate the impact in the time spent by a student on their assignments.

The alternative hypothesis formulated to research questions 3 and 4 (RQ3 and RQ4) were based on the premise that students who presented more planned behaviors would take similar amounts of time to finish their assignments than those students who presented more tinkerers behaviors.

RQ3: Is the total time to complete an online assignment different among the identified tinkering behaviors?

- Null Hypothesis: There is no difference in **the total time to complete** an online assignment among the identified students' tinkering behaviors.

  H_0**:** $W_(T_i) = W_(T_j)$, for all (i,j) contained in 1,2,...,n

- Alternative Hypothesis: There is a difference in **the total time to complete** an online assignment among the identified students' tinkering behaviors.

H_0**:** $W_(T_i, S_i) = W_(T_j, S_j)$, for any i!=j, contained in 1,2,...,n Where $W()$ is the total time to complete the online assignments (total time) of the students' S_n tinkering behavior T_n.

RQ4: Is the students' time on task for online assignments different among the identified tinkering behaviors?

- Null Hypothesis: There is no difference in the students' time on task for online assignments among the identified students' tinkering behaviors.

  H_0**:** $ToT_(T_i, S_i) = ToT_(T_j, S_j)$, for all (i,j) contained in 1,2,...,n

- Alternative Hypothesis: There is a difference in the students' time on task for online assignments among the identified students' tinkering behaviors.

  H_0**:** $ToT_(T_i) = ToT_(T_j)$, for any i!=j, contained in 1,2,...,n Where $TTS()$ is the students' time on task during online assignments of the students' S_n tinkering behavior T_n.

The last research question was formulated based on the premise that the students may change their tinkering behaviors while attempting to solve problems (TURKLE; PAPERT, 1992; VOSSOUGHI; BEVAN, 2014; SHARMA *et al.*, 2018).

RQ5: Do students present different tinkering behaviors during online assignments?

- Null Hypothesis: Students don't present different tinkering behaviors during online assignments.
- Alternative Hypothesis: Students present different tinkering behaviors during online assignments.

3.2 Materials

3.2.1 Data source

To achieve the objectives listed in 1.2 and to answer the research questions in section 3.1, it was used real data on the students' problem-solving strategies employed in online assignments. The selected data source was composed of a class of sixty (60) undergraduate students regularly enrolled in the subject SSC600 - Introduction to Computer Science I in the University of São Paulo, campus São Carlos, during the first semester of 2017. This subject has the objective of *"Present the basic concepts of computational thinking applied to problem-solving. Develop skills to write small programs using a programming language. Basic concepts about computers and computing. Problem-solving and algorithm design. Programming structures. Simple data types. Modularization. Composite data types. Files. Debugging. Structured programming language"* (USPDIGITAL, 1999)

The choice of an introductory subject was due to the following factors: i) TURKLE; PAPERT (1992) argues that the negotiating style of tinkers is more present when they experience new challenges, so a student that has new concepts presented to them at each assignment would be more prone to employ tinkering strategies through these assignments; and ii) So the results obtained in this work could be easier compared with others related works present in the literature, such as BLIKSTEIN *et al.* (2014), SHARMA *et al.* (2018) and BRUCE *et al.* (2004).

3.2.2 Assignments

During the semester, the students were presented with eighteen (18) online assignments in C language that covered three main concepts: i) Algorithm and conditional structures, ii) Looping structures and Arrays, and iii) Functions and recursion. The assignments titles and the concepts comprehended can be seem in Table 2. The students' final grade was composed of grades of three different assignment types: In-class pen-and-paper assignments, online assignments, and pen-and-paper tests.

In the pen-and-paper assignments, the students were asked to solve small tests proposed by the teacher about the topic passed on that class. The teacher corrected the assignments. The grades were given to the students based on the number of correct answers on their assignments. The average grade of this assignment comprehended 25% of the students' final grade. Since these assignments were applied during the classes, the time that the students had to solve them was limited by their classes' length.

In the online assignments, it was possible to observe students' behaviors, each of eighteen (18) assignments were composed of a test case file and a single problem to be solved by the students using an open-source learning platform (moodle.org) [1]. At each submission, the student would be presented by the automatic evaluation of the platform and the system's feedback in the form of compilation messages and test results, and similar to the pen-and-paper assignments, the average of online assignments grades comprehended 25% of the students' final grade. These assignments had to be delivered before the end of the subject, giving the students some freedom to choose when to solve them.

The tests were divided into the midterm and final tests developed and evaluated by the teacher and would comprehend all the topics discussed in class until that period. These tests were solved by the students during the classes using pen and paper and also had a time limit of a class. The average of their test grades comprehended 50% of the students' final grades.

Even though the tinkering behaviors could only be observed in the online environments, the students' grades in the other assignments were also assessed as it could be an indication of changes in the students' problem-solving skills and the impacts of the assignments' limitations may have on students. In this case, the inability to employ their tinkering problem-solving

[1] Moodle system version 3.1

Table 2 – Assignments applied during the semesters

ID	Title	Concept
493	Hello World	Algorithm and conditional structures
496	Square Root	Algorithm and conditional structures
497	Triangle Area	Algorithm and conditional structures
504	Temperature monitor	Algorithm and conditional structures
-	In class activity	Algorithm and conditional structures
498	Metabolic rate	Algorithm and conditional structures
515	Diet calculator	Algorithm and conditional structures
-	In class activity	Algorithm and conditional structures
-	in class Test	All content seen
545	Proper divisor	Looping structures and Arrays
457	Hailstone numbers	Looping structures and Arrays
-	In class activity	Looping structures and Arrays
520	Fibonacci sequence	Looping structures and Arrays
522	Christmas tree	Looping structures and Arrays
-	In class activity	Looping structures and Arrays
528	Sequence of power	Looping structures and Arrays
524	ATM	Looping structures and Arrays
478	Semiprime numbers counter	Looping structures and Arrays
-	In class activity	Looping structures and Arrays
534	Fibonacci's polinomial	Functions and recursion
538	Lucky number generator	Functions and recursion
-	In class activity	Functions and recursion
542	Planning poker	Functions and recursion
544	Palindrome's counter	Functions and recursion
545	Maze's exit	Functions and recursion
-	In class activity	Functions and recursion
-	in class Test	All content seen

Source: Elaborated by the author based on the subject's material

strategies.

The average and standard deviation of the students' grades in the evaluated assignments can be seen in Table 3.

3.2.3 Moodle

The Moodle (*Modular Object-Oriented Dynamic Learning Environment*) is an open-source course management system to create personalized learning environments that provides a

Assignment	Pen-and-paper midterm test	Pen-and-paper final test	Tests' average	Pen-and-paper assignments	Online assignments	Final grades
Average	6.817702	8.961864	7.877071563	7.389943503	8.648439907	7.958301
Standard Deviation	2.043714	1.941853	1.491891893	1.380093953	1.254343785	1.132759

Table 3 – Students' average grades by assignment

Source: Elaborated by the author based on the subjects' material

set of tools to support inquiry and discovery-based approaches for online learning (BRANDL, 2005).

Using the *Virtual Programming Lab* (VPL) module from Moodle (PINO, 2011), the students were able to develop, submit, run and evaluate their online assignments during the semester, Figure 4 illustrates the initial submission process, where the student could develop and compile their codes. Through the Moodle platform, the students were presented with a problem statement, hints to guide them to solve the problem, and the files needed to evaluate their code before submission.

In the assignment's assessment option in Figure 5, the students could assess their codes and allowed multiple submissions during its time window. To improve the completion of the code based on the test cases, the students also received feedback from the system in the form of compilation outputs, Figure 6 and Figure 5, and the tests' summary at each submission, as can be seen in Figure 7.

3.3 Methodology

To investigate the students' tinkerer and planner behaviors in online environments and its impact on the students' grades performance using data prior to this work's development, it was used the *Ex Post Facto* research methodology (SIMON; GOES, 2013; LORD, 1973). In the ex post facto research methodology, the researcher identifies events that have already occurred and investigate the relationships between these events and the outcomes (LEEDY; ORMROD, 2014). As outlined by ISAAC (1971), this methodology can be defined in the seven (7) following steps:

1. Problem definition.
2. Survey the literature.
3. Hypotheses creation.
4. List of assumptions that will serve as the base for the procedures.
5. Approach design:
    a) Selection of subjects and materials.
    b) Selection of data collecting techniques.

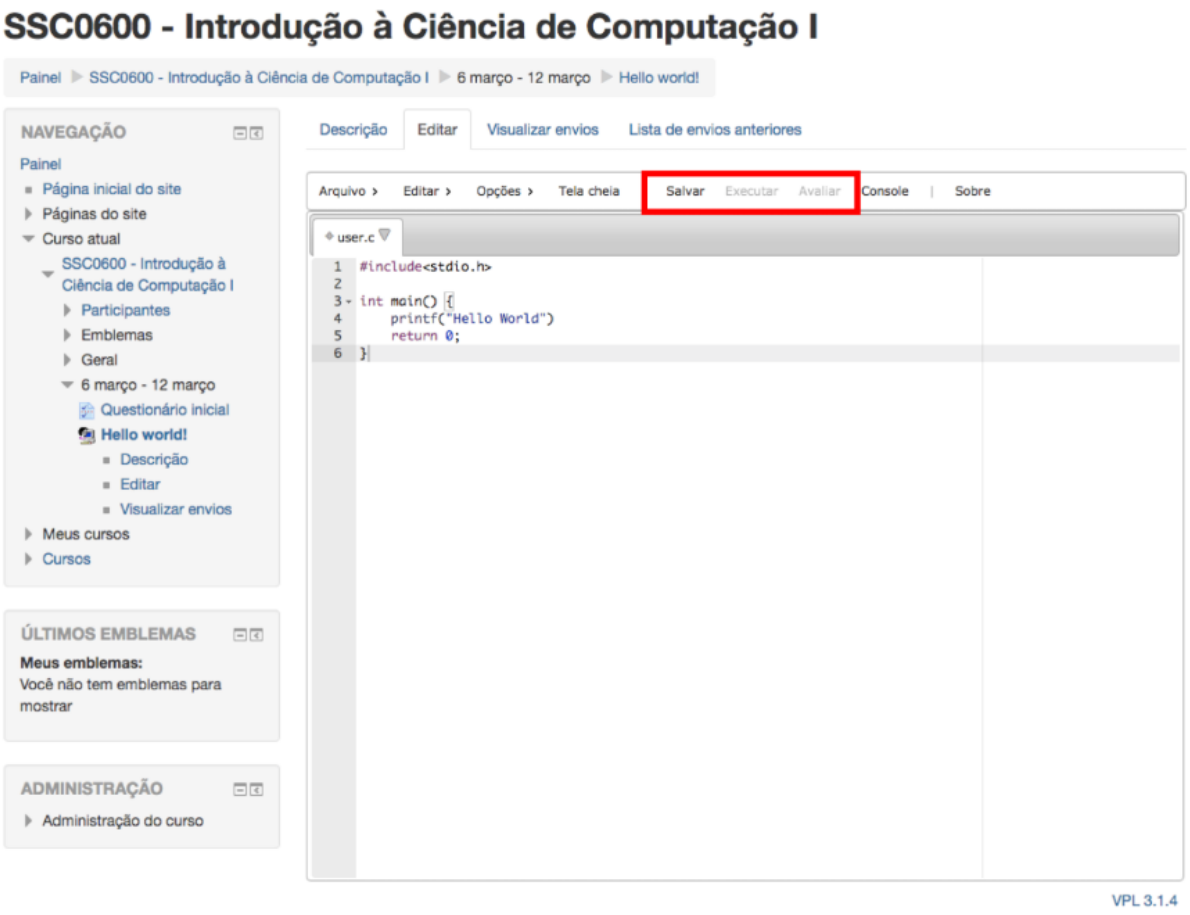


Figure 4 – Moodle's main page

Source: Screenshot from moodle's system

c) Data classification.

6. Data validation.

7. Findings analyzes

Each step of the methodology is described in more detail during the following sections: the first step - *problem definition* was discussed through chapter 1; Subsection 2 addresses the second step of the process, *survey the literature*, discussing the related works and state of the art of the topics approached in this work; Subsection 3.1, underlines the hypotheses creation process and the assumptions on which they were based upon, that represents the third and fourth steps - *Hypotheses creation* and *List of assumptions that will serve as the base for the procedures.* respectively;

The fifth step *approach design* is divided in three parts: *a) selection of subjects and materials* is discussed through subsections 3.3.1 and 3.2 describes the selection of subjects and

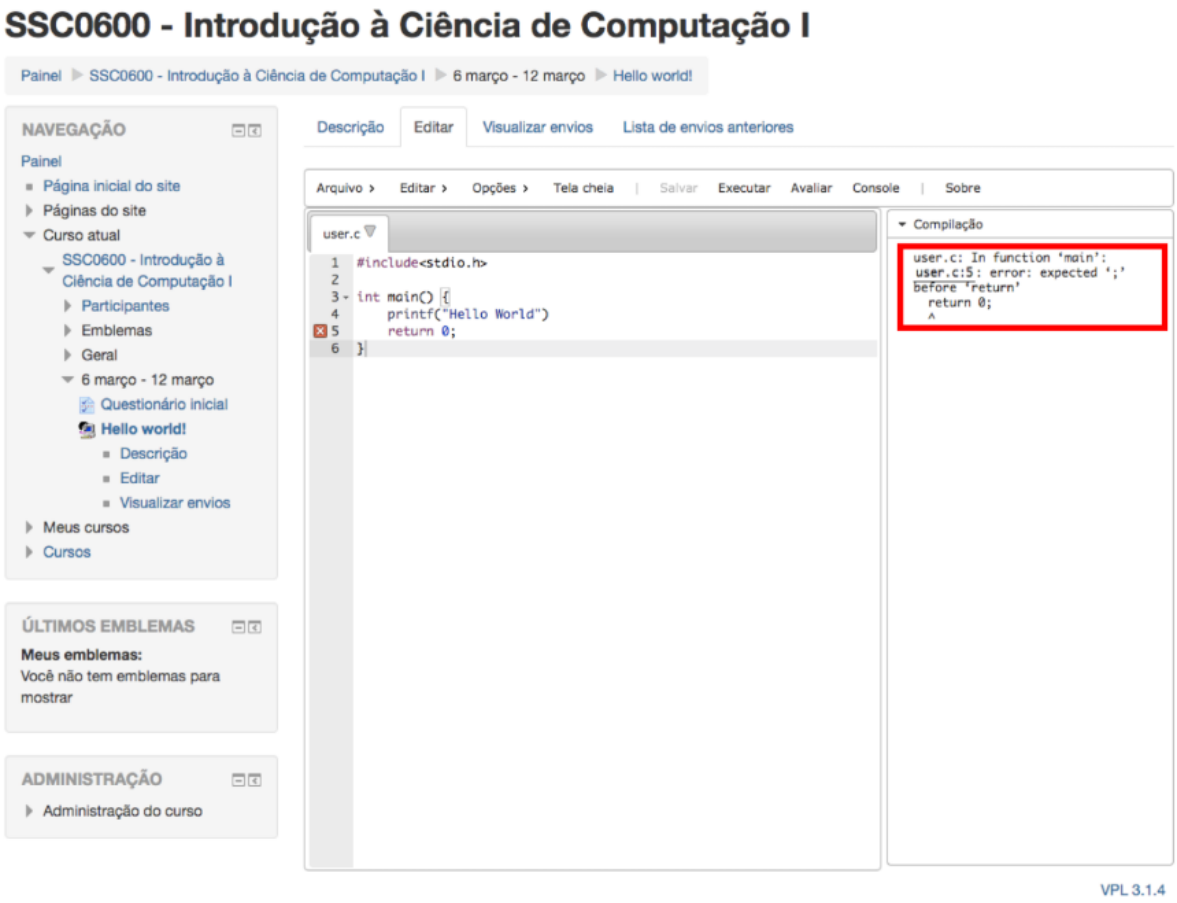


Figure 5 – Moodle's compilation page

Source: Screenshot from moodle's system

obtention of the students' data, as well as an overview of the data collected; *b) Selection of data collecting techniques* and *c) data classification* are discussed through subsections 3.3.2 and 3.3.3, where it addresses the data preprocessing and transformation, and is presented the feature extraction process, the data reduction process and data mining choices made during the development of this work;

Steps six and seven are discussed in the chapters 3.3.3 and 4, where the data mining techniques validation and the results obtained are presented and reported accordingly.

In addition to the ex post facto methodology used, a data mining process outlined by FAYYAD; PIATETSKY-SHAPIRO; SMYTH (1996) was adopted to help answer the first research question proposed in this work - *RQ1: Which students' tinkering behaviors can be identified using the selected assignment features ?*-. This process can be summarized in nine steps ranging from understanding the application domain and dataset creation to knowledge discovery and application (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996).

Figure 6 – Moodle's test page

Source: Screenshot from moodle's system

1. Understanding of the application domain: Acquiring the relevant knowledge about the application domain and its goals;
2. Dataset creation: Selection of the dataset in which further steps will be performed;
3. Preprocessing data: Handling incomplete data or noise from the selected dataset;
4. Data reduction: Selecting relevant features to represent the data amidst the selected dataset;
5. choice of data mining approach: Selecting the data mining approach based on the predetermined goals (e.g., classification, regression or clustering);
6. choice of data mining algorithm: Selecting the data mining algorithms to be used amidst the available algorithms given the approach chosen in the previous step;
7. Data mining: Application of the previously chosen data mining algorithm;
8. Results interpretation: Interpreting the results obtained to be applicable for the predefined goals;

Figure 7 – Moodle's test summary page

Source: Screenshot from moodle's system

9. Application of the discovered knowledge: Acting on the knowledge discovered (e.g., reporting, implementing into a system and/or documenting it);

3.3.1 Data collection

During the students' interaction with the moodle system internal system's records were generated in log files. Log files are automatically generated files that contain records of events that occurred in the system. In this case, these files included the crucial data related to the student's submissions and interactions with the system, such as timestamps, changes in the students' code, and students' and system's data. This available data was selected and collected for further analysis.

To collect the students' data and to facilitate a local dataset creation based on the students' data from the Moodle web platform as part of step 5 b) *selection of subjects and materials*, an automated tool to extract data from the moodle platform was developed using python language[2]

[2] using Python version 3.6.5

and the BeautifulSoup [3] library [4].

The data collected was divided by assignments, which was subdivided by students' IDs, and each student's ID contained all submissions made by the students for that assignment. All the data collected was stored locally to simplify its handling in the subsequent steps.

In this initial phase, it was collected a little over eleven thousand and four hundred (11,400) students' codes snapshots with timestamps, from nine hundred and fifty-one students' (951) assignments from the moodle platform. A sample of the collected data can be seen in Figure 8.

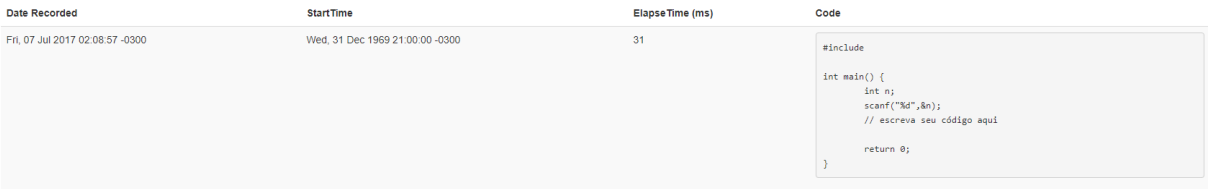


Figure 8 – Snapshot of students' code with timestamp

Source: Screenshot from moodle's system

3.3.2 Features selection

This section describes the feature selection process and extraction of students' assignments features related to tinkering behavior proposed to achieve this work's specific objective 1: "Identify students' assignments features related to the tinkering problem-solving strategy" (see Section 1.2.2).

Turkle and Papert (TURKLE; PAPERT, 1992) classify students dichotomically between tinkerers and planners, where planners are students that present a structured, top-down strategy to solve problems, while tinkerers employ an alternative, less structured, bottom-up problem-solving strategy, that uses system's feedback to achieve their goals. However, as some authors argue, a binary definition may not take into account some nuances in those behaviors (BLIKSTEIN *et al.*, 2014).

In this work, as discussed by BLIKSTEIN *et al.* (2014), it was considered that students might present different behaviors, where tinkerers and planners were considered two extremes of the tinkering behaviors that the students could present during the assignments. On one side, a top-down and more structured approach, with a low number of interactions with the system, the planners, and on the other side, a bottom-up and flexible approach, on which was expected the student to complete the assignment through system feedback, the tinkerers.

[3] <http://www.crummy.com/software/BeautifulSoup/> version 4.8.1

[4] The code used can be seen at <https://github.com/fcarvalhos/masters/tree/master/Python%20Crawlers>

Based on the premise that the target variables had to express students' interactions with the system or the time dedicated by the students to complete the assignment, the features extracted were selected to express the tinkering behaviors that the students may present during the assignments. Based on this and in the related work presented in section 2, the following features were selected:

- **time on-task:** This feature represents the amount of time that the student spent actively solving the assignment (time-on-task) through any number of sessions using Moodle's VPL module, and it's expressed by the Formula 3.1. A session was defined as any amount of time that the student is logged and interacting with the system, where the time difference between two interactions has to be lower than one hour. If the time difference is greater than one hour, it was considered that the student initiates a new session. To estimate the duration of a session, it was calculated the difference of time between timestamps from the system logs.

$$\textbf{time on-task} = \sum_{0}^{n} S_{n_{end}} - S_{n_{start}} \tag{3.1}$$

  Where n is the total number of sessions that one student took to complete the assignment, $S_{n_{start}}$ is the first timestamp of the session n and $S_{n_{end}}$ is the last timestamp of the session n.

- **total time:** It expresses the total amount of time that the student took to complete the assignment. It's calculated by the time difference between the first and last interaction with the moodle's VPL module. This feature takes into account that students may seek advice from tutors, friends, or online to solve the problem, as well that some students may pause the assignment and resume it at a later date for any other reasons.

- **Submissions:** Represents the number of submissions that the students needed to complete the assignment. A higher number of submissions indicates a higher number of interactions with the system by the student, which results in a greater amount of feedback from the system.

- **difNloc:** This feature represents the average number of lines of code (NLOC) that are added or excluded by a student during each assignment submission during any period of time and it's calculated as the average of the difference between the number of lines of code in submission and the number of lines of code in the previous submission, through all the student's submissions in the same assignment (e.g., the variation of lines of codes between the submissions presented in source code 1 and source code 2 was plus fourteen). This feature can be expressed by the Formula 3.2. This way, this feature may reflect the students' tinkering behaviors since these features are affected by the number of submissions that take to a student to finish the assignment and consequently, the number of interactions between the student and the system. In addition, the size of the changes made by the

student between the submissions, which indicates how significant are the changes made by the student at each submission represented by the number of lines.

Source code 1 – Student's submission n-1

```
1
2     int main() {
3      char g; #gender
4      int tmb, i; #ingest
5      scanf("%d %d %c", &tmb, &i, &g);
6      if(g == 'm'){
7          printf("Male ");
8          if(i<1800){
9              printf("Warning");
10         }
11         else if(i>=1800 && i< tmb -
       400){
12             printf("Lose weight");
13         }
14         else if(i>=tmb-400 && i<=tmb
       +500){
15             printf("Keep weight");
16         }
17         else if(i > tmb +500){
18             printf("Gain weight");
19         }
20     }
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38     return 0;
39 }
40
```

Source code 2 – Student's submission n

```
1
2
3     int main() {
4     char g; #gender
5     int tmb, i; #ingest
6     scanf("%d %d %c", &tmb, &i, &g);
7     if(g == 'm'){
8          printf("Male ");
9          if(<1800){
10             printf("Warning");
11         }
12         else if(i>=1800 && i< tmb - 400){
13             printf("Lose weight");
14         }
15         else if(i>=tmb-400 && i<=tmb+500){
16             printf("Keep weight");
17         }
18         else if(i > tmb +500){
19             printf("Gain weight");
20         }
21    }
22
23    if(g == 'f'){
24         printf("Female ");
25         if(i < 1200){
26             printf("Warning");
27         }
28         else if(i >=1200 && i < tmb-500){
29             printf("Lose weight");
30         }
31          else if(i>=tmb-500 && i<=tmb+400)
      {
32             printf("Keep weight");
33         }
34         else if(i > tmb+400){
35             printf("Gain weight");
36         }
37    }
38
39    return 0;
40 }
41
```

$$\textbf{difNLOC} = \frac{\sum_{n}^{1} NLOC_i - NLOC_{i-1}}{n} \tag{3.2}$$

Where n is the number of submissions made by a student in a given assignment i, with $i = 0, \ldots, n$, and $NLOC_i$ is the number of lines of code in the submission i. [5]

- **difCCN:** This feature represents the variation of the students' code cyclomatic complexity through a given assignment. Similar to difNLOC, it's calculated as the average of the difference between the cyclomatic complexity in submission and the cyclomatic complexity in the previous submission, through all the student's submissions made during any period of time in the same assignment (Formula 3.3). The cyclomatic complexity is a metric that measures the number of linearly independents paths through a given fragment of code (MCCABE, 1976). In the previous example, the variation of cyclomatic complexity between the submissions presented in source code 1 (CCN 8) and source code 2 (CCN 15) was plus 7 . Similar to the difNLOC feature, this feature also indicates the amount of feedback that the student received from the system through the number of submissions and consequently the number of interactions between the student and the system, as well how significant are the changes made by the student at each submission due to the changes in the code's complexity.

$$\textbf{difCCN} = \frac{\sum_{n}^{1} CCN_i - CCN_{i-1}}{n} \tag{3.3}$$

Where n is the number of submissions made by a student in a given assignment i, with $i = 0, \ldots, n$, and CCN_i the cyclomatic complexity value (number) in the submission i. [6]

It's worth pointing out that while the time-on-task and total time reflect the amount of time that a student dedicated to complete the assignment, the other features, submissions, difNLOC, and difCCN, even though related, may have their variations during any amount of time (e.g., a student X may have two submissions during a seven days period, while a student Y may have five submissions in a ten minutes session).

Once the features were selected, [7] each student had a set of the selected features for each assignment submitted. This way, each student-assignment pair was treated as a new data object entry in the dataset. This approach derives from the different tinkering behaviors that students may express during the assignments, depending on how familiar they are with the problem domain and if they already acquire the skills necessary to complete the assignment

[5] It's considered that all students start with zero lines of code before the first submission, to handle cases where a student makes only one submission

[6] It's considered that all students start with cyclomatic complexity zero before the first submission, to handle cases where a student makes only one submission

[7] the file with the selected features and students' grades can be seen at <https://github.com/fcarvalhos/masters/tree/master/Files>

Table 4 – Header of the dataset

studentID	difNLOC	difCCN	submissions	total time	time on-task	actv
10169	0.049202	0.050325	0.194805195	1.41E-05	0.021943075	454
10170	0.293144	0.324675	0.025974026	1	0.098028303	454
10171	0.170213	0.24026	0.038961039	2.18E-05	0.03386866	454
10172	0.446809	0.493506	0.012987013	3.59E-06	0.005565273	454
10174	0.087234	0.088312	0.116883117	7.88E-05	0.122356495	454
10175	0.285461	0.409091	0.038961039	0.002614832	0.107409763	454

Source: Elaborated by the author

(VOSSOUGHI; BEVAN, 2014). In addition, to avoid that the expected difference in magnitudes between each assignment interfered in the dataset evaluation, for each assignment, each feature was normalized separately using equation 3.4, so each feature would have values between zero and one, as it can be seen in the Table 4.

$$f_{norm} = \frac{f - f_{min}}{f_{max} - f_{min}} \tag{3.4}$$

Where f_{norm} is the new normalized feature value, f is the feature value to be normalized, and f_{min} and f_{max} are respectively the lowest and highest values of that feature among all the students in that assignment.

In order to select relevant features to represent the data amidst the selected dataset, a reduction technique was employed. To reduce the dataset's dimensions and avoid any redundant information in the data, correlation tests were made over the new dataset, using Spearman's rank correlation coefficients (SPEARMAN, 1904) (Figure 9). Due to the high correlation between the features difCCN and difNLOC ($\rho = 0.95$), it was opted to remove difCCN due to the higher explicability of lines of code over cyclomatic complexity.

3.3.3 Selection of the data mining approach

This work's main goal is to investigate the students' tinkerer and planner behaviors in online environments and its impact on the students' grades performance. However, it's necessary to define a data mining approach to identify which features better describes the students' tinkering behaviors and how many different behaviors can be identified in the dataset. To mitigate possible biases in these definitions, the created dataset did not present any human assigned labels. Thus, an algorithm capable of identifying similar behaviors between students during the assignments and to differentiate this behavior from others is necessary.

To identify the student's behaviors based on their features, a clustering approach was chosen. As defined in subsection 2.2.2, clustering algorithms are unsupervised learning process that may lead to the discovery of new groups and information within a chosen dataset (HAN;

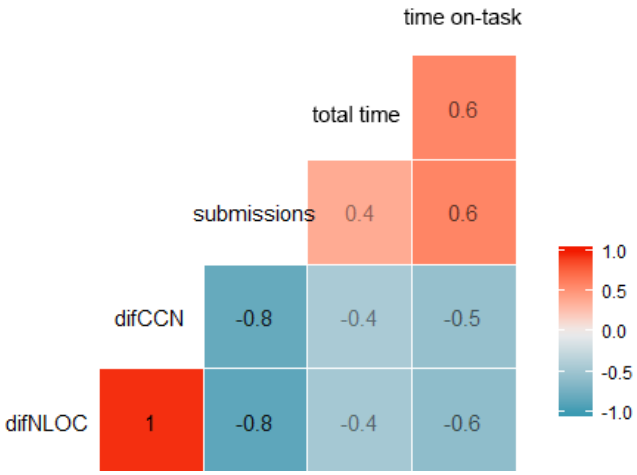


Figure 9 – Feature's correlation

Source: Elaborated by the author

KAMBER; PEI, 2011). This definition is adequate to the tasks faced in this step of the data mining process: Define which features better describes the student's tinkering behaviors and identify how many behaviors are present in the dataset.

3.3.4 Choice of data mining algorithm

To select an unsupervised educational data mining approach to identify students' tinkering behaviors related to problem-solving, two different clustering approaches were chosen, a partitional clustering algorithm (k-means) and a hierarchical agglomerative clustering algorithm (Ward's AHC). These approaches were chosen based on the systematic review conducted by DUTT; ISMAIL; HERAWAN (2017) and the related literature presented in subsection 2.2.1,

Once the clustering algorithms were chosen, a quality comparison process, similar to the process presented in subsection 3.3.2, was made to ensure that results obtained would represent the most accurate information about the students' tinkering behaviors and the features that describe those behaviors. This process was composed of the following steps:

- **Define the number of clusters(k):** Choosing the right number of cluster k isn't a trivial task. Most clustering algorithms rely upon the definition of the number of clusters by the user. However, the desired K is usually unknown (VENDRAMIN; CAMPELLO; HRUSCHKA, 2010). Moreover, the number of clusters depends on several factors that vary by each problem and involves data characteristics such as data distribution, scale, and the knowledge domain (HAN; KAMBER; PEI, 2011).

  Several methods to define the number of clusters can be employed depending on the task, from simple methods as using $\sqrt{\frac{n}{2}}$ clusters for a dataset with n data object, to more

complex ones, such as information-theoretic approaches (HAN; KAMBER; PEI, 2011) [8] . In this work, the Elbow method was used to determine the number of clusters. The elbow method is a heuristic approach to reduce the sum of within-cluster variance. This variance is obtained by applying a clustering algorithm to a dataset n times, and for each n is calculated the sum of square errors (SSE). The most significant turning point in a plot with SSE in respect to k present the indicate number of clusters k (e.g., in Figure 10 the suggested k is 3) (HAN; KAMBER; PEI, 2011).

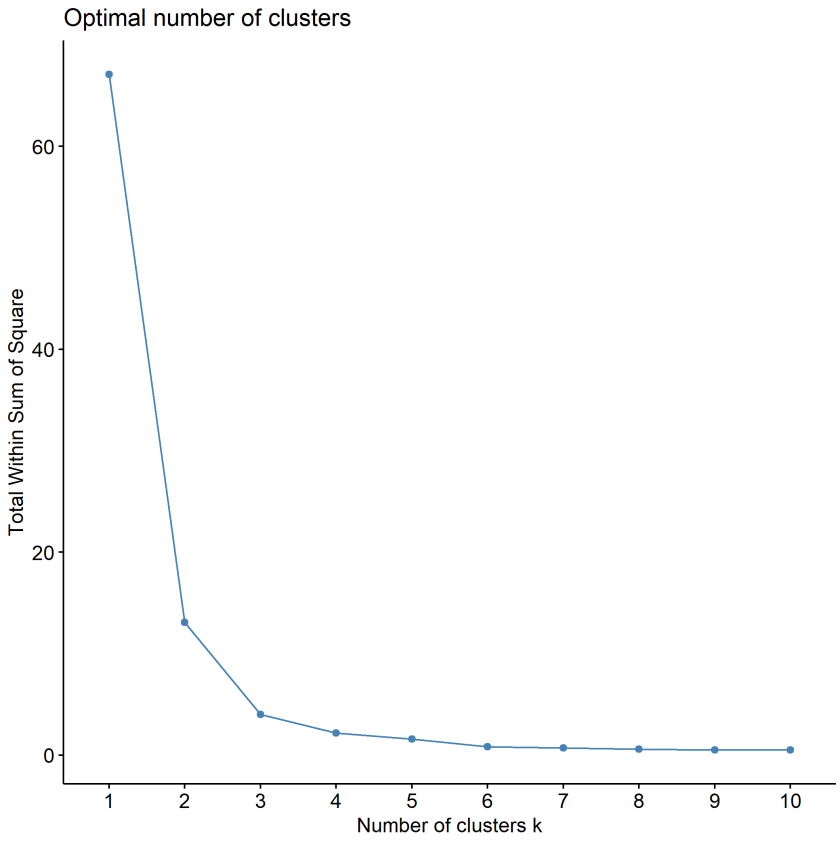


Figure 10 – Elbow method applied to the total time feature using K-means algorithm

Source: Elaborated by the author

Due to the low number of features present in the dataset, it was possible to employ a broad range assessment of the clusters using the elbow method in all combinations of features for the selected algorithms (Ward's AHC and K-means), which resulted in a range of suggested number of clusters between two and four clusters depending on the set of features used. Then, both clustering algorithms were ran using different settings, each one for the suggested number of clusters (k).

- **Assessment of the clustering quality:** One of the main goals of clustering algorithms is to find partitions so that similar data objects are assigned to the same group, and dissimilar

[8] For a more throughout bibliography it's suggested the reading the Section 10.9 of HAN; KAMBER; PEI (2011)

ones are assigned to other groups. However, an improper attribution of parameters, such as the number of clusters K, may result in a non-optimum partitioning result for a given dataset (VAZIRGIANNIS, 2009).

Several measures can be used in order to evaluate the validity of clusters, such as Davie-Bouldin (DAVIES; BOULDIN, 1979) index, Dunn's index (CALIŃSKI; HARABASZ, 1974) and silhouette width criteria (ROUSSEEUW, 1987).

The choice of validity criterion used in this work is based on the comparative review conducted by VENDRAMIN; CAMPELLO; HRUSCHKA (2010), where the performances of forty validity criteria were compared using five different clustering algorithms and one thousand and eighty datasets were analyzed. As a result, the PBM index (PAKHIRA; BANDYOPADHYAY; MAULIK, 2004) validity criterion excels with other criteria performance, electing the best partition correctly in 92.59% of the assessed scenarios.

The PBM index is a criterion based on within-group and between-group distance and it's defined as follows:

$$PBM(k) = (\frac{1}{k} \times \frac{E_1}{E_k} \times D_k)^2 \tag{3.5}$$

Where k is the number of clusters, E_1 is the sum of distances between the data objects and the mean of the data, E_k represents the sum of distances within-group and D_k is the maximum separation between a pair of centroids (PAKHIRA; BANDYOPADHYAY; MAULIK, 2004). This way, the best partition is indicated when the PBM value is maximized.

Based on the elbow analysis of the suggested k, the PBM index criterion was applied to $K = 2, 3, 4$, which resulted in seventy-two PBM index values that represented a possible partition approach [9] (twelve sets of features, for each number of clusters for each algorithm used). From these, six sets of PBM index values were selected through a basic comparison, three for the Ward's AHC method and three for the K-means method, the Ward's AHC's example can be seen in Table 5.

In the column Features on Table 5, each row represents the features considered to partition the dataset (e.g., the feature difNLOC represents a partitioning where only this feature was considered to classify the students' tinkering behaviors). These features combinations were obtained from all possible combinations of the four selected features, without repetition of features and independent of order. Each value in the column PBM represents the assessed quality of the partitioning based on those features.

In cases where two partitionings present the same amount of features (e.g., a partitioning based only on time-on-task and another based only on total time), a simple comparison

[9] During the combination of all features, the combinations of difNLOC and submissions were not considered as the difNLOC feature was created from the feature submission and presented a high (negative) correlation 3.3.2

Table 5 – PBM's index values for each combination of features, using Ward's AHC with K= 2, 3 and 4

Features\PBM's value by number of clusters (k)	*k=2*	*k=3*	*k=4*
difNLOC	0.21414755	0.401072118	0.641710265
submissions	0.30918171	0.516417557	0.553049953
total time	0.434243089	1.80446314	1.738226701
time on-task	0.099257515	0.515387703	0.462405834
difNLOC, total time	0.254792653	0.346595971	0.290321767
difNLOC, time on-task	0.220398289	0.361947906	0.357609706
submissions, total time	0.287605601	0.295234193	0.270144753
submissions, time on-task	0.236711128	0.237493524	0.255021056
total time, time on-task	0.217092525	0.227943866	0.213702323
difNLOC, total time, time on-task	0.113164345	0.209178153	0.200033135
submissions, total time, time on-task	0.135713287	0.179884334	0.1493109
difNLOC, submissions, total time, time on-task	0.148334517	0.186329413	0.161318606

Source: Elaborated by the author

of values indicated which partitioning had better performance. However, in cases with a different number of features (dimensions of the dataset), a normalization process was necessary to mitigate the bias present when comparing distances in two different dimensions (DY; BRODLEY, 2000).

$$NormVal(C_n) = crit(F_n, C_n) \times crit(F_n, C_m) \tag{3.6}$$

$$NormVal(C_m) = crit(F_m, C_m) \times crit(F_n, C_m) \tag{3.7}$$

Where $crit(F,C)$ is the criterion used to assess the partitioning quality, in the case of this work, the PBM's index, $crit(Fn,Cm)$ is the criterion value obtained by using the set of features Fn selected and Cm the clustering assignment of each data object in the dataset, and NormVal is the normalized value of the equation. The recommended clustering-Features pair is defined by the greater NormVal obtained. In the case where both NorVals were equal, it was recommended the clustering-features pair with lower dimensionality (DY; BRODLEY, 2000).

This heuristic normalization was used in this work through the following steps (Figure 11):

1. As a pre-selection step, initial values were gathered through a comparison of PBM values obtained by partitionings with the same number of features (dimensions). This process was done separately for each algorithm (Table 6).
2. In order to assess the best partitions obtained by each clustering algorithm (K-means and Ward's AHC) a normalization between the selected PBM's values presented in the previous step was done, and a new feature's ranking was created for each algorithm (Table 7);

Table 6 – Selected features, PBM values and number of clusters in the pre-normalization phase

Features	PBM	K
Ward's AHC		
difNLOC	0.641710265	4
difNLOC, time on-task	0.361947906	3
difNLOC, total time, time on-task	0.209178153	3
K-means		
difNLOC	0.626860118	4
difNLOC, total time	0.361246041	3
difNLOC, total time, time on-task	0.252435065	3

Source: Elaborated by the author

Table 7 – Ward's PBM index values of crit(Fn,Cm) and crit(Fn,Cn) in highlights

	difNLOC	**difNLOC, time on-task**	difNLOC, total time, time on-task
difNLOC	**0.641710265**	0.1782833	0.124075125
difNLOC, time on-task	0.216003074	**0.361947906**	0.273695061
difNLOC, total time, time on-task	0.097313622	0.100940077	**0.209178153**

Source: Elaborated by the author

As a result, in this step were selected the features and number of clusters by each algorithm that achieved the best performance partitioning the dataset. A PBM value of 0.64171 was achieved by the the Ward's AHC algorithm using the difNLOC feature and the students' behaviors partitioned in four clusters. For the K-means algorithm, a PBM value of 0.36124 was selected, using the difNLOC and the total time features to partition the dataset in three clusters representing the different students' tinkering behaviors in the dataset.

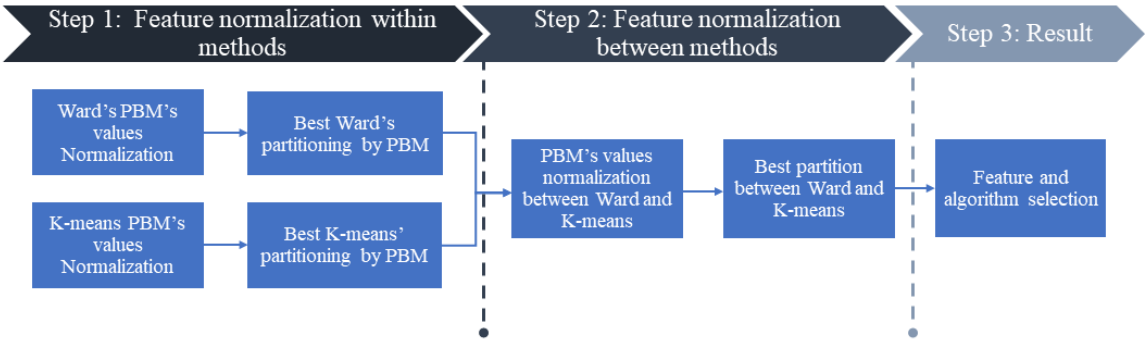
To compare the best partitionings obtained between the clustering algorithms, a new normalization was made using the PBM's values. As a result, the Ward's normalized PBM value equals 0.07727, and the K-means' normalized PBM value obtained was equal to 0.03825.

3. Comparing the normalized values obtained in the previous step (**Ward's PBM = 0.07727 > Kmeans PBM = 0.03825**), the **Ward's AHC** algorithm was chosen as the clustering method in this work.

3.4 Final Remarks

In this chapter were presented the Research Questions elaborated and the methodology adopted during the development of this work. Then, it was presented a brief description of the data source and systems used, as well as the creation of the dataset, the feature's extraction, and the algorithm selection process as part of the chosen methodology.

Figure 11 – PBM index values' normalization process



Source: Elaborated by the author

Furthermore, this chapter describes and proposes a set of features that could be used to describe tinkering behavior in students during online programming assignments. The analysis of the proposed classification of possible behaviors that may occur between those discussed by TURKLE; PAPERT (1992), as well as the impact of tinkering behavior on the students' grades performance, will be present in the next chapter.

CHAPTER

4

RESULTS

Based on the partitionings settings defined in the previous chapter (Chapter 3), this chapter presents the students' tinkering behaviors found through the clustering process in the dataset, an analysis of the behaviors' characteristics and its impact on the students' grades. Section 4.1 presents the clustering results of the chosen partitioning method and a brief description of each cluster based on the selected features. In section 4.2, an analysis of the data is presented combined with the association rule mining process. In addition, the hypothesis tests and discussions are presented in this section. Lastly, section 4.3 discuss this work's threats to validity and the steps made to mitigate them.

4.1 Partitioning result

During the process presented in Section 3.3.4, seventy-two partitionings were analyzed to find the best partitioning to identify students' tinkering behaviors in the dataset. From these, the approach using Ward's AHC algorithm presented the best partitioning quality dividing the students by their average number of lines of code added or removed at each assignment's submission (difNLOC feature). This approach partitioned the dataset into four behavior clusters according to this feature and was chosen as the data mining approach to be employed in this work (Figure 12).

As an answer to the **RQ1:** - *Which students' tinkering behaviors can be identified using the selected assignment features?* - the difNLOC feature presented as the best feature to identify the students' tinkering behaviors while using the Ward's AHC. The difNLOC feature (section 3.3.2) represents the average number of lines of code that the students change at each submission and the number of students' submissions during a given assignment. This way, this feature also represents the students' interactions with the system, the amount of feedback that the students received to finish their assignment, and, consequently, the tinkering behaviors expressed by the students during an assignment.

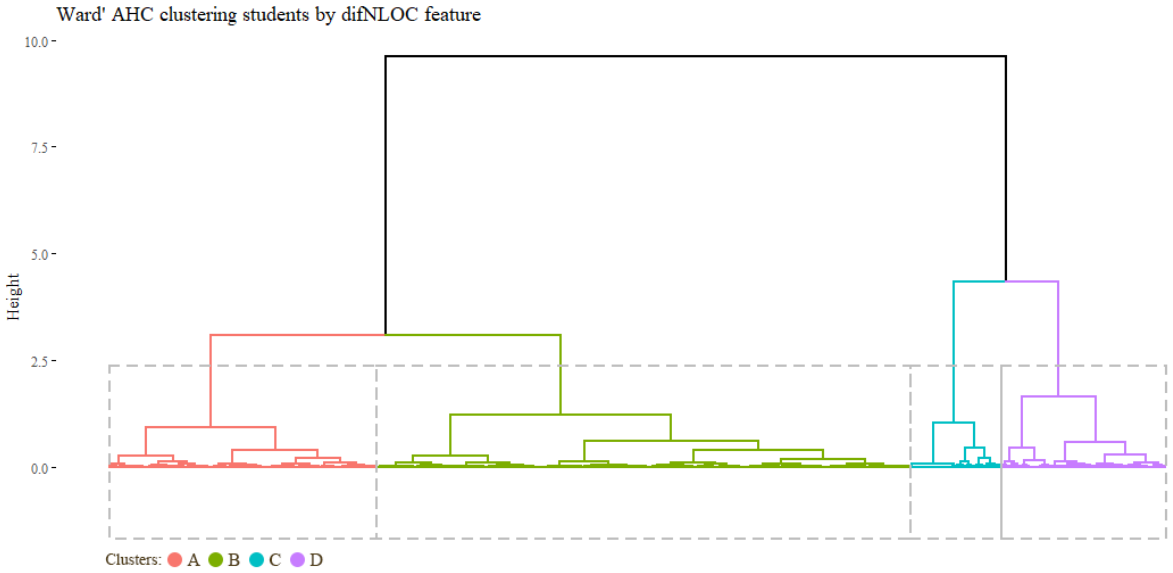


Figure 12 – Students' difNLOC Ward's clustering Dendrogram

Source: Elaborated by the author

Four different tinkering behaviors were identified as clusters in the dataset, accordingly with the following distribution: cluster A represented 50.47% of the dataset entries, cluster B represented 25.34% of the dataset and clusters C and D represented 15.56% and 8.63% of the dataset respectively (Figure 13).

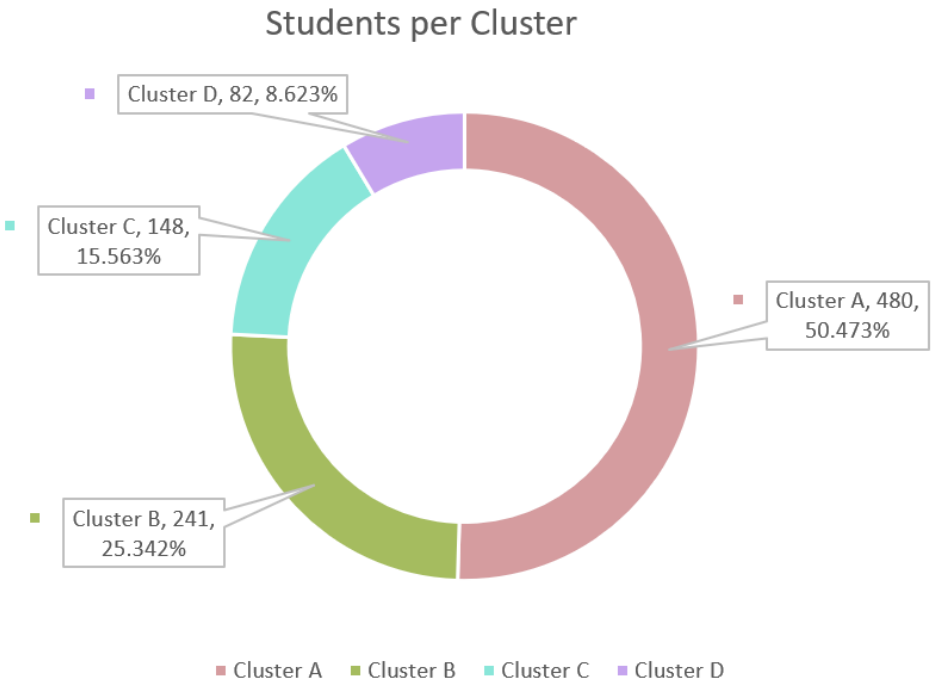


Figure 13 – Distribution of students per clusters

Source: Elaborated by the author

Table 8 – difNLOC range values by cluster

Cluster	min. value	max value
Tinkerers (A)	0	0.16318
Fixers (B)	0.165272	0.341004
Adjusters (C)	0.352364	0.7106871
Planners (D)	0.725057	1

Source: Elaborated by the author

As defined in section 3.3.2, all features used in this work were normalized to express relative values between zero and one, where zero represents the lowest value of a feature for a given assignment and one the highest value. This way, a student that expresses a higher tinkering would have a lower difNLOC value, whereas a student that employed a more planned approach (lower tinkering) would have a higher difNLOC value.

To achieve the specific objective three proposed in this work - *"Identify students' problem-solving behaviors and its relation to the Turkle and Papert's definition of Tinkerers and Planner"* -, in addition to the identification of tinkering behaviors, it was analyzed the assigned range values of the difNLOC feature for each cluster and its relation with Turkle and Papert's definition.

Observing Table 8, it can be seen that the clusters A and D presented the most extreme values and, therefore, best express the dichotomous definition of tinkerers and planners (respectively) made by Turkle and Papert (TURKLE; PAPERT, 1992). Albeit clusters B and C still express similar characteristics to tinkerers and planner, both clusters presented more intermediate values, and yet, are distinct enough to be defined as different clusters from A and D. With that, the four identified behaviors are **Tinkerers** (cluster A), **Fixers** (cluster B), **Adjusters** (cluster C) and **Planners** (cluster D)

In the tinkerers cluster were assigned students with lowest values of difNLOC (from 0 to 0.163), which indicates that the students needed a higher number of feedback messages from the system in order to complete their assignments. Even though students in this cluster presented significant changes in their lines of code between some submissions, they also tended to make submissions to test small changes in their codes, Figure 14 and Figure 15 demonstrate alterations between submissions of a tinkerer student in the online assignment 528 - Power sequences.

Fixers composed the second biggest cluster with two hundred and forty-one student (241) entries, and, similarly to tinkerers, this cluster is characterized by low values of difNLOC (between 0.1652 and 0.3410). Even though these students also presented high amounts of tinkering, their difNLOC values indicate that overall, the students in this cluster needed a smaller number of feedback messages from the system in order to complete their assignments compared with tinkerers (Figure 16).

The adjusters had the biggest difNLOC feature's range (between 0.352364 and 0.710681) and yet, the second smallest number of students' entries (148). The higher values of difNLOC

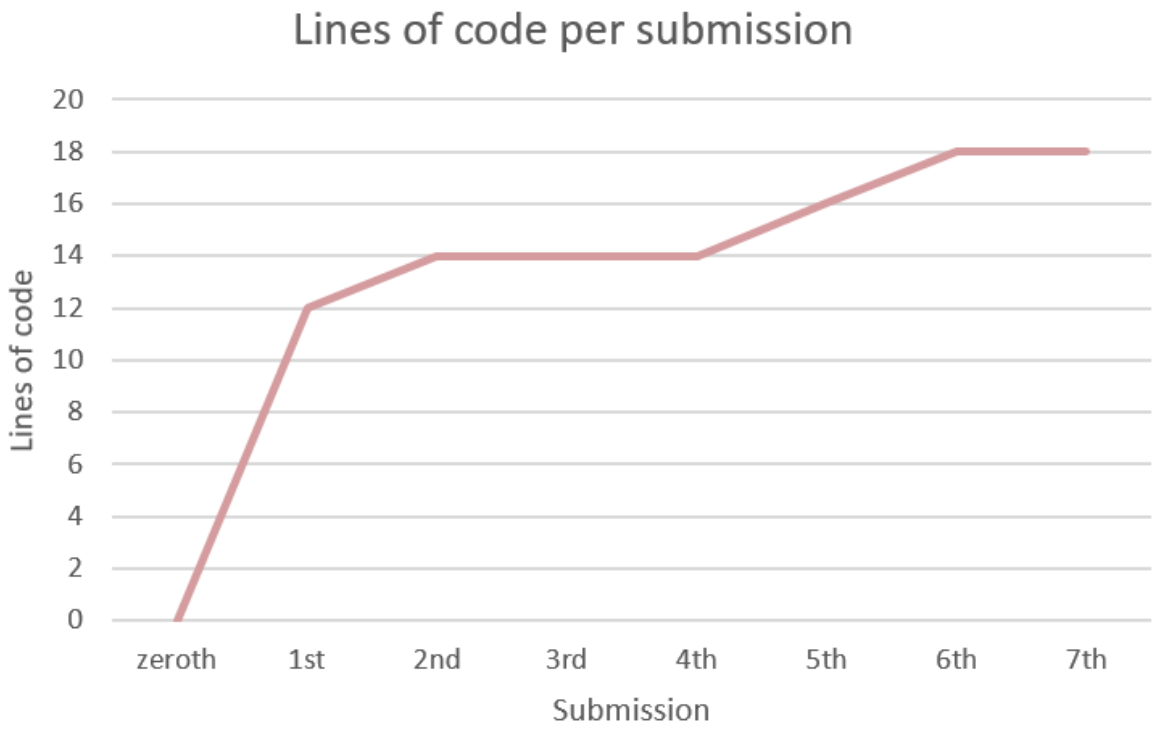


Figure 14 – Example of a tinkerer student - Lines of code per submission

Source: Elaborated by the author

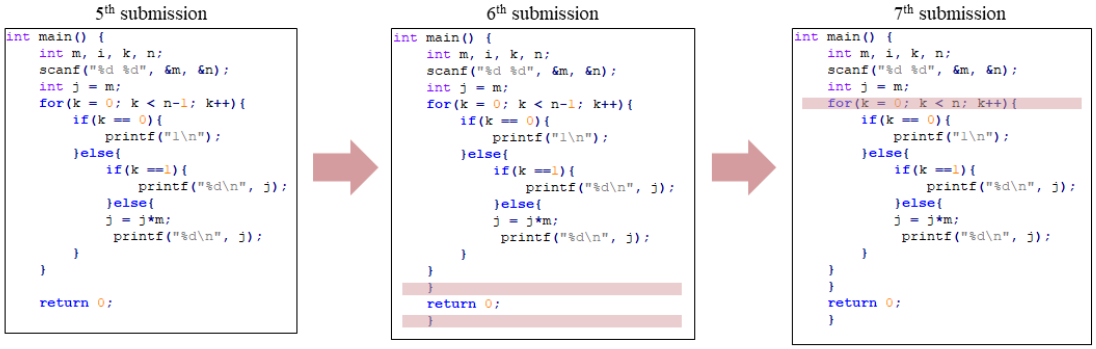


Figure 15 – Example of a tinkerer student - changes during submissions, two lines of code added and a change of a parameter on the *for* function

Source: Elaborated by the author

indicate that these students needed a small amount of feedback from the system to finish their assignments and that they tend to make significant changes in their codes at each submission (Figure 18).

Planner students, as the name suggests, were the closest to the planner's definition. With the highest values in the difNLOC feature (between 0.725057 and 1), these students usually

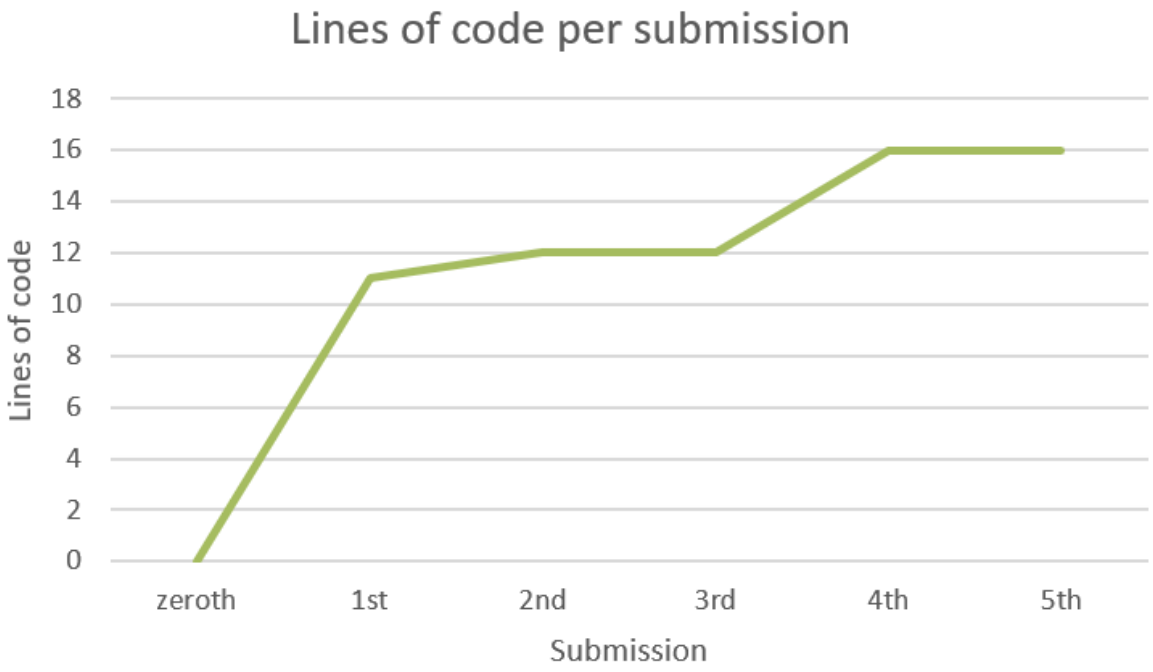


Figure 16 – Example of fixer student - Lines of code per submission

Source: Elaborated by the author

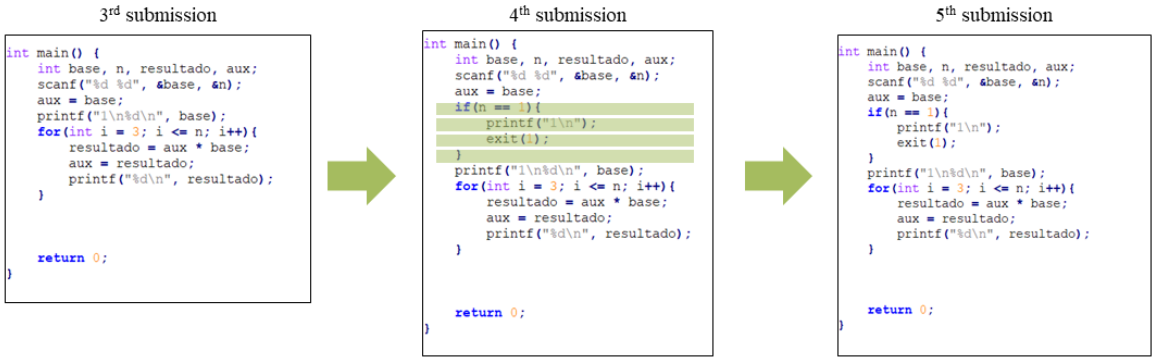


Figure 17 – Example of a fixer student - changes during submissions, four lines of code added and no changes between the 4th and 5th submissions

Source: Elaborated by the author

didn't receive any feedback from the system, completing the assignment in a single submission, which indicates a more planned approach to the problem.

4.2 Data analysis and discussion

The four tinkering behaviors were obtained using the difNLOC feature. The analysis of the students' behaviors and how they related with all the extracted features are divided into the

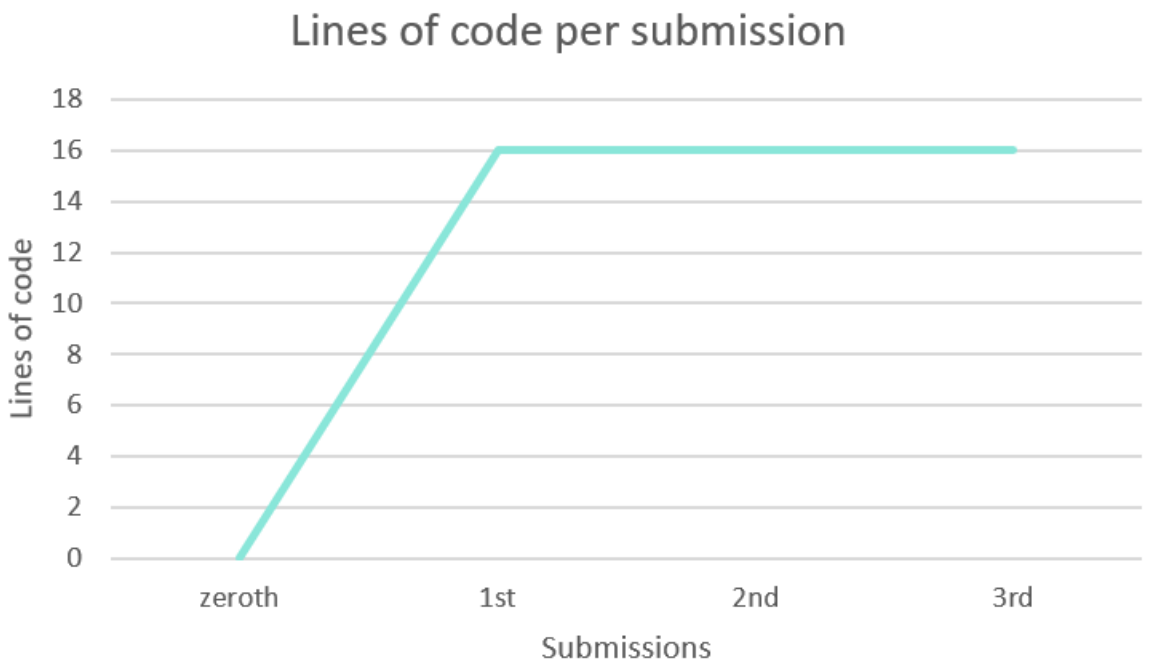


Figure 18 – Example of a fixer student - Lines of code per submission

Source: Elaborated by the author

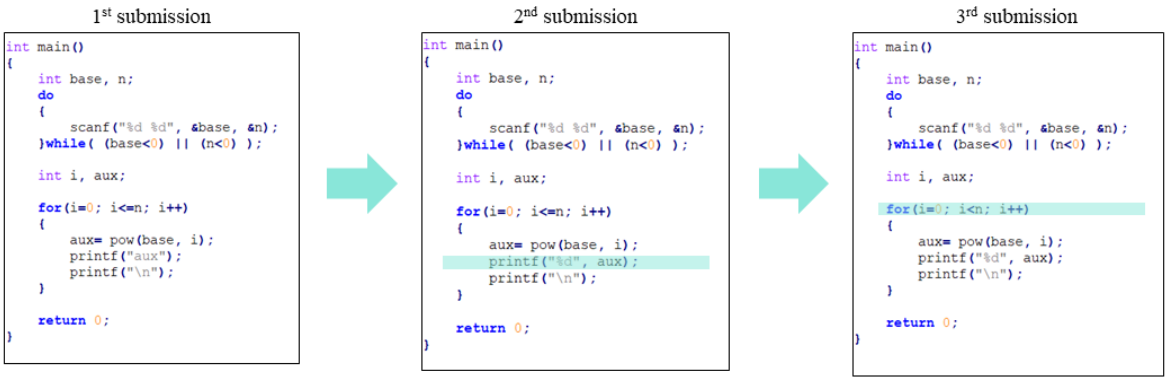


Figure 19 – Example of a fixer student - changes during submissions, no lines added/removed, with only two lines edited, a parameter in the *print* function and a equal sign in the *for* function

Source: Elaborated by the author

following parts: Subsection 4.2.1 presents the results of the association rule mining technique *apriori* used in this work, and discuss the rules obtained and how they can be used to describe the students' tinkering behaviors. The following subsection 4.2.2 presents the hypothesis' tests defined in Section 3.1.

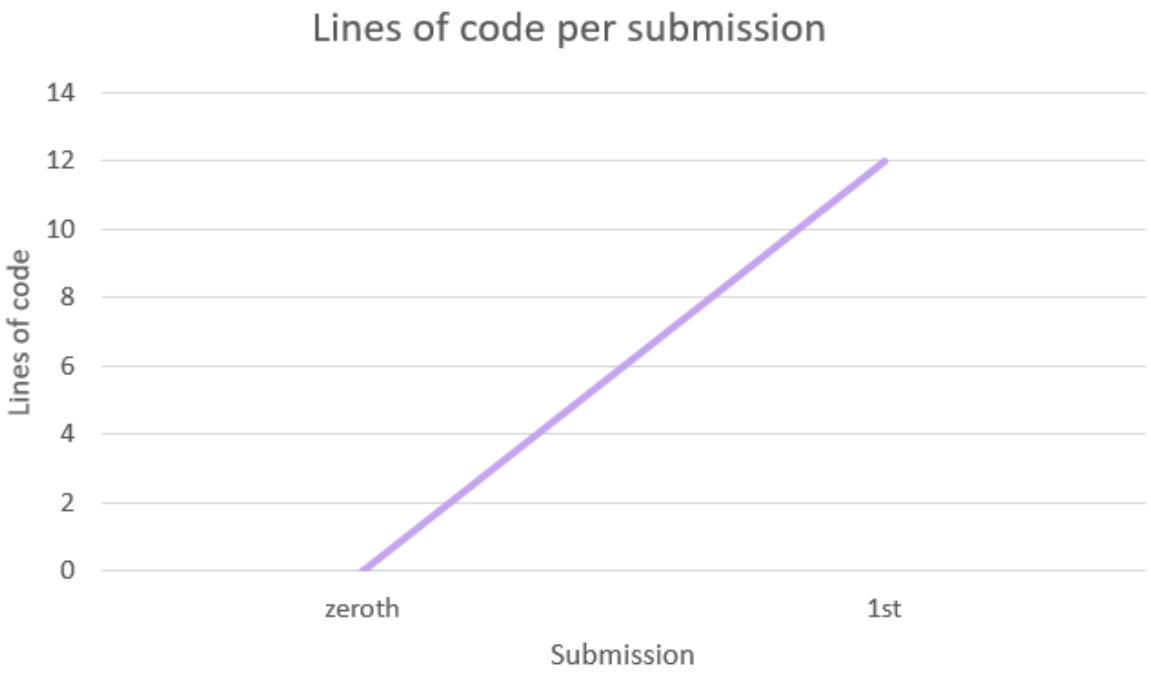


Figure 20 – Example of a planner student - Lines of code per submission

Source: Elaborated by the author

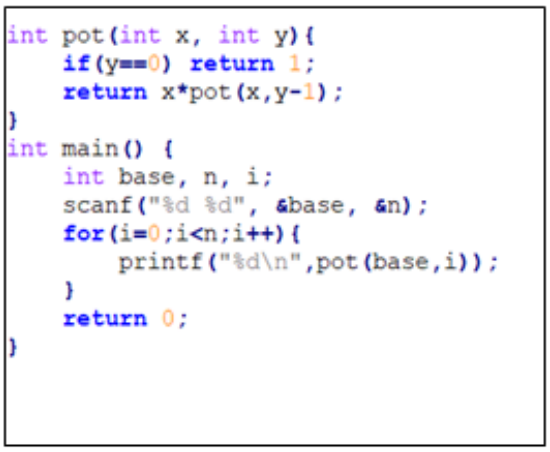
4.2.1 Cluster description through association rule mining

As discussed in section 2.2.3, association rule mining techniques are used to discover associations and correlations among larges amounts of elements in a dataset. However, In order to extract the most relevant rules related to the tinkering behaviors, a discretization of the continuous features in the dataset was necessary. In this work it was used the entropy-based method, infoGain[1] (FAYYAD; IRANI, 1993), that divided each feature of the dataset by intervals that are best aligned with the cluster that represent the students' tinkering behaviors (Table 9. After the discretization process the apriori algorithm (AGRAWAL; IMIELIŃSKI; SWAMI, 1993) was applied to the students' dataset using minimum confidence ≥ 0.8 and minimum support ≥ 0.1 [2]. As a result, it was obtained a set of twenty-one rules that fit within the parameters and expressed which set of features (left-hand rules) implied in one of the clusters (right-hand rules). The rules and the values obtained can be seen in Table 10.

After analyzing the obtained rules, it can be observed that the students that expressed Tinkerers and Fixer's behaviors tend to take relatively longer to finish their assignments, both in time-on-task spent and total time, with rules 8 and 20 (confidence 1) are strong rules in support of this observation. In addition, as expected due the difNloc nature, tinkerers and fixers expressed the first and second-highest number of submissions, respectively. It can also be observed in

[1] Using the OneR's optbin function available in <https://rdrr.io/cran/OneR/man/optbin.html>
[2] Using the arules R package v1.6-4, available in <https://www.rdocumentation.org/packages/arules>

1st submission



```
int pot(int x, int y){
    if(y==0) return 1;
    return x*pot(x,y-1);
}
int main() {
    int base, n, i;
    scanf("%d %d", &base, &n);
    for(i=0;i<n;i++){
        printf("%d\n",pot(base,i));
    }
    return 0;
}
```

Figure 21 – Example of a planner student - only one submission

Source: Elaborated by the author

Table 9 – Features' discretization values

Feature	1st interval	2nd interval	3rd interval	4th interval
difNLOC	(-0.001, 0.164]	(0.164, 0.347]	(0.347, 0.718]	(0.718, 1]
submissions	(-0.001, 0]	(0, 0.0167]	(0.0167, 0.0682]	(0.0682, 1]
total time	(-0.001,3.0.0000333]	(0.0000333 - 0.000081]	(0.000081, 0.000517]	(0.000517, 1]
time on-task	(-0.001, 0.000234]	(0.000234, 0.014]	(0.014, 0.11]	(0.11, 1]

Source: Elaborated by the author

Table 10 through rules 1, 16, and 21, where each of the three behaviors (Tinkerers, Fixers, and Adjusters) are divided by their difNLOC feature. It's worth pointing that the apriori algorithm didn't generate any strong rules related to planners that would fit in the defined minimum confidence (≥ 0.8), presumably due to the low number of data objects present in their cluster.

4.2.2 Hypothesis test

In order to test the hypothesis formulated Section 3.1, the Kruskal Wallis test (KRUSKAL; WALLIS, 1952) was used. Similarly to the one-way ANOVA (*Analysis of Variance*) test, the Kruskal Wallis test is recommended to assess for significant differences on a continuous dependent variable (e.g., students' grades, total time, and time-on-task) by a categorical independent variable (Students' tinkering behaviors (clusters)). However, differently from the ANOVA test, the Kruskal Walis test does not assume normality of the data. Table 11 shows the Shapiro-Wilk normality test for the students' assignments features (difNLOC, number of submissions, total time and time-on-task) and Table 12 shows the normality tests for the students' grades in the

Table 10 – Rules obtained using apriori algorithm

Association Rules					
ID	**Features**	**Behavior**	**Support**	**Confidence**	**lift**
1	{difNLOC=(-0.001, 0.164]}	{Tinkerer}	0.504732	1	1.98125
2	{difNLOC=(-0.001, 0.164], submissions=(0.0682, 1]}	{Tinkerer}	0.432177	1	1.98125
3	{difNLOC=(-0.001, 0.164], total time=(0.000517, 1]}	{Tinkerer}	0.353312	1	1.98125
4	{difNLOC=(-0.001, 0.164], time on-task=(0.11, 1]}	{Tinkerer}	0.325973	1	1.98125
5	{difNLOC=(-0.001, 0.164], submissions=(0.0682, 1], total time=(0.000517, 1]}	{Tinkerer}	0.320715	1	1.98125
6	{difNLOC=(-0.001, 0.164], submissions=(0.0682, 1], time on-task=(0.11, 1]}	{Tinkerer}	0.309148	1	1.98125
7	{difNLOC=(-0.001, 0.164], total time=(0.000517, 1], time on-task=(0.11, 1]}	{Tinkerer}	0.271293	1	1.98125
8	{difNLOC=(-0.001, 0.164], submissions=(0.0682, 1], total time=(0.000517, 1], time on-task=(0.11, 1]}	{Tinkerer}	0.255521	1	1.98125
9	{difNLOC=(-0.001, 0.164], time on-task=(0.014, 0.11]}	{Tinkerer}	0.162986	1	1.98125
10	{difNLOC=(-0.001, 0.164], submissions=(0.0682, 1], time on-task=(0.014, 0.11]}	{Tinkerer}	0.118822	1	1.98125
11	{difNLOC=(-0.001, 0.164], total time=(8.1e-05, 0.000517]}	{Tinkerer}	0.11041	1	1.98125
12	{submissions=(0.0682, 1], total time=(0.000517, 1], time on-task=(0.11, 1]}	{Tinkerer}	0.255521	0.945525	1.873322
13	{submissions=(0.0682, 1], time on-task=(0.11, 1]}	{Tinkerer}	0.309148	0.933333	1.849167
14	{submissions=(0.0682, 1], total time=(0.000517, 1]}	{Tinkerer}	0.320715	0.918675	1.820124
15	{submissions=(0.0682, 1]}	{Tinkerer}	0.432177	0.878205	1.739944
16	{difNLOC=(0.164, 0.347]}	{Fixer}	0.253417	1	3.946058
17	{difNLOC=(0.164, 0.347], submissions=(0.0167, 0.0682]}	{Fixer}	0.178759	1	3.946058
18	{difNLOC=(0.164, 0.347], time on-task=(0.014, 0.11]}	{Fixer}	0.15142	1	3.946058
19	{difNLOC=(0.164, 0.347], total time=(0.000517, 1]}	{Fixer}	0.113565	1	3.946058
20	{difNLOC=(0.164, 0.347], submissions=(0.0167, 0.0682], time on-task=(0.014, 0.11]}	{Fixer}	0.109359	1	3.946058
21	{difNLOC=(0.347, 0.718]}	{Adjuster}	0.155626	1	6.425676

Source: Elaborated by the author

Table 11 – Shapiro-Wilk test for the students' assignments features

Shapiro-Wilk normality test		
Feature	**W**	**p-value**
difNLOC	0.81066	2.2e-16
Number of submissions	0.6782	2.2e-16
total time	0.60632	2.2e-16
time on-task	0.72197	2.2e-16

Source: Elaborated by the author

different assignments [3].

In this work, the Kruskal-Walis test was used to assess significant differences ($p < 0.05$) between the amount of time that the students dedicated for the online assignments (total time and time-on-task), as well the differences between five different students' grades (in-class pen-and-paper assignments, online assignments, first and second pen-and-paper Tests, average test's grade and students' final grades). The results for each hypothesis and research questions are

[3] The Kruskal-Wallis and Shapiro-Wilk tests were made using R package stats v3.6.0. Available at <https://www.rdocumentation.org/packages/stats/versions/3.6.1>

Table 12 – Shapiro-Wilk test for the students' assignments grades

Shapiro-Wilk normality test		
Assignments	**W**	**p-value**
First in-class Test	0.96155	4.028e-15
Second in-class Test	0.55821	2.2e-16
Average of in-class Tests	0.87938	2.2e-16
Average of class assignments	0.96744	9.512e-14
Average of online assignments	0.86509	2.2e-16
Final grades	0.91002	2.2e-16

Source: Elaborated by the author

presented below.

The first hypothesis to be tested refers to the **RQ2** - *Is there a grades' performance difference between the identified students' tinkering behaviors?*. Table 13 present the tests' results to identify significant grades differences between the behaviors.

Table 13 – Kruskal-Wallis test for significant differences between assignments' grades between the four behavior clusters. ** indicates statistical significance

Kruskal-Wallis		
Assignments	**chi-squared**	**p-value**
First in-class Test	1.825	0.6095
Second in-class Test	35.335	1.035e-07**
Average of in-class Tests	6.1569	0.1042
Average of in-class assignments	2.5903	0.4592
Average of online assignments	4.8248	0.1851
Final grades	5.906	0.1163

Source: Elaborated by the author

As can be seen in the column p-value, there's a statistically significant difference only in the in-class Test 2 (final exam) assignment with a p-value of < 0.05 (0.0000001035). However, in order to identify between which clusters the difference was present, a pairwise post-hoc comparison was made using the Dunn's Test (DUNN, 1961) with Bonnferroni adjustment method (BLAND; ALTMAN, 1995) (Table 14).

Table 14 shows the z-statistics and p-value results of the pairwise comparisons related to students' grades in the in-class Test 2 (final exam), the values in column p-value shows that there's a difference ($p < 0.05$) between the tinkerers and fixers, tinkerers and adjusters, and tinkerers and planners. This way, it's possible to **refute the null hypothesis** (i.e., There is no difference between students' grades performance between the identified tinkering behaviors) that students would have similar grades. As can be seen in Figure 22, students in the tinkerers' cluster had the best performance in the final exams for that subject when comparing to other clusters.

Table 14 – Pairwise comparisons of the in-class Test 2 assignment using Dunn's Test. *indicates difference between the clusters

Pairwise comparisons		
Pair of clusters	**Z-statistics**	**p-value**
Tinkerers - Fixers	3.617854	(0.0009)*
Tinkerers - Adjusters	2.761884	(0.0172)*
Fixers - Adjusters	-0.215615	(1.0000)
Tinkerers - Planners	3.964119	(0.0002)*
Fixers - Planners	1.371767	(0.5104)
Adjusters - Planners	1.433449	(0.4552)

Source: Elaborated by the author

So, to answer the **RQ2** - *Is there a grades' performance difference between the identified students' tinkering behaviors?*. Yes. It was observed a difference in the students' grades performance by students that expressed tinkerers' behaviors in the final exam.

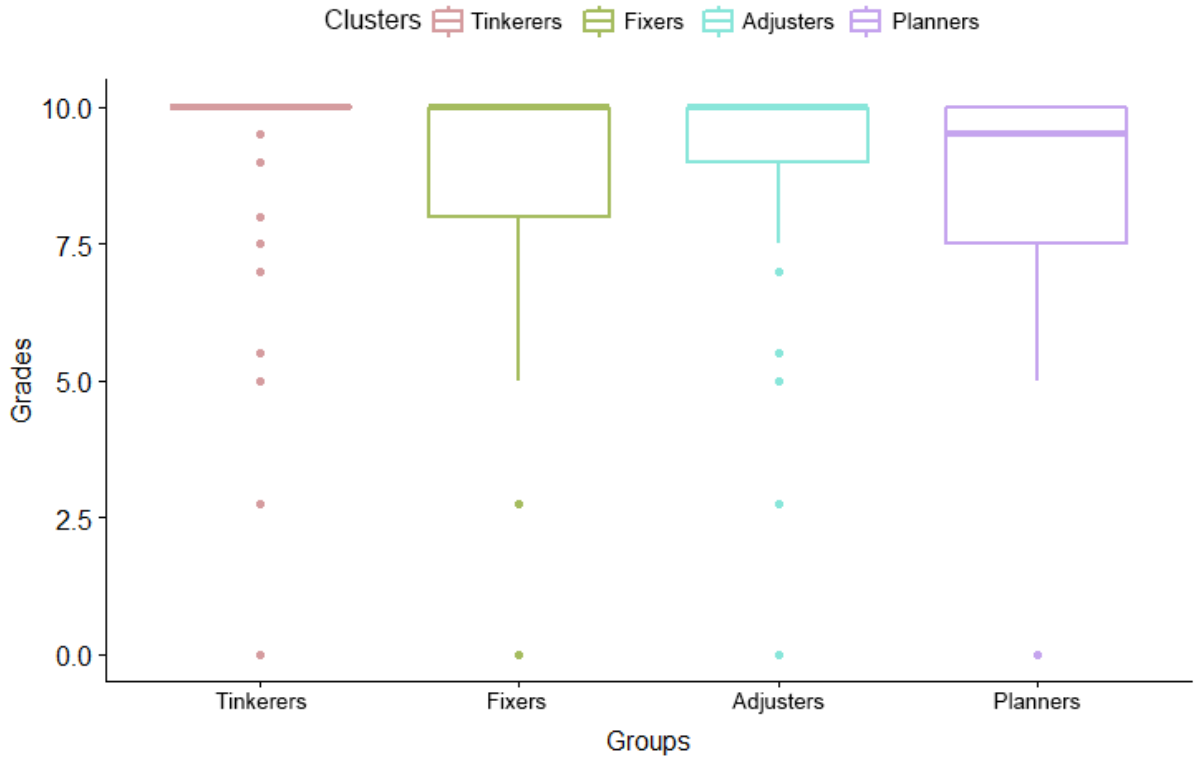


Figure 22 – Boxplot of the students' final test grades by clusters

Source: Elaborated by the author

Similarly to the previous question, the answer to the **RQ2.1** - *Is there a significant difference in performance among the tinkering behaviors identified depending on the assignment format (pen-and-paper class assignment, online assignments, and pen-and-paper tests)?* can be seen in Tables 13 and 14, where only in the in-class Test 2 (final exam), a pen and paper

test, the students presented a significant difference. As an answer: Yes, the students presented different performances among the assignment types, where students with the highest tinkering (tinkerers) outperformed the other clusters. It's worth notice that, even though a significant statistical difference was observed only in the final exam, these exams comprehend all the content seen during the semester. As a pen-and-paper activity, the students could not employ their tinkering strategies, which show the impact that those strategies applied in online environments might have on the students' learning.

The **RQ3** - *Is the total time to complete an online assignment different among the identified tinkering behaviors?* - seeks to evaluate how much time did the students that fit in the identified behaviors spend online to complete an online assignment. This way, a Kruskal-Wallis test was executed comparing the time-on-task feature among the four tinkering behaviors (tinkerers, fixers, adjusters, and planners), which indicated the existence of a difference between the behaviors ($chi-squared = 244.82$ and $p-value = 2.2e-16$). Table 15 shows the z-statistic and p-values results of the pairwise comparisons related to the amount of time spent online in an assignment (time-on-task feature) by the students.

Complementing the results obtained in Table 10, the statistical results show that only two clusters that didn't present a significant difference ($p < 0.05$) between them, adjusters, and planners. In this dataset, the students with the highest tinkering (tinkerers) spent the largest amount of time in sessions to finish the online assignments, followed by fixers and adjusters, while students with the lowest tinkering (planners) spent the least amount of time in sessions to finish their online assignments (Figure 23). Based on this, **RQ 3's null hypothesis can be rejected** (i.e., There is a difference in the total time to complete an online assignments among the identified students' tinkering behaviors.).

To answer the **RQ3** - *Is the total time to complete an online assignment different among the identified tinkering behaviors?*-: Yes. The obtained results show that students that expressed Tinkerers and Fixers tend to spend more time to complete online assignments. This difference can be due to several factors, such as students' previous knowledge of the programming language, online environment, studied topic, or students' interest in the subject. In order to better understand the nature of these differences, a larger study is necessary, with a higher number of participants and evaluating the students' previous knowledge and interest.

Similarly to the previous research question, the **RQ 4** - *Is the total time that the students spend on the system to complete an online assignment different among the identified tinkering behaviors?* - seeks to evaluate the amount of time that a student that fit the identified behaviors spend to complete an online assignment. However, it differs from the previous research question by accounting not only the online time during the sessions but also the time between the first and last sessions that a student takes to complete an assignment. So, a Kruskal-Wallis test was executed comparing the total time feature among the four tinkering behaviors (tinkerers, fixers, adjusters, and planners), which indicated the existence of a difference between the clusters

Table 15 – Pairwise comparisons of the students' amount of time spent online to finish an assignment using Dunn's Test. *indicates difference between the clusters

Pairwise comparisons		
Pair of clusters	**Z-statistics**	**p-value**
Tinkerers - Fixers	9.742636	(0.0000)*
Tinkerers - Adjusters	11.87611	(0.0000)*
Fixers - Adjusters	3.327089	(0.0026)*
Tinkerers - Planners	10.81403	(0.0000)*
Fixers - Planners	4.091173	(0.0001)*
Adjusters - Planners	1.275425	(0.6065)

Source: Elaborated by the author

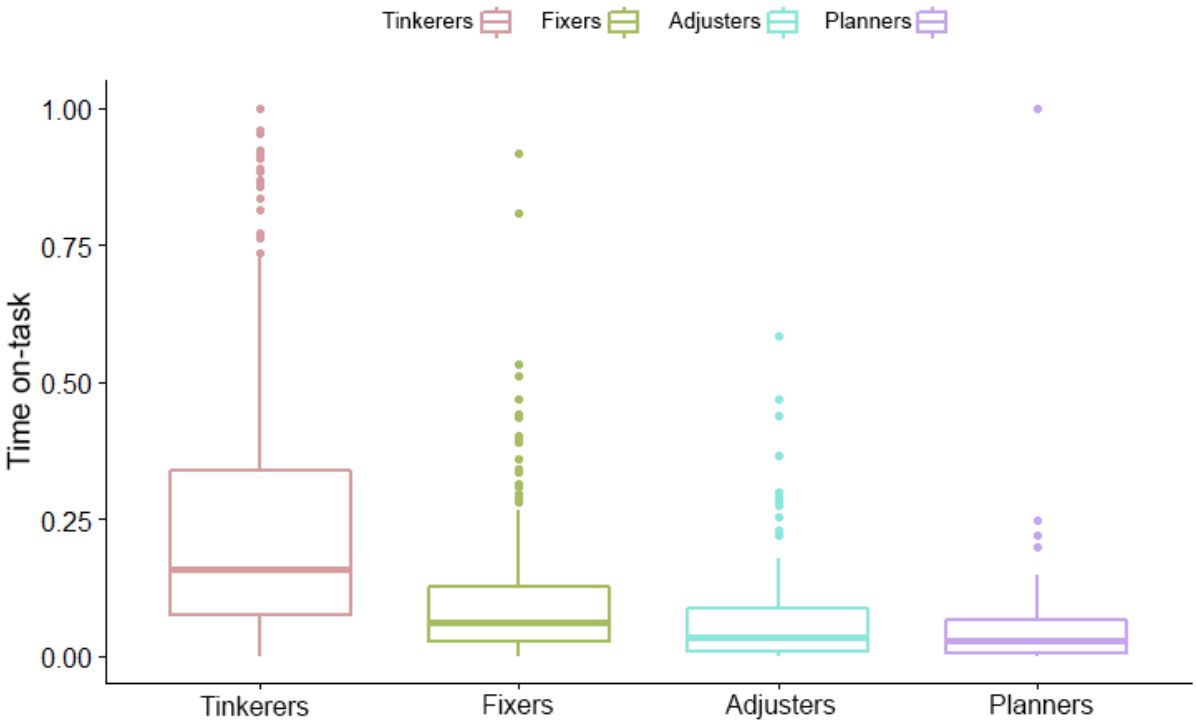


Figure 23 – Boxplot of the students' time on-task online to complete an online assignment

Source: Elaborated by the author

($chi - squared = 151.11$ and $p - value = 2.2e - 16$). The z-statistic and p-values results of the pairwise comparisons of the amount of the time that a student takes to complete an assignment (total time feature) can be seen in Table 16.

Analogous to the results obtained in **RQ3**, the statistical results show that only the same two clusters that didn't present a significant difference ($p < 0.05$) between them are the Adjusters and Planners. In this dataset, the students that expressed a Tinkerers behaviors took the largest amount of time to finish the online assignments, followed by Fixers and Adjuster students, while students with the lowest tinkering behavior (Planners) spent the least amount of time to finish

their online assignments (Figure 24). This way, the **null hypothesis can be rejected** (i.e., There is a difference in the total time spend on the system to complete the online assignments among the identified students' tinkering behaviors).

Table 16 – Pairwise comparisons of the time on-task using Dunn's Test. *indicates difference between the clusters

Pairwise comparisons		
Pair of clusters	**Z-statistics**	**p-value**
Tinkerers - Fixers	6.743988	(0.0000)*
Tinkerers - Adjusters	9.323567	(0.0000)*
Fixers - Adjusters	3.295881	(0.0029)*
Tinkerers - Planners	9.019820	(0.0000)*
Fixers - Planners	4.265923	(0.0001)*
Adjusters - Planners	1.461384	(0.4317)

Source: Elaborated by the author

To answer the **RQ4** - *Is the total time that the students spend on the system to complete an online assignment different among the identified tinkering behaviors?* - Yes. The statistical results show that tinkerers students take more time between their first and last sessions to complete their online assignments. This difference can be due to several factors, such as the availability of different sources of information that students may resort to answering their questions (e.g., subject's class or special tutoring times), students' previous knowledge, or even access to the online environment. In order to better understand the nature of these differences, a larger study is necessary, with a higher number of participants and evaluating the students' previous knowledge, interest, and access to the educational systems.

After analyzing the time characteristics of each cluster, a formal answer to the **RQ1.1** of this work - *How can the differences and similarities in student's tinkering behaviors be characterized based on the selected features?* can be made: As discussed above, this work proposes a series of comparisons to select the features and methods that best identify students' tinkering behaviors in the dataset. As a result, a clustering approach using the Ward's AHC method and the students' average line of code's changes feature was selected. This approach uses (dis)similarity measures to partition students in four clusters that presented similar behaviors related to tinkering. Furthermore, statistical analysis of the times that the students in each cluster take to finish their online assignment, as well as an association rule mining was presented in order to deepen the descriptions and characteristics of the behaviors. This way, the identified tinkering behaviors can be characterized as follows:

- **Tinkerers:** Students that tend to make small-to-no changes in the total number of lines of code at each submission (presented the lowest value of difNLOC feature) and usually takes more time to complete their online assignments

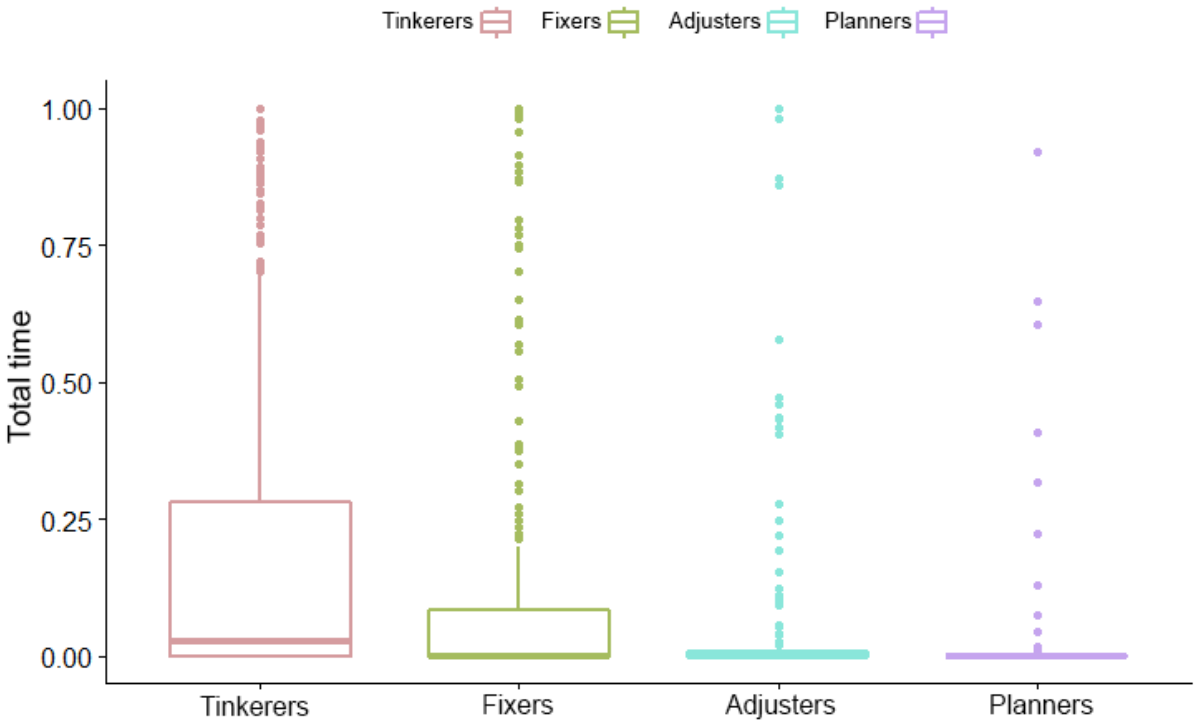


Figure 24 – Boxplot of the students' time on-task

Source: Elaborated by the author

- **Fixers:** Students that tend to make small changes in the total number of lines of code at each submission presented second lowest values of difNLOC feature and took more time to complete their online assignments when compared to those students in clusters C and D
- **Adjusters:** Students that tend to make significant changes in the total number of lines of code at each submission (presents higher values of difNLOC feature) and usually smaller amounts of time to complete their online assignments
- **Planners:** Students that tend to make the most significant changes in the total number of lines of code at each submission (presents higher values of difNLOC feature) and the smallest amounts of time to complete their online assignments, usually completing the assignments in one submission.

To summarize, the average amount of changes in the students' line of code at each submission varies according to the behaviors identified, as well the times that students that expressed the said behavior take to finish the online assignments.

The last research question presented in this work **RQ5** - *Do students present different tinkering behaviors during the online assignments?* - seeks to assess if the students present different tinkering behaviors according to their problem-solving strategies through the online assignments or if they tend to present similar behaviors independent of the assignment. In order

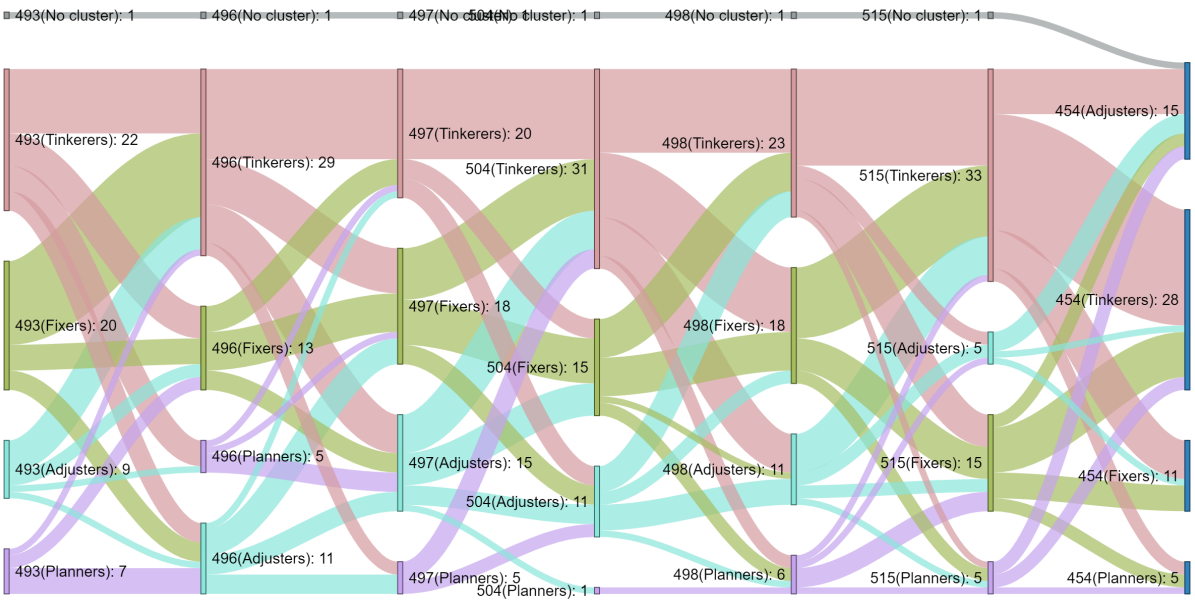


Figure 25 – Sankey diagram of the online assignments of the algorithm and conditional structures concept

Source: Elaborated by the author

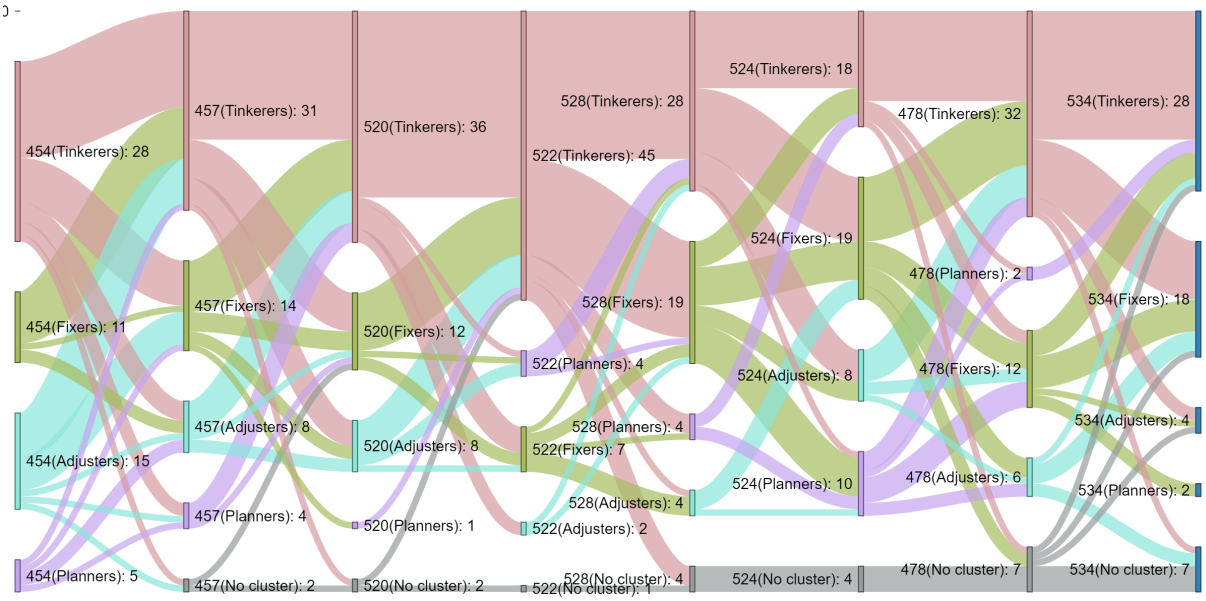


Figure 26 – Sankey diagram of the online assignments of the Looping structures and vectors subject concept

Source: Elaborated by the author

to answer this research question, three Sankey diagrams were created to allow the visualization of the identified students' tinkering behaviors through the online assignments (Figure 25, Figure 26 and Figure 27). The Sankey diagram is a type of flow diagram in which the arrow's width is linearly proportional to the data flow rate.

As it can be observed in Figures 25, 26 and 27, students tend to change between the

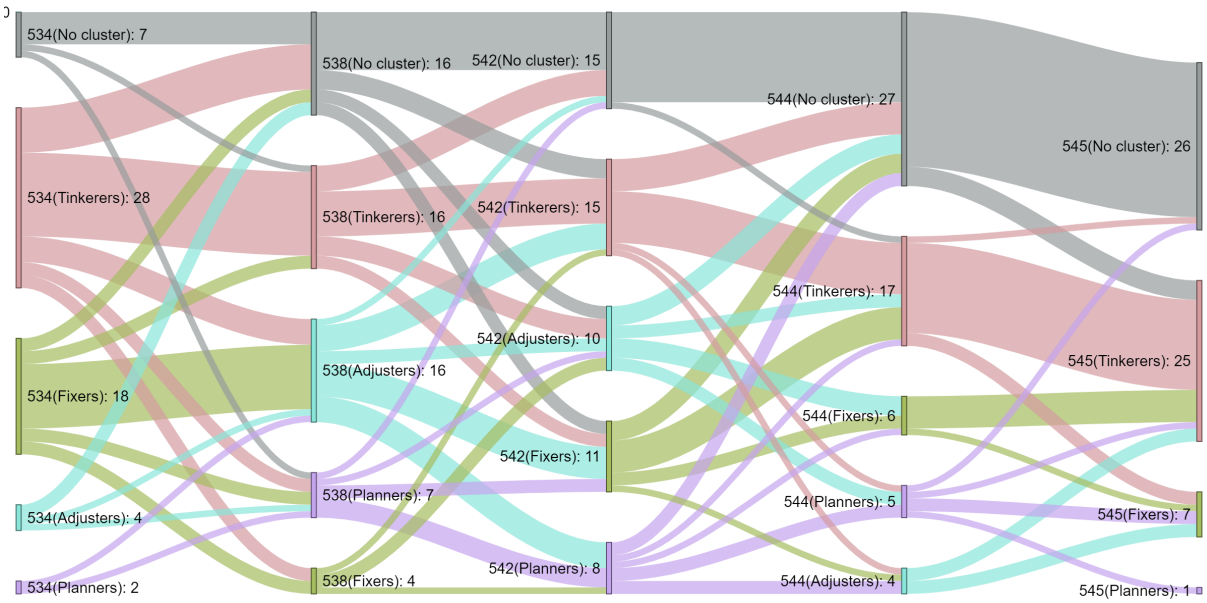


Figure 27 – Sankey diagram of the online assignments of the Functions and recursion concept

Source: Elaborated by the author

different students' clusters that represents tinkering behaviors. In addition, it can be observed that most of the students expressed the tinkerer behavior on almost all online assignments. This change in behaviors corroborates with other works present in literature that observed that students could express more than one behavior category during their problem-solving attempts (VOSSOUGHI; BEVAN, 2014; TURKLE; PAPERT, 1992; SHARMA *et al.*, 2018). The changes in the behavior of each student through all online assignments can be seen in Appendix A.

It's worth pointing that the number of students that didn't make any submission for and assignment (No cluster represented by the gray color) increases significantly towards the end of the semester. This lack of submissions could be explained due to several factors, such as other subjects' exams and assignments, or students' interest in the subject, and assignment's increasingly difficulty.

To answer the **RQ5** - *Do students present different tinkering behaviors during the online assignments?*: Yes. As can be observed in the Figures 25, 26 and 27, the students presented different tinkering behaviors during the online assignments.

4.3 Threats to validity

To assist in the replicability and to highlight the caveats of this work, this section presents this work's main threats to validity according to the WOHLIN *et al.* (2012).

Conclusion validity threats are concerned with factors that may temper with the results obtained. In this work, the *reliability of measures* was identified as a threat: the features used to

identify tinkering behaviors in the students' assignments fit in the TURKLE; PAPERT (1992) description of tinkering. However, it is not possible to discard the possibility that the selected features were not the best representation of the behaviors. To mitigate this threat, it was selected non-subjective features that were extracted through an automatic process to avoid human bias. Furthermore, the feature selection process and the dataset partitioning were conducted in the same way to also avoid human bias (Section 4.1).

Random irrelevancies in experimental setting were the second type of conclusion validity threats identified: other random elements that cannot be predicted, as noisy environments and interruptions during the students' online assignment sessions may be responsible by temper with the results obtained. Despite that, since the online assignments were not applied during the classes, and the data used is prior to this work, this threat could not be mitigated.

Internal validity threats refers to factors that may temper with the causality effect of dependent and independent variables. Among the possible internal validity threats, *Maturation* was identified as the main threat. *Maturation* represents different (positive or negative) reactions that a participant may present as time passes. In this work, the students may have felt fatigued, or boredom during the assignments (negative reactions), as well the students may have learned different topics and overcome difficulties during the time analyzed (positive effects). However, the students were free to choose when to do the online assignments (from the moment that it was available until the end of the semester), which should help to mitigate the negatives effects of maturation. On the other hand, the positive effects were expected in this work and considered part of the tinkering behaviors identified.

Construct validity is related to the generalization of the results obtained in the study. *Confounding constructs and levels of constructs* refers to the effect where the presence of a result is confounded with the effect of the result. In this work, the different tinkering behaviors expressed by students, in special planners, may be related to previous knowledge in the subject matter approached. Unfortunately, as the students' data used is prior to this work development, a pretest to assess the students' previous knowledge was not possible to be applied to mitigate this threat. This threat may also be expressed through students implementing the online assignments in other environments and submitting the final, or almost final, results. Even tough paste features were disable in the VPL moodle's module, due to the prior period and natural limitations of the students' data, this threat also could not be mitigated.

In addition, the ex post facto methodology used in this work also belongs in this section. In this case, due to the nature of the methodology, the lack of control by the researchers is expected. However, to mitigate this threat, it was followed the actions proposed by (ARY; JACOBS; RAZAVIEH, 1972; COHEN; MANION; MORRISON, 2013), where the inclusion of an extraneous variable (tinkering) is included as another independent variable (difNLOC), on which the students were classified.

CHAPTER

5

FINAL REMARKS

In this chapter, we address the final remarks, the main contributions, the publications resulting from this work, and the future works.

5.1 Conclusions

Problem-solving is a high order cognitive process that plays an essential role in computer science education, making it essential for educators to consider in addition to programming paradigms, environments, and tools, the teaching, and development of problem-solving strategies. Based on this, researchers have made several attempts to identify and classify students according to their employed problem-solving strategies. Amongst those attempts, the classification of students between tinkerers and planners proposed by TURKLE; PAPERT (1992) is the most well-accepted. This classification introduces the existence of a more playful, and less structured, bottom-up problem-solving strategy (tinkerers), that contradicts the usual well structured, top-down problem-solving strategy present in software engineering (planners).

In this context, this work's objective is to investigate the students' tinkerer and planner behaviors in online environments and their impact on the students' grades performance. To achieve this objective, this work presented an unsupervised data mining approach to identify students' problem-solving behaviors based on the amount of tinkering employed by the students. Clustered by their behaviors, the students' performance was compared by five different grades (pen and paper assignments, midterm and final exam, online assignments, exam's average, and subject's final grade), as well the amount of time that the students took to finish the online assignments. Based on data analysis, it was possible to answer the research questions of this work, Table 17.

The results obtained indicate four different students' behaviors based on the amount of tinkering employed by the students. While two of these behaviors can be closely related to the

tinkerers and planners' behaviors defined by TURKLE; PAPERT (1992), the results also indicate the existence of the other two behaviors with less intense use of the tinkering/planning strategies (Fixers and Adjusters).

From the data analysis, it was found a significant difference in performance between students that expressed the Tinkerer behavior and students in other clusters during the final exams. This result corroborates with the findings in the work of BLIKSTEIN *et al.* (2014), where students that presented a higher number of changes in their codes also presented a better performance. The data analysis also allows the observation of different students' paces to complete the online assignments. Students that expressed Tinkerer and Fixer behaviors spent more time time-on-task and total time to complete their online assignments than those that expressed Adjuster and Planner behaviors. These results also corroborate with the findings in the related studies (LUSTRIA, 2007; GOLDHAMMER *et al.*, 2014; THOMPSON *et al.*, 2017) where time-on-task was positively correlated with students' grades performance in online environments. However, those studies do not consider the students' problem-solving strategies employed and it's correlations with their findings.

In addition, the obtained association rules offered some insights to understand the students' behaviors and how they related to the other features. These patterns could be used to help in the course and assessment planning that involves students' tinkering behaviors—benefiting not only the students but the teachers and institutes concerned to extract the students' potentials.

Moreover, it can be observed in the diagram in subsection 3.1 that students shifted their behaviors during the assignments. This result corroborates with the works conducted by TURKLE; PAPERT (1992), VOSSOUGHI; BEVAN (2014), and SHARMA *et al.* (2018), that also observed changes in students behaviors related to their problem-solving strategies.

These findings may help to better understand the impact of the identified behaviors have on the students' grades performance, indicating that students that employ tinkering as a problem-solving strategy had similar or better performances than the students that employed more planned strategies. These findings also indicates that tinkering can be a valid problem-solving strategy to be employed by students during online strategies as discussed by Turkle and Papert (TURKLE; PAPERT, 1992),

5.2 Contributions

In order to attain our objectives, this work proposes an unsupervised data mining approach to identify and classify students in an introductory programming subject according to the amount of tinkering they express during online assignments. Based on an optimum hierarchical partitioning algorithm, it was identified four different tinkering behavior expressed by the students related to their employed problem-solving strategies through eighteen online assignments. An assessment of these tinkering behaviors based on the students' grades in different assignments

Table 17 – Summary of research questions' answers

Research Question	Answer
RQ1: Which students' tinkering behaviors can be identified using the selected assignment features ?	Four students' behaviors were identified using the difNLOC feature
RQ1.1: How can the differences and similarities in student's tinkering behaviors be characterized based on the selected features?	The average amount of changes in the students' line of code at each submission varies according to their behaviors, as well the times to finish the online assignments
RQ2: Is there a grades' performance difference between the identified students' tinkering behaviors?	Yes. Tinkerers students, which expressed the highest amounts of tinkering, presented a better performance in the final exam than those students that expressed lower amounts of tinkering
RQ2.1: Is there a significant difference in performance among the tinkering behaviors identified depending on the assignment type ?	Yes. Tinkerers students, which expressed the highest amounts of tinkering, presented a better performance in the final exam (pen and paper) than the students in other clusters.
RQ3: Is the total time to complete an online assignment different among the identified tinkering behaviors?	Yes. The obtained results show that students' that expressed Tinkerer and Fixer behaviors spent more time to complete online assignments
RQ4: Is the students' time on task for online assignments different among the identified tinkering behaviors?	Yes. The statistical results show that Tinkerer and Fixer students takes more time between their first and last sessions to complete their online assignments
RQ5: Do students present different tinkering behaviors during the online assignments?	Yes. As can be observed in the Sankey's diagrams in subsection \ref{sec:post-hoc}, the students' presented different tinkering behaviors during the online assignments

Source: Elaborated by the author

(online assignments, in-class assignments, and tests) through the semesters allowed a performance evaluation of these behaviors in different settings.

In summary, the main contributions of this research are:

- Identification of students' problem-solving strategies based on the tinkering behaviors expressed by the students during the online assignments;
- This work presents assertive about the impacts of students' tinkering problem-solving strategies on their grades performance, as well as evidence of these strategies validity on learning environments. In addition, all the tools developed and data collected were made available in <https://github.com/fcarvalhos/masters>;

It's expected that the contributions presented in this work and the data collected and analyzed, may be employed by educators and researchers to help develop courses and assessment planning that involves students' problem-solving strategies related to tinkering. This benefits not only the students but also the teachers and institutes concerned to extract the students' potentials.

5.3 Publications

During this work's development, several scientific papers were produced and published—one of them directly related to this work.

1. SILVA, F. H. C.; TODA, A. M.; ISOTANI, S. Tinkering as a Problem-Solving strategy employed by students in online environments. **ACM Transactions on Computing Education (TOCE)**, to be published.

Furthermore, other works related to computer science in education were developed and published in collaboration with the research group and institute colleagues.

2. SILVA, F. H. C.; TODA, A. M.; ISOTANI, S. Towards a link between Instructional Approaches and Gamification - A Case Study in a Programming Course. In: WORKSHOP DE INFORMATICA NA ESCOLA, 24., 2018, Fortaleza. **Proceedings...** Porto Alegre: SBC, 2018. p. 157 - 165. ISSN 2316-6541. Available at: <https://br-ie.org/pub/index.php/wie/article/view/7884>. Accessed in: 21 Nov. 2019. DOI:<http://dx.doi.org/10.5753/cbie.wie.2018.157>.

3. PEREIRA, L. T.; SILVA, F; PALOMINO, P. T.; TOLEDO, C. F. M.; ISOTANI, S. A abordagem construtivista no desenvolvimento de um serious game do gênero escape room. In: SIMPOSIO BRASILEIRO DE JOGOS E ENTRETENIMENTO DIGITAL, 17., 2018, Foz do Iguaçu. **Proceedings...** Porto Alegre: SBC, 2018. p.1011 - 1018. ISSN 2179-2259. Available at: <http://www.sbgames.org/sbgames2018/files/papers/EducacaoFull/186874.pdf>. Accessed: 21 nov. 2019.

4. LYRA, K. T.; ALVES, M. L.; SILVA, F. H. C.; SOUZA, K.; ISOTANI, S. An agile project management experience: points of view of graduate students. In: BRAZILIAN SYMPOSIUM OF SOFTWARE ENGINEERING, 32., 2018, São Carlos. **Proceedings...** New York: ACM, 2018. p. 240–249. ISBN 9781450365031. Available at: <https://dl.acm.org/doi/abs/10.1145/3266237.3266248>. Accessed: 21 Nov. 2019.

5. FORTES, R. P. d. M.; SALGADO, A. d. L.; CORREA, C. A. S.; SILVA, F. H. C.; ARAVECHIA,H. d. A. T.; GIBERTONI, L. H. S. Desafios e avanços de conhecimentos sobre acessibilidade em sistemas computacionais interativos. In: SILVA, S., DIGIAMPIETRI, L. (org). **(Re) Conhecendo a USP:** contribuições do ensino,da pesquisa e da extensão no campo das deficiências. São Paulo: FEUSP, 2017. p.281 - 294.

6. SANTOS, W. O. dos; SILVA, F. C.; HINTERHOLZ, L. T.; ISOTANI, S.; BITTENCOURT, I. I.Computação desplugada: Um mapeamento sistemático da literatura nacional. **RENOTE**,v. 16,n. 2, p. 626 - 635, 2018. Available at: <https://seer.ufrgs.br/renote/article/view/89241/51486>. Accessed: 21 nov. 2019.

5.4 Limitation and Future Work

In this work, only students in an introductory programming subject were evaluated using their online assignments and grades obtained in the subject. Additionally, this work was limited by the demographic factors of being conducted during one semester with one class of students. For a broader and more robust definition of the identified students' tinkering problem-solving strategies, the assessment of more advanced subjects and a higher number of students with different cultural and social backgrounds is necessary.

The data analysis of the students' performance, as well as their changes in their problem-solving strategies between the assignments, may lack information since the data used in this work is prior to its development. In this sense, further research that involves surveys with the students, as well as retention tests, could be beneficial to the findings in this work. Other threats to this work's validity are discussed in subsection 4.3.

This way, it's suggested as future works, investigation of the students' tinkerer and planner problem-solving strategies in different computer science subjects, and the application of other tools and methods (e.g., think-aloud or surveys) to better analyze their employed problem-solving strategies through their perspective. Lastly, the replication of this work in other subjects is also desired as a means to extend the generalization power of the assertions made in this work.

BIBLIOGRAPHY

AGGARWAL, C. C. **Recommender systems: The Textbook**. Cham: Springer, 2016. 498 p. ISSN 0001-0782. ISBN 9783319296573. Citation on page 27.

AGRAWAL, R.; IMIELIŃSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. In: ACM. **Acm sigmod record**. [S.l.], 1993. v. 22, n. 2, p. 207–216. Citation on page 61.

ANTONENKO, P. D.; TOY, S.; NIEDERHAUSER, D. S. Using cluster analysis for data mining in educational technology research. **Educational Technology Research and Development**, Springer, v. 60, n. 3, p. 383–398, 2012. Citations on pages 29, 31, and 32.

ARY, D.; JACOBS, L. C.; RAZAVIEH, A. **Examination Questions for Introduction to Research in Education**. [S.l.]: Holt, Rinehart and Winston, 1972. Citation on page 72.

BARNES, D. J.; FINCHER, S.; THOMPSON, S. Introductory problem solving in computer science. In: **5th Annual Conference on the Teaching of Computing**. [S.l.: s.n.], 1997. p. 36–39. Citation on page 22.

BLAND, J. M.; ALTMAN, D. G. Multiple significance tests: the bonferroni method. **Bmj**, British Medical Journal Publishing Group, v. 310, n. 6973, p. 170, 1995. Citation on page 64.

BLIKSTEIN, P.; WORSLEY, M.; PIECH, C.; SAHAMI, M.; COOPER, S.; KOLLER, D. Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. **Journal of the Learning Sciences**, Taylor & Francis, v. 23, n. 4, p. 561–599, 2014. Citations on pages 18, 23, 24, 25, 31, 32, 36, 43, and 74.

BRANDL, K. Review of are you ready to" moodle"? **Language learning & technology**, University of Hawaii National Foreign Language Resource Center, v. 9, n. 2, p. 16–23, 2005. Citation on page 38.

BRUCE, C.; BUCKINGHAM, L.; HYND, J.; MCMAHON, C.; ROGGENKAMP, M.; STOODLEY, I. Ways of experiencing the act of learning to program: A phenomenographic study of introductory programming students at university. **Journal of Information Technology Education: Research**, Informing Science Institute, v. 3, n. 1, p. 145–160, 2004. Citations on pages 18, 23, 32, and 36.

BUDIMAN, E.; KRIDALAKSANA, A. H.; WATI, M. *et al.* Performance of decision tree c4. 5 algorithm in student academic evaluation. In: SPRINGER. **International Conference on Computational Science and Technology**. [S.l.], 2017. p. 380–389. Citation on page 25.

CALIŃSKI, T.; HARABASZ, J. A dendrite method for cluster analysis. **Communications in Statistics-theory and Methods**, Taylor & Francis, v. 3, n. 1, p. 1–27, 1974. Citation on page 51.

CHI, M.; GLASER, R. **Problem-solving ability. Dalam RJ Sternberg (Ed.), Human abilities: An information-processing approach (227–250)**. [S.l.]: New York: Freeman, 1985. Citation on page 22.

COHEN, L.; MANION, L.; MORRISON, K. **Research methods in education**. [S.l.]: routledge, 2013. Citation on page 72.

DAVIES, D. L.; BOULDIN, D. W. A cluster separation measure. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, n. 2, p. 224–227, 1979. Citation on page 51.

DUNN, O. J. Multiple comparisons among means. **Journal of the American statistical association**, Taylor & Francis Group, v. 56, n. 293, p. 52–64, 1961. Citation on page 64.

DUTT, A.; AGHABOZRGI, S.; ISMAIL, M. A. B.; MAHROEIAN, H. Clustering algorithms applied in educational data mining. **International Journal of Information and Electronics Engineering**, IACSIT Press, v. 5, n. 2, p. 112, 2015. Citation on page 26.

DUTT, A.; ISMAIL, M. A.; HERAWAN, T. A systematic review on educational data mining. **IEEE Access**, IEEE, v. 5, p. 15991–16005, 2017. Citations on pages 25, 26, 27, 32, and 49.

DY, J. G.; BRODLEY, C. E. Feature subset selection and order identification for unsupervised learning. In: CITESEER. **ICML**. [S.l.], 2000. p. 247–254. Citation on page 52.

EDUCATIONAL Data Mining Society. Available: <http://educationaldatamining.org/>. Citation on page 26.

FAYYAD, U.; IRANI, K. Multi-interval discretization of continuous-valued attributes for classification learning. 1993. Citation on page 61.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. The kdd process for extracting useful knowledge from volumes of data. **Communications of the ACM**, ACM, v. 39, n. 11, p. 27–34, 1996. Citations on pages 25 and 40.

FEDORENKO, E.; IVANOVA, A.; DHAMALA, R.; BERS, M. U. The language of programming: A cognitive perspective. **Trends in cognitive sciences**, Elsevier, 2019. Citation on page 17.

FENG, Q.; ZHU, L.-q.; CHENG, Z.-k.; ZHANG, Q. Research on student performance evaluation based on random forest. **DEStech Transactions on Engineering and Technology Research**, n. eeta, 2017. Citations on pages 26 and 32.

GAN, G.; MA, C.; WU, J. **Data Clustering: Theory, Algorithms, and Applications**. Berlin, Heidelberg: Springer, 2007. 466 p. Citations on pages 27 and 28.

GICK, M. L. Problem-solving strategies. **Educational psychologist**, Taylor & Francis, v. 21, n. 1-2, p. 99–120, 1986. Citation on page 22.

GOLDHAMMER, F.; NAUMANN, J.; STELTER, A.; TÓTH, K.; RÖLKE, H.; KLIEME, E. The time on task effect in reading and problem solving is moderated by task difficulty and skill: Insights from a computer-based large-scale assessment. **Journal of Educational Psychology**, American Psychological Association, v. 106, n. 3, p. 608, 2014. Citation on page 74.

GOLDSTEIN, F. C.; LEVIN, H. S. Disorders of reasoning and problem-solving ability. Guilford Press, 1987. Citation on page 17.

HAMSA, H.; INDIRADEVI, S.; KIZHAKKETHOTTAM, J. J. Student academic performance prediction model using decision tree and fuzzy genetic algorithm. **Procedia Technology**, Elsevier, v. 25, p. 326–332, 2016. Citations on pages 26 and 32.

HAN, J.; KAMBER, M.; PEI, J. Data mining concepts and techniques third edition. **The Morgan Kaufmann Series in Data Management Systems**, p. 83–124, 2011. Citations on pages 24, 27, 28, 29, 30, 49, and 50.

ISAAC, S. **Handbook in Research and Evaluation: A Collection of Principles, Methods and Strategies... Studies in Education and Behavioral Sciences**. [S.l.]: R Knapp, 1971. Citation on page 38.

JUNIOR, F. P. e Elaine Oliveira e David Fernandes e Leandro Carvalho e H. Otimização e automação da predição precoce do desempenho de alunos que utilizam juízes online: uma abordagem com algoritmo genético. **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)**, v. 30, n. 1, p. 1451, 2019. ISSN 2316-6533. Citations on pages 26 and 32.

KANTARDZIC, M. **Data mining: concepts, models, methods, and algorithms**. [S.l.]: John Wiley & Sons, 2011. Citations on pages 24 and 25.

KIESMÜLLER, U. Diagnosing learners' problem-solving strategies using learning environments with algorithmic problems in secondary education. **ACM Transactions on Computing Education (TOCE)**, ACM, v. 9, n. 3, p. 17, 2009. Citations on pages 18 and 22.

KRUSKAL, W. H.; WALLIS, W. A. Use of ranks in one-criterion variance analysis. **Journal of the American statistical Association**, Taylor & Francis Group, v. 47, n. 260, p. 583–621, 1952. Citation on page 62.

KUMAR, M.; SINGH, A. Evaluation of data mining techniques for predicting student's performance. **International Journal of Modern Education and Computer Science**, v. 8, p. 25–31, 01 2017. Citations on pages 28 and 32.

LEEDY, P. D.; ORMROD, J. E. **Practical research: Planning and design**. [S.l.]: Pearson Education, 2014. Citation on page 38.

LÉVI-STRAUSS, C. *et al.* **La pensée sauvage**. [S.l.]: Plon Paris, 1962. Citation on page 24.

LISHINSKI, A.; YADAV, A.; ENBODY, R.; GOOD, J. The influence of problem solving abilities on students' performance on different assessment tasks in cs1. In: ACM. **Proceedings of the 47th ACM technical symposium on computing science education**. [S.l.], 2016. p. 329–334. Citation on page 22.

LORD, H. G. Ex post facto studies as a research method. special report no. 7320. ERIC, 1973. Citation on page 38.

LUSTRIA, M. L. A. Can interactivity make a difference? effects of interactivity on the comprehension of and attitudes toward online health content. **Journal of the American Society for Information Science and Technology**, Wiley Online Library, v. 58, n. 6, p. 766–776, 2007. Citation on page 74.

MACKIE-MASON, J. K.; GROTH, D. P. Why an informatics degree? ACM, 2010. Citations on pages 17 and 21.

MARTINEZ, S. L.; STAGER, G. Invent to learn: Making. **Tinkering, and Engineering in the Classroom**, 2013. Citation on page 24.

MAYER, R. E. **Thinking, problem solving, cognition**. [S.l.]: WH Freeman/Times Books/Henry Holt & Co, 1992. Citation on page 22.

MAYER, R. E.; WITTROCK, M. C. Problem solving. **Handbook of educational psychology**, v. 2, p. 287–303, 2006. Citation on page 22.

MCCABE, T. J. A complexity measure. **IEEE Transactions on software Engineering**, IEEE, n. 4, p. 308–320, 1976. Citation on page 47.

MEC. **RESOLUCAO CNE/CES Nº 5, DE 16 DE NOVEMBRO DE 2016**. 2016. <https://www.semesp.org.br/legislacao/resolucao-cneces-no-5-de-16-de-novembro-de-2016/>. [Online, accessed on October 5, 2019]. Citations on pages 17 and 21.

NEWELL, A.; SIMON, H. A. *et al.* **Human problem solving**. [S.l.]: Prentice-hall Englewood Cliffs, NJ, 1972. Citation on page 21.

ORMROD, J. E. **Human learning**. [S.l.]: Pearson Higher Ed, 2011. Citation on page 21.

PAKHIRA, M. K.; BANDYOPADHYAY, S.; MAULIK, U. Validity index for crisp and fuzzy clusters. **Pattern recognition**, Elsevier, v. 37, n. 3, p. 487–501, 2004. Citation on page 51.

PEREZ, V.; RICHARDSON, K.; ROSENBLUM, J. Literature review: Making and tinkering as an educational tool. 2017. Citations on pages 18 and 24.

PERKINS, D. N.; HANCOCK, C.; HOBBS, R.; MARTIN, F.; SIMMONS, R. Conditions of learning in novice programmers. **Journal of Educational Computing Research**, SAGE Publications Sage CA: Los Angeles, CA, v. 2, n. 1, p. 37–55, 1986. Citations on pages 18, 22, and 32.

PETRICH, M.; WILKINSON, K.; BEVAN, B. It looks like fun, but are they learning? In: **Design, make, play**. [S.l.]: Routledge, 2013. p. 68–88. Citation on page 24.

PINO, J. Rodriguez-del. **VPL—Virtual Programming Lab for Moodle**. 2011. Citation on page 38.

POLYA, G. **How to solve it: A new aspect of mathematical method**. [S.l.]: Princeton university press, 2004. Citation on page 21.

REISBERG, D.; MAYER, R. E. **Problem Solving**. Oxford University Press, 2013. Available: <https://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780195376746.001.0001/oxfordhb-9780195376746-e-48>. Citation on page 21.

RESNICK, M.; ROSENBAUM, E. Designing for tinkerability. **Design, make, play: Growing the next generation of STEM innovators**, p. 163–181, 2013. Citation on page 24.

ROMEIKE, R. What's my challenge? the forgotten part of problem solving in computer science education. In: SPRINGER. **International Conference on Informatics in Secondary Schools-Evolution and Perspectives**. [S.l.], 2008. p. 122–133. Citations on pages 17 and 21.

ROSE, S. Bricolage programming and problem solving ability in young children: An exploratory study. 2016. Citation on page 24.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. **Journal of computational and applied mathematics**, Elsevier, v. 20, p. 53–65, 1987. Citation on page 51.

SANTOS, R. P. dos; COSTA, H. A. X. Análise de metodologias e ambientes de ensino para algoritmos, estruturas de dados e programação aos iniciantes em computação e informática. **INFOCOMP**, v. 5, n. 1, p. 41–50, 2006. Citation on page 17.

SBC. **Curriculos de referencia/1177 diretrizes para ensino de computacao na educacao basica**. 2017. <https://www.sbc.org.br/documentos-da-sbc/send/131-curriculos-de-referencia/1177-diretrizes-para-ensino-de-computacao-na-educacao-basica>. [Online, accessed on August 11, 2019]. Citations on pages 17 and 21.

SHARMA, K.; MANGAROSKA, K.; TRÆTTEBERG, H.; LEE-CULTURA, S.; GIANNAKOS, M. Evidence for programming strategies in university coding exercises. In: SPRINGER. **European Conference on Technology Enhanced Learning**. [S.l.], 2018. p. 326–339. Citations on pages 17, 18, 22, 23, 32, 35, 36, 71, and 74.

SILVA, D. F.; BATISTA, G. E. B.; KEOGH, E. K. Large-scale similarity-based time series mining. In: SBC. **31º Concurso de Teses e Dissertações (CTD_2018)**. [S.l.], 2018. v. 31, n. 1/2018. Citation on page 27.

SIMON, M. K.; GOES, J. Ex post facto research. **Retrieved from**, 2013. Citation on page 38.

SPEARMAN, C. (1904)." general intelligence", objectively determined and measured. **Am. J. Psych. l5: 20l-293**, 1904. Citation on page 48.

THOMPSON, M.; KOLBO, J.; GILKEY, S.; ZHANG, L.; PRITCHARD, M. The effects of move to learn on student time on task and time on task transitions. **National Teacher Education Journal**, v. 10, n. 1, 2017. Citation on page 74.

TURKLE, S.; PAPERT, S. Epistemological pluralism and the revaluation of the concrete. **Journal of Mathematical Behavior**, v. 11, n. 1, p. 3–33, 1992. Citations on pages 18, 23, 24, 35, 36, 43, 54, 57, 71, 72, 73, and 74.

USPDIGITAL. **Disciplina: SSC0600 - Introdução à Ciência de Computação I**. 1999. <https://uspdigital.usp.br/jupiterweb/obterDisciplina?sgldis=SSC0600&codcur=97001&codhab=0>. [Online, accessed on October 3, 2019]. Citation on page 35.

VAZIRGIANNIS, M. Clustering validity. In: ____. **Encyclopedia of Database Systems**. Boston, MA: Springer US, 2009. p. 388–393. ISBN 978-0-387-39940-9. Available: <https://doi.org/10.1007/978-0-387-39940-9_616>. Citation on page 51.

VENDRAMIN, L.; CAMPELLO, R. J.; HRUSCHKA, E. R. Relative clustering validity criteria: A comparative overview. **Statistical analysis and data mining: the ASA data science journal**, Wiley Online Library, v. 3, n. 4, p. 209–235, 2010. Citations on pages 49 and 51.

VITAL, T. P.; LAKSHMI, B.; REKHA, H. S.; DHANALAKSHMI, M. Student performance analysis with using statistical and cluster studies. In: **Soft Computing in Data Analytics**. [S.l.]: Springer, 2019. p. 743–757. Citation on page 31.

VOSSOUGHI, S.; BEVAN, B. Making and tinkering: A review of the literature. **National Research Council Committee on Out of School Time STEM**, National Research Council Washington, DC, p. 1–55, 2014. Citations on pages 24, 35, 48, 71, and 74.

WANG, Y.; CHIEW, V. On the cognitive process of human problem solving. **Cognitive systems research**, Elsevier, v. 11, n. 1, p. 81–92, 2010. Citations on pages 17 and 22.

WARD, J. H. **Hierarchical grouping to optimize an objective function**. 1963. 236–244 p. Citation on page 28.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. **Experimentation in software engineering**. [S.l.]: Springer Science & Business Media, 2012. Citation on page 71.

ZAHRA, S.; GHAZANFAR, M. A.; KHALID, A.; AZAM, M. A.; NAEEM, U.; PRUGEL-BENNETT, A. Novel centroid selection approaches for kmeans-clustering based recommender systems. **Information Sciences**, v. 320, n. Supplement C, p. 156 – 189, 2015. ISSN 0020-0255. Available: <http://www.sciencedirect.com/science/article/pii/S0020025515002352>. Citation on page 28.

APPENDIX

A

STUDENTS' TINKERING BEHAVIOR CLUSTER BY ONLINE ASSIGNMENT

Table 18 – Students' clusters assignment by each online assignment. Tinkerers are marked as T, Fixers as F, Adjusters as A and Planners as P. Students that didn't make a submission for that assignment are marked as N/C

	493	496	497	504	498	515	454	457	478	520	522	524	528	534	538	542	544	545
10169	T	A	F	T	A	T	T	T	F	T	F	T	P	F	A	P	T	T
10170	T	T	F	T	T	T	F	T	F	T	T	F	F	P	A	P	F	F
10171	P	F	A	A	A	F	F	T	A	A	T	P	F	T	T	A	F	T
10172	P	T	F	F	T	P	A	T	T	A	T	A	F	T	A	F	T	T
10174	T	F	T	F	T	T	T	F	T	T	T	F	T	T	N/C	T	T	N/C
10175	A	T	A	F	F	T	F	F	T	T	T	T	T	F	F	A	T	F
10176	T	T	A	T	F	T	T	F	T	T	T	T	T	F	A	F	T	T
10178	P	A	P	T	A	T	A	F	F	T	F	P	T	T	N/C	F	F	T
10179	F	T	F	F	F	F	T	F	T	A	T	F	F	T	N/C	T	N/C	N/C
10181	T	P	A	F	T	T	P	A	T	T	P	P	F	F	P	F	T	T
10183	T	T	T	T	T	T	T	N/C	A	F	T	F	T	F	N/C	N/C	N/C	N/C
10184	A	F	T	F	F	P	T	F	T	T	T	T	F	T	T	F	T	T
10185	T	P	A	F	T	T	P	A	T	A	P	A	T	F	N/C	A	N/C	N/C
10186	F	T	F	T	F	T	A	T	F	F	T	P	F	F	P	P	A	F
10187	F	T	F	A	F	T	F	T	T	T	F	F	F	T	N/C	N/C	N/C	N/C
10188	F	A	P	A	A	A	T	T	T	F	T	A	T	T	T	T	T	T
10189	F	T	T	A	T	T	A	F	F	T	T	P	F	F	A	F	N/C	N/C
10190	A	P	A	F	F	P	A	P	T	T	A	T	F	A	N/C	A	P	F
10191	T	A	A	T	F	T	A	A	T	F	T	T	T	T	T	T	P	F
10192	A	T	T	T	T	F	T	F	T	F	P	T	T	T	A	T	N/C	N/C
10193	T	F	A	A	A	T	T	F	T	F	T	P	F	T	T	A	F	T
10196	T	T	T	T	T	T	T	T	T	T	T	T	T	F	A	A	P	T
10197	F	T	T	T	T	T	T	T	F	T	T	F	T	T	N/C	T	N/C	N/C
10198	A	F	T	T	T	T	T	A	T	T	T	T	T	F	A	T	T	F
10199	F	A	F	T	T	F	T	T	N/C	F	T	N/C	N/C	A	P	P	N/C	T
10200	T	T	T	A	T	F	T	T	T	T	T	T	T	T	T	N/C	N/C	N/C
10201	F	T	T	A	T	T	T	T	T	A	T	F	T	T	A	T	T	T
10202	T	T	T	T	A	T	T	T	T	T	T	T	T	T	N/C	N/C	N/C	N/C
10203	T	T	T	T	F	T	T	A	T	T	A	A	T	N/C	T	N/C	N/C	T
10204	T	P	T	T	F	T	T	T	T	T	T	F	T	F	T	F	T	T
10206	F	F	F	F	P	T	A	F	T	P	T	A	T	T	T	T	T	T
10208	F	T	T	T	T	A	A	T	T	T	T	F	T	T	F	T	T	T
10209	F	F	F	F	A	T	F	A	F	T	T	A	T	T	A	T	T	T
10210	P	F	T	T	P	F	T	P	T	T	T	F	T	T	P	F	A	T
10211	P	A	A	F	P	F	P	F	T	A	F	F	A	F	A	N/C	N/C	N/C
10212	F	T	P	A	A	F	F	T	N/C	T	T	F	F	N/C	N/C	N/C	N/C	N/C
10213	T	T	A	T	F	T	T	T	F	T	T	A	F	F	A	F	F	T
10214	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	N/C	N/C
10215	A	T	F	F	F	T	A	F	T	T	T	T	T	F	F	P	A	F
10216	T	T	A	P	P	P	P	P	F	F	F	P	F	P	P	N/C	N/C	N/C
10217	A	A	A	T	A	A	F	T	T	F	T	F	A	T	T	T	A	T
10218	A	T	P	T	F	T	T	P	A	T	T	P	P	N/C	P	P	P	N/C
10219	F	F	F	T	T	T	A	F	P	F	F	P	A	T	P	A	N/C	N/C
10220	F	F	F	F	T	F	A	T	N/C	A	T	N/C	N/C	F	T	N/C	N/C	N/C
10221	F	T	T	F	T	T	F	T	A	T	T	A	F	F	A	P	P	P
10222	F	A	F	T	P	F	T	A	A	A	P	F	T	N/C	N/C	N/C	N/C	N/C
10223	T	F	A	A	P	A	A	T	N/C	T	T	T	P	T	T	N/C	N/C	N/C
10226	P	A	P	T	F	F	T	T	P	N/C	T	T	F	T	T	A	N/C	T
10227	F	T	A	T	T	T	F	T	T	T	T	F	T	T	N/C	N/C	T	T
10228	F	T	A	T	T	A	A	T	T	F	T	P	P	A	A	F	N/C	N/C
10230	T	A	T	T	A	T	T	T	T	T	T	T	F	T	T	T	N/C	N/C
10231	T	F	F	F	F	F	A	T	T	T	T	T	T	T	T	T	T	T
10232	F	T	T	T	T	T	T	T	F	T	T	T	T	T	F	A	F	T
10233	P	A	F	A	T	F	F	T	N/C	F	T	N/C	N/C	N/C	N/C	N/C	N/C	N/C
10234	T	F	F	A	F	F	F	A	A	T	T	F	F	F	A	A	T	T
10237	T	P	F	T	F	F	P	T	F	T	T	F	T	A	N/C	F	N/C	N/C
10238	A	T	T	T	A	P	T	T	N/C	T	F	F	A	N/C	N/C	N/C	N/C	N/C
10240	F	T	F	T	F	T	T	F	F	T	T	F	F	F	A	P	N/C	N/C


ICMC
USP
SÃO CARLOS