Pontifícia Universidade Católica
do Rio de Janeiro

Raphael Oliveira Cabral

Investigating the Impact of SOLID Design Principles on Machine Learning Code Understanding

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Marcos Kalinowski

Rio de Janeiro
December 2023

Pontifícia Universidade Católica
do Rio de Janeiro

Raphael Oliveira Cabral

Investigating the Impact of SOLID Design Principles on Machine Learning Code Understanding

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee:

Prof. Marcos Kalinowski
Advisor
Departamento de Informática – PUC-Rio

Prof. Hélio Côrtes Vieira Lopes
PUC-Rio

Profª. Maria Teresa Baldassarre
University of Bari

Rio de Janeiro, December 6th, 2023

Raphael Oliveira Cabral

Graduated in Information Systems from Estácio de Sá University. Software architect at SERPRO.

Bibliographic data

Oliveira Cabral, Raphael

Investigating the Impact of SOLID Design Principles on Machine Learning Code Understanding / Raphael Oliveira Cabral; advisor: Marcos Kalinowski. – 2023.

118 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2023.

Inclui bibliografia

1. Informática – Teses. 2. SOLID. 3. Compreensão de código. 4. Aprendizado de Máquina. I. Kalinowski, Marcos. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To my parents and family for their support and encouragement. And especially to my father, who passed away during this journey.

Acknowledgments

To my advisor Professor Marcos Kalinowski for the encouragement and partnership in completing this work.

To the professors who participated in the examination committee, Professor Hélio Lopes and Professor Maria Teresa Baldassarre.

To CNPq and PUC-Rio, for the grants provided, without which this work could not have been accomplished.

To my parents for the education, attention, and care at all times. Especially to my father, who passed away during this master's journey, but wherever he is, he surely cheers for me.

To my fiancée Sandra Martins for all the support, patience, and understanding.

To professors Alessandro Garcia, Sérgio Lifschitz, Clarisse Sieckenius, Marcus Poggi, and Simone Barbosa for their words of support.

To my colleagues at PUC-Rio.

To all the professors and staff of the Department for their teachings and assistance.

To all friends and family who, in one way or another, encouraged or helped me.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Oliveira Cabral, Raphael; Kalinowski, Marcos (Advisor). **Investigating the Impact of SOLID Design Principles on Machine Learning Code Understanding**. Rio de Janeiro, 2023. 118p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Applying design principles has long been acknowledged as beneficial for understanding and maintainability in traditional software projects. These benefits may similarly hold for machine learning (ML) projects, which involve iterative experimentation with data, models, and algorithms. However, ML components are often developed by data scientists with diverse educational backgrounds, potentially resulting in code that doesn't adhere to software development best practices. In order to better understand this phenomenon, we investigated the impact of the SOLID design principles on ML code understanding. To this end, we conducted a controlled experiment with three independent trials (exact replications), overall involving 100 data scientists. We restructured ML code from a real industrial setting that did not use SOLID principles. Within each trial, one group was presented with the original ML code, while the other one was presented with ML code incorporating SOLID principles. Participants of both groups were asked to analyze the code and fill out a questionnaire that included both open-ended and closed-ended questions on their understanding. The study results provide statistically significant evidence that the adoption of the SOLID design principles can improve code understanding within the realm of ML projects. We put forward that software engineering design principles should be spread within the data science community and considered for enhancing the maintainability of ML code.

Keywords

SOLID; Code Understanding; Machine Learning.

Resumo

Oliveira Cabral, Raphael; Kalinowski, Marcos. **Investigando o Impacto da Aplicação de Princípios de Projeto SOLID na Compreensão de Código de Machine Learning**. Rio de Janeiro, 2023. 118p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A aplicação de princípios de design tem sido reconhecida há muito tempo como benéfica para a compreensão e manutenção em projetos de software tradicionais. Esses benefícios podem ser válidos de forma semelhante para projetos de aprendizado de máquina (ML), que envolvem experimentação iterativa com dados, modelos e algoritmos. No entanto, os componentes de ML são frequentemente desenvolvidos por cientistas de dados com diversas formações educacionais, resultando potencialmente em código que não segue as práticas recomendadas de desenvolvimento de software. Para compreender melhor esse fenômeno, investigamos o impacto dos princípios de design SOLID na compreensão do código de ML. Para tanto, conduzimos um experimento controlado com três *trials* independentes (replicações exatas), envolvendo no total 100 cientistas de dados. Reestruturamos o código de ML real da indústria que não usava princípios SOLID. Dentro de cada ensaio, um grupo foi apresentado ao código de ML original, enquanto o outro foi apresentado ao código de ML incorporando princípios SOLID. Os participantes de ambos os grupos foram convidados a analisar o código e preencher um questionário que incluía perguntas abertas e fechadas sobre a sua compreensão. Os resultados do estudo fornecem evidências estatisticamente significativas de que a adoção dos princípios de design SOLID pode melhorar a compreensão do código no âmbito dos projetos de ML. Propomos que os princípios de design de engenharia de software devem ser difundidos na comunidade de ciência de dados e considerados para melhorar a capacidade de manutenção do código de ML.

Palavras-chave

SOLID; Compreensão de código; Aprendizado de Máquina.

Table of contents

List of figures

List of tables

List of codes

List of Abreviations

ML - Machine Learning.

SOLID - Is a mnemonic acronym for the principles introduced by Robert C. Martin (also known as Uncle Bob): Single Responsibility Principle; Open Closed Principle; Liskov Substitution Principle; Interface Segregation Principle; and Dependence Inversion Principle.

SRP - Single Responsibility Principle.

OCP - Open-Closed Principle.

LSP - Liskov Substitution Principle.

ISP - Interface Segregation Principle.

DIP - Dependence Inversion Principle.

From a trace, architecture is born. And when it's beautiful and creates surprise, it can, if handled well, reach the top level of a work of art.

Oscar Niemeyer.

1
Introduction

1.1
Context and Motivation

Contemporary advances in machine learning (ML) and the availability of vast amounts of data have both given rise to the feasibility and practical relevance of incorporating ML components into software-intensive systems. ML is inherently driven by experimentation, requiring data scientists to explore data, algorithms, and models to find the most satisfying way of achieving their objectives (AHO et al., 2020). Moreover, ML is also often used in the context of proofs of concept, which allow for iterative refinement and validation before implementation into a production environment. This may encourage quick deliveries over clean code.

Furthermore, data scientists who are in charge of developing these ML components may have a variety of educational backgrounds, such as economics, mathematics, and physics, and typically lack Software Engineering (SE) foundations (KIM et al., 2017). Therefore, ML code often falls short of adhering to software development best practices, resulting in low-quality code that may pose challenges in terms of maintenance and long-term sustainability (OORT et al., 2021).

Despite code understanding being studied for over 40 years (WYRICH; BOGNER; WAGNER, 2022) and the existence of evidence indicating that maintenance requires a considerable amount of work related to understanding code (ZELKOWITZ; SHAW; GANNON, 1979; FJELDSTAD, 1983; CORBI, 1989; KO et al., 2006; MINELLI; MOCCI; LANZA, 2015), we are not aware of studies extending code understanding investigations to the domain of ML code. This gap is particularly noteworthy because there seem to be several ways in which ML code could benefit from well-established SE practices. For instance, as data scientists often work on common ML components such as data pre-processing, model training, evaluation, and deployment, they could break down that code into reusable modules or functions, enhancing code reuse and saving time and effort.

1.2 Goal

In response to the dynamic environment of ML development and recognizing this identified research gap, in this dissertation, we investigate the impact of using SOLID design principles - which are well-known object-oriented design principles for writing clean code (MARTIN, 2009) - on ML code understanding capabilities of data scientists.

1.3 Method

We conducted a controlled experiment with 100 data scientists from three different organizations that were divided into two groups. The control group was presented with ML code from a real industrial setting that did not incorporate SOLID principles, while the experimental group was presented with that same ML code restructured by applying the SOLID principles. Subsequently, the data scientists were tasked with analyzing the code and filling out a questionnaire that included both closed-ended and open-ended questions related to their understanding of the code and their agreement with statements related to typical implications of applying the SOLID principles.

1.4 Summary of the Results

The results indicate that the adoption of each of the five SOLID design principles significantly improves ML code understanding. More specifically, the application of the principles was also perceived to lead to expected benefits related to having clearly distributed and defined ML code responsibilities, facilitating ML code extensions without substantially changing existing code, favoring low coupling and enabling substituting ML code elements, and resulting in proper segregation of interface operations, not forcing ML code to depend upon methods that it does not use.

1.5 Dissertation Methodology Overview

To understand how code comprehension has been investigated from a software engineering perspective, we first conducted an initial literature review (Step 1). More specifically, in order to understand how code comprehension has been investigated in the context of machine learning software, we performed a systematic mapping study (Step 2), which did not retrieve any results. The next step involved studying each of the SOLID principles (MARTIN,

1995) (Step 3) in order to understand how they could be applied to machine learning code to prepare the instrumentation for the experiment. Finally, to evaluate the impact of the application of SOLID design principles on machine learning code understanding, we conducted an experimental study comparing the code understanding of a "traditional" ML code with a "restructured" one using SOLID principles (Step 4). The subjects of the experiment analyzed source code and answered questions related to their level of understanding and their degree of agreement with statements related to the SOLID principles. An Overview of the Methodology can be seen in Figure 1.1.

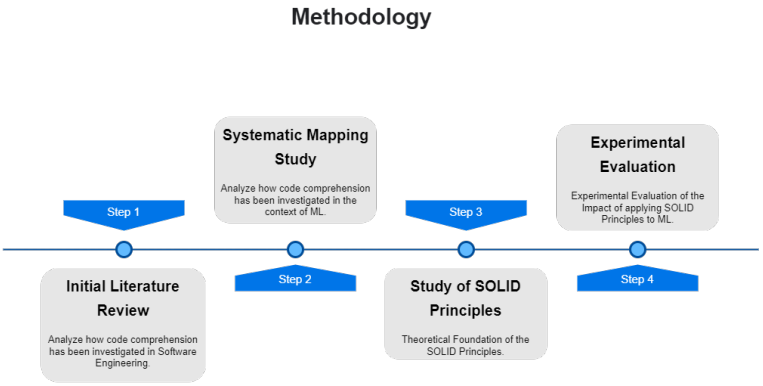

Figure 1.1: Overview of the methodology.

1.6 Text Organization

The remainder of this document is organized as follows. Chapter 2 presents the Theoretical Foundation and the Related Work. Chapter 3 describes the Experimental Study Plan. Chapter 4 presents the experimental study results. Chapter 5 discusses the results and their major implications. Chapter 6 discusses threats to validity. Finally, Chapter 7 presents the conclusion of this dissertation.

2
Theoretical Foundation and Related Work

In this chapter, the theoretical foundation and related work that supports this dissertation are discussed. The section 2.1 introduces the SOLID principles. The section 2.2 explores the relevance of understanding the code. In the section 2.3, the characteristics of professionals dedicated to developing systems based on machine learning are presented. The section 2.4 analyzes the challenges inherent in understanding code in machine learning contexts. Finally, the section 2.5 describes the absence of code comprehension studies focused on machine learning.

2.1
SOLID Principles

Several object-oriented design principles have emerged, such as SOLID (MARTIN, 2000) and GRASP (LARMAN et al., 1998), aiming at making code more understandable, flexible, and maintainable. In this dissertation, we focus on the five SOLID principles, which encapsulate fundamental design concepts. This choice was also motivated by the inclusion of these principles in popular best practices books for software developers (*e.g.*, (MARTIN, 2009)), their consideration in academic investigations regarding design principles in general (*e.g.*, (BRÄUER et al., 2018)), grey literature industrial sources (*e.g.*, (UMANEO, 2023)) and data science blog posts (*e.g.*, (SOULY, 2023; MUSIC, 2023)) anecdotally advocating for benefits of using SOLID within the ML context, and even online courses teaching "SOLID principles for machine learning Engineers" (*e.g.*, (VELARDO, 2023)). Hence, while other design principles might also be of interest, the need to scientifically dig deeper into the use of the SOLID principles within ML is supported by multifaceted scientific and practical motivations.

SOLID is an acronym that refers to a set of 5 object-oriented principles intended to make object-oriented designs more understandable, flexible, and maintainable. The Principles are the Single Responsibility Principle, Open-Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, and Dependency Inversion Principle. They will be presented in more detail in this session.

These principles are a subset of many principles promoted by Robert C. Martin (also known as Uncle Bob), first introduced in 2000 in his paper Design Principles and Design Patterns (MARTIN, 2000). However, the SOLID

acronym was introduced later, around 2004, by Michael Feathers after noting that five object-oriented principles could fit this word.

2.1.1 SRP - Single Responsibility Principle

The first principle that we are going to discuss in this section is the *Single Responsibility Principle* (SRP), and a simple translation of it is cohesion, "have cohesive classes." Everyone has heard of cohesion, and there is still a famous maxim that all your classes always have to be very cohesive. But the question is: how to do it? How to create classes that are cohesive all the time? How to avoid creating classes that have low cohesion?

To achieve this goal of high cohesion, this principle says that the class must have a single responsibility in the software and be specialized in a single subject, therefore having a single reason for change (KALINOWSKI et al., 2023).

For example, let's take a look at the following Animal class in Code 1 (all code examples of this chapter were adapted from (KALINOWSKI et al., 2023)). The Animal class has two jobs, manage the animal's property and store the animal in the database. Later, if you want to save the Animal in different storage, such as a file, you'll need to change the save() method. In this scenario, the responsibility of the class grows with each new type of storage included in the requirement. To make the Animal class conform to the Single Responsibility Principle, you'll need to create another class that is in charge of storing the Animal in a database, as in Code 2.

Violating this principle can result in low cohesion and high coupling. Because classes that have many responsibilities, they can have a lot of code, be more complex, difficult to understand, and difficult to test. They can still have a low reusability, because the other system will rarely need everything it does.

Another pertinent comment is that this principle fits perfectly the norm of modular system development, proposed by Ken Thompson in the 1970s, "Do one thing and do it well" from the Unix Philosophy (Ken Thompson, 2022).

Code 1: Code without SRP.

```
1 class Animal:
2   def __init__(self,name):
3     self.__name = name
4   def get_name(self):
5     return self.__name
6   def save(self):
7     print(f'Save the animal {self.__name} to the database')
```

Code 2: Code with SRP.

```
class Animal:
  def __init__(self,name):
    self.__name = name
  def get_name(self):
    return self.__name

class AnimalDB:
    def save(self, animal):
       print(f'Save the animal {animal.get_name()} to the database')
```

2.1.2 OCP - Open-Closed Principle

The *Open-Closed Principle* (OCP) proposes that classes should be open for extension but closed for modification (KALINOWSKI et al., 2023). Open for extension means that when you receive a new requirement, you can add new behavior. Closed to modification means that in order to introduce a new behavior, an extension, it is not necessary to modify the existing code.

To understand better, let's look at the example present in Code 3, where the Open-Closed Principle is not used. In this example, the Animal class tends to grow with each animal introduced into the system. In the example present in Code 4, where the Open-Closed Principle is used, it is possible to observe the behavior being extended without changing the code of the Animal class. The Animal class is used as an abstraction, and the behavior is implemented in the specialized child classes, Dog and Cat.

Code 3: Code without OCP.

```
class Animal:
  def __init__(self,name):
    self.__name = name
  def get_name(self):
    return self.__name
  def make_sound(self):
    if self.__name == "Dog":
      print("Au Au")
    if self.__name == "Cat":
      print("Miau")
```

Code 4: Code with OCP.

```
class Animal:
  def __init__(self,name):
```

```
3       self.__name = name
4     def get_name(self):
5       return self.__name
6     def make_sound(self):
7      pass
8
9   class Dog(Animal):
10    def make_sound(self):
11      print("Au Au")
12
13  class Cat(Animal):
14    def make_sound(self):
15      print("Miau")
```

2.1.3 LSP - Liskov Substitution Principle

This *Liskov Substitution Principle* (LSP) says that a subclass must be able to be replaced by its superclass (KALINOWSKI et al., 2023). This means that objects can be replaced by their subtypes without affecting the correct execution of the program.

To achieve this goal of being replaceable by its subtype, every child class has to think about the preconditions and postconditions of the parent class. In the precondition, it can never squeeze. And in the postcondition, it can never slack off. If not, the references that point to the parent class, when given a child class, will not work as expected. For example, the child class can only slack off preconditions. Think about the case where we have the parent class, and the parent class has a method that can receive integers from 1 to 10. Then the child class changes that, only allowing it to receive integers from 1 to 5. See that 1 to 5 is more restrictive than 1 to 10. This can affect the functioning of the client classes of the parent class. Likewise, a child class can never squeeze a postcondition. Imagine that we have a method that returns an integer, and this integer is from 1 to 10. Then, the child class overrides the method and starts to return from 1 to 20. This can break the client classes that only expected a return of 1 to 10.

The example present in Code 5 makes it very clear. The rectangle class has no preconditions for the sides. The square class squeezes your preconditions, where its sides must be equal. In this case, it cannot be replaced by the parent class.

Code 5: LSP violation.

```
1 class Rectangle():
2     def __init__(self, l, w):
```

```
3         self.length = l
4         self.width  = w
5
6     def area(self):
7         return self.length*self.width
8
9 class Square(Rectangle):
10    def __init__(self, l, w):
11        if(l != w):
12          raise Exception("The sides of the square must be equal.")
13        super.__init__(l,w)
```

2.1.4 ISP - Interface Segregation Principle

The *Interface Segregation Principle* (ISP) says that specific interfaces are better than a single general purpose interface (KALINOWSKI et al., 2023). That is when, depending on an interface, it shouldn't know methods it doesn't need. Also, when implementing an interface, you shouldn't be forced to implement methods that you don't need.

For a better understanding, we will use the AllInOnePrinter class present in Code 6. It has the methods print, scan, and send_fax. If we use the AllInOnePrinter class as a basis for creating the MultifunctionalPrinter class, we won't have a problem because the MultifunctionalPrinter also prints, scans, and send fax. But if we use the AllInOnePrinter as a basis for creating the DefaultPrinter class, we will violate the Interface Segregation Principle because it only needs to print. So, to solve this problem, let's refactor our example in Code 7, creating the Printer, Scanner, and Fax interfaces, segregating them into specialized interfaces.

Code 6: Code without ISP.

```
1 class AllInOnePrinter:
2   def print(self):
3     pass
4   def scan(self):
5     pass
6   def send_fax(self):
7     pass
8
9 class MultifunctionalPrinter(AllInOnePrinter):
10  def print(self):
11    pass
12  def scan(self):
13    pass
14  def send_fax(self):
15    pass
```

Code 7: Code with ISP.

```
class Printer:
    def print(self):
      pass

class Scanner:
   def scan(self):
     pass

class Fax:
   def send_fax(self):
     pass

class MultifunctionalPrinter(Printer,Scanner,Fax):
   def print(self):
     pass
   def scan(self):
     pass
   def send_fax(self):
     pass

class DefaultPrinter(Printer):
    def print(self):
      pass
```

2.1.5 DIP - Dependency Inversion Principle

The *Dependency Inversion Principle* (DIP) has the following premise: "Depend on abstractions, not on concretions" (KALINOWSKI et al., 2023). The idea is that whenever it is necessary to couple to another class or module, the less stable one should depend on the more stable one. A stable class is one that tends to change very little. The advantage is that if it changes very little, it won't propagate the change to our implementation.

Let's see Code 8 as a basis for our example, where we have the Dog class, Cat class, and Animal abstraction. Code 9 presents the Owner class without DIP because it depends directly on an implementation, which is not very stable. Code 10 presents the Owner class with DIP, which directly depends on the Animal abstraction, which is more stable.

Code 8: Base code to explain the Dependency Inversion Principle (DIP).

```
class Animal:
  def make_sound(self):
    pass

class Dog(Animal):
  def make_sound(self):
    self.bark()
  def bark(self):
    print("Au Au")

class Cat(Animal):
  def make_sound(self):
    self.meow()
  def meow(self):
    print("Miau")
```

Code 9: Code without DIP.

```
class Owner:
  def toStroll(self,dog: Dog):
    dog.bark()
```

Code 10: Code with DIP.

```
class Owner:
  def toStroll(self,animal: Animal):
    animal.make_sound()
```

2.2 Importance of Code Comprehension

As evidenced by the systematic mapping conducted by MARVIN WYRICH et al.(WYRICH; BOGNER; WAGNER, 2022), the importance of code comprehension has been recognized for over 40 years by the scientific community. Where during these years, several studies show that developers indeed invest a considerable amount of their daily work in understanding code (CORBI, 1989), (FJELDSTAD, 1983), (KO et al., 2006), (MINELLI; MOCCI; LANZA, 2015), (ZELKOWITZ; SHAW; GANNON, 1979).

Zelkowitz et al. (ZELKOWITZ; SHAW; GANNON, 1979) claim that understanding the program takes more than half the time spent on maintenance of the software. This statement is also confirmed by Fjeldstad and Hamlen (FJELDSTAD, 1983) and Corbi (CORBI, 1989). Demonstrated through empirical studies by Minelli et al. (MINELLI; MOCCI; LANZA, 2015) and by Xia et al. (XIA et al., 2017).

In this sense, there is great motivation among researchers to optimize this process through scientific research. As a consequence, many studies are conducted to support the developer in the task of understanding the source code and discovering which characteristics contribute to this understanding.

This effort by the scientific community can also be seen with the establishment of the International Conference on Program Comprehension (ICPC), which in 2022 celebrates its 30th anniversary. Even making it clear that understanding the source code is only a subset of understanding the program, understanding the program goes beyond understanding the source code. It can involve understanding the software architecture, requirements, diagrams, etc.

2.3 Characteristics of the Professional Who Develop ML-Enabled Systems

Data science has become popular in recent years as companies recognize the value of large volumes of data at their disposal, either to optimize their operations, to use it as a decision support element, or even to create new products that make use of the intelligence that is available by analyzing this data.

Data scientists generally do not have a typical four-year educational background in some computer-oriented degree (MAY, 2009), where it is possible to acquire skills related to Software Engineering. In research, Harris et al. (HARRIS; MURPHY; VAISMAN, 2013) asked more than 250 data science professionals how they viewed their skills, careers, and experiences. They observed evidence that data scientists have the depth of their skills “T-shaped" and have a wide range of skills, with depth in a single area of expertise.

As evidenced by the study carried out by Kim et al.(KIM et al., 2016), many have degrees in other fields, such as statistics, physics, mathematics, bioinformatics, applied mathematics, business, economics, and finance. His interdisciplinary background contributes to his strong skills in numerical reasoning for data analysis, where many have titles of Masters or Doctorate, and many have previous work experience with big data.

According to the statement made by Kim et al. (KIM et al., 2016), Data Scientists with the analytical and software engineering skills to analyze these huge volumes of data have been hard to come by, and only recently have software companies started to develop skills in software-driven data analysis.

2.4 Challenges in ML Code Comprehension

ML projects present unique challenges in terms of code comprehension. One of the primary factors contributing to these challenges is the inherently experimental nature of ML. Data scientists often work on interactive computational notebooks (SHEN, 2014) that combine code, text, and execution results to handle data, train, and evaluate models iteratively. This emphasis on rapid prototyping and literate programming documents can encourage poor coding practices and results that are difficult to reproduce (PIMENTEL et al., 2019; PIMENTEL et al., 2021).

In addition to the experimentation-driven challenges, data scientists have a variety of educational backgrounds, such as statistics, physics, mathematics, and economics (AHO et al., 2020). While this diversity of expertise is valuable for tackling complex ML problems, it also poses challenges for code comprehension. Many of these team members may lack formal SE foundations, leading to code that may not adhere to established best practices. In this line, Kim *et al.* (KIM et al., 2016) state that it is difficult to find data scientists who combine analytical and SE skills. As a consequence, there have been studies reporting the prevalence of code smells in ML code (OORT et al., 2021) and indicating that ML code is commonly subject to technical debt and refactoring (TANG et al., 2021).

2.5 Absence of ML Code Comprehension Studies

Taking into account the importance of code understanding for the developer's everyday work, we performed a systematic mapping of the literature to provide an overview of research contributions of code comprehension in ML-oriented software.

The systematic mapping study was carried out following the secondary study guidelines proposed by Kitchenham and Charters (KITCHENHAM; CHARTERS, 2007) and the specific systematic mapping guidelines by Petersen (PETERSEN; VAKKALANKA; KUZNIARZ, 2015).

2.5.1 Research questions

RQ1. What code understanding contributions have emerged to support the software development of ML-based systems? This question is intended to provide an overview of research contributions on code understanding in ML-oriented software.

RQ2. What metrics are used to measure code comprehension in machine learning software? The focus of this question is to identify which metrics are being used to perform static code analysis and measure understanding, such as cyclomatic complexity, cognitive complexity, lines of source code, lines of executable code, etc.

RQ3. What design patterns (GAMMA et al., 1995), SOLID principles (MARTIN, 2003), or even clean code rules (MARTIN, 2009) are addressed in scientific contributions to understanding code in software aimed at machine learning? The focus of this question is to identify the SOLID principles, design patterns, or even clean code rules present in the scientific contributions of code understanding in software focused on machine learning.

RQ4. What are the types of research contributions? The purpose of this question is to classify articles according to their research type facets. We will adopt the scheme of Wieringa classification (WIERINGA et al., 2006).

RQ5. What types of empirical evaluations were used to evaluate research contributions? Obtaining this information provides an idea of the scientific rigor of the reported evidence.

2.5.2 Research strategy

The mapping employs a hybrid search strategy (MOURÃO et al., 2020), through a search in the Scopus Database based on a search string. Soon after, he uses the snowball technique backward and forward, following Wohlin's guidelines (WOHLIN, 2014) in order to increase the article base.

The Search String to perform the database search from Scopus was developed using the PICO (Population, Intervention, Comparison, Outcome) criteria (LEONARDO, 2018).

Our research focuses on ML-based systems and aims to identify Code Understanding contributions to such systems. For this reason, it was necessary to create keywords for machine learning and Code Comprehension.

The defined search string was applied to titles, abstracts, and keywords: "(software OR applications OR systems) AND ((machine AND learning) OR ml) AND (code AND comprehension)", where we had a return of 118 documents.

2.5.3 Inclusion criteria

1. The main criteria for inclusion of studies will be contributions from code comprehension in the context of machine-oriented software Learning.

2. When more than one contribution references the same study, only the most recent contribution will be considered.

3. When multiple studies are referenced in a contribution, each study is considered separately.

2.5.4 Exclusion criteria

1. Articles that do not meet the inclusion criteria.

2. Articles that are not written in English.

3. Articles that make use of machine learning to understand the source code.

4. Articles published in journals classified below B3 in the Qualis Capes system.

5. Articles with less than 5 pages.

2.5.5 Study selection

The first step consisted of searching for articles using the search string in the Scopus database. The search string was applied to titles, abstracts, and keywords on November 6, 2022, and returned 118 documents, as can be seen through Figure 2.1.

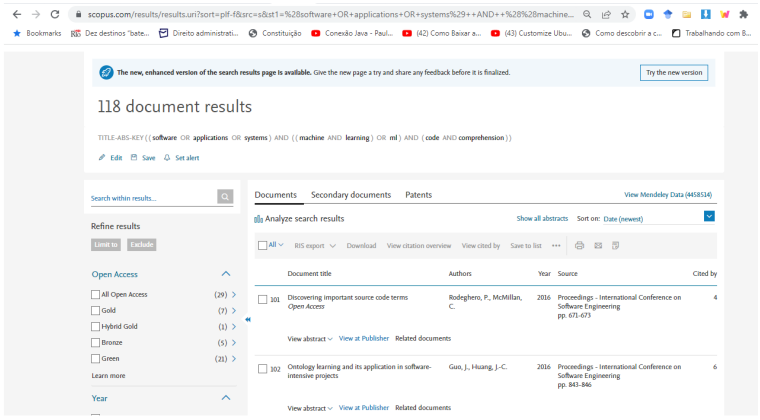

Figure 2.1: Scopus database search return.

In the second stage, the exclusion and inclusion criteria were applied, and no document was selected for the next stage.

Of the 118 articles selected, by reading titles and abstracts, 114 articles did not meet the main inclusion criteria, studies with contributions from code comprehension in the context of machine learning software.

The remaining 4 articles (VISWANATHAN; KUMAR; SOMAN, 2019) (SHALABY et al., 2017) (GONÇALES et al., 2020) (HUANG et al., 2020) were removed by the exclusion criteria; they corresponded to articles that make use of machine learning to understand the source code.

2.5.6 Conclusion of systematic mapping study

As no study passed the inclusion and exclusion criteria of the systematic mapping, it was possible to highlight the absence of studies with code comprehension contributions specifically aimed at machine learning. This serves as a motivation to continue the research.

In this dissertation, we provide an initial step to address this gap by investigating the impact of using the SOLID design principles on ML code understanding.

3
Experimental Study Plan

We decided to conduct a controlled experiment because we were interested in investigating the impact of specific factor levels (using or not using the SOLID design principles) on code understanding, in isolation from other confounding factors (WOHLIN et al., 2012). Hereafter, we detail the experimental study planning steps as suggested by Wohlin *et al.* (WOHLIN et al., 2012) (Figure 3.1), as well as the study operation, data collection, and analysis procedures. While we assessed and mitigated threats to validity during the experimental study planning, we will discuss them in Section 6.

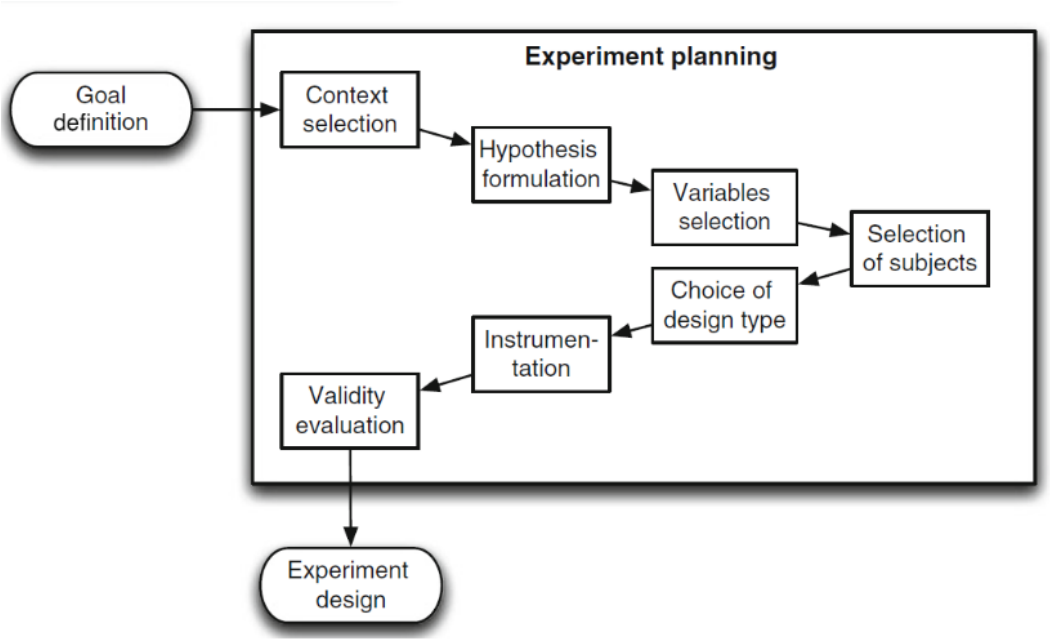

Figure 3.1: Overview of experiment planning.
(WOHLIN et al., 2012)

3.1
Goal

The research goal was elaborated using the Goal Question Metric (GQM) approach, proposed by Basili et al. (BASILI CALDIERA, 1994).

Analyze the <application of SOLID design principles>
for the purpose of <characterization>
with respect to their <impact on ML code understanding>
from the point of view of <data scientists>

in the context of <industrial ML code>.

3.2 Context Selection

The selected context consists of an off-line experiment (*i.e.*, without direct participation of the researchers), where data scientists (students and professionals, as detailed and characterized later) were randomly assigned to two different treatments and asked questions related to their understanding of real industrial ML code.

To increase representativeness, we decided to use the ML code of a real ML-enabled system created in partnership with the Exacta (ExACTa, 2023) initiative. The system was built to, based on an ML model prediction, emit alerts for oil refineries about the likelihood of emitting strong odors that could result in claims from the community. The ML code for this system was produced in Python using Jupyter Notebooks without employing SOLID design principles. It is noteworthy to mention that the system has been deployed and is currently in use in several oil refineries.

While being part of a specific solution, the ML code sample is representative of code that is typically part of ML applications, including code for tasks such as data loading, preprocessing, ML model building, and ML model evaluation. The treatments involved receiving either the original ML code or the original ML with a restructuring using the SOLID design principles and answering the same questions.

3.3 Hypothesis Formulation

We formulated the following null hypothesis (H0) and alternative hypothesis (H1). We aim to reject the H0 by showing statistically significant differences in the levels of code understanding with an alpha value of 0.1.

- H0: There is no relationship between applying the SOLID Design Principles and ML code understanding.
- H1: Applying the SOLID Design Principles improves ML code understanding.

3.4 Variables Selection

The independent variable of interest (experimental factor) corresponds to the application of the SOLID design principles. It can receive as treatment the use of SOLID principles or not using the principles. Other independent variables captured during the subject characterization included the experience with ML software development and the experience with SOLID principles.

The experimental study has two dependent variables. The first is the difficulty level of understanding of the source code perceived by the developer. Therefore, we prepared ordinal Likert scale questions on the difficulty in understanding the source code, which could assume the following values: 1 - Very Hard, 2 - Hard, 3 - Normal, 4 - Easy, 5 - Very Easy. The second is the level of agreement with statements related to the theoretical benefits of the SOLID principles. Again, we prepared ordinal Likert scale questions for these hypothesized benefits, which could assume the following values: 1 - Totally disagree, 2 - I disagree, 3 - I neither agree nor disagree, 4 - I agree, 5 - Totally agree.

3.5 Selection of Subjects

We used convenience sampling for our population of data scientists. The team of authors had access to data science graduate students from PUC-Rio and the University of Bari and to data science professionals from SERPRO, a large-scale public IT company with more than 8,000 employees in Brazil.

We managed to have access to samples of 32 data science students from the University of BARI, 32 data science students from PUC-Rio, and 36 professional data scientists from the SERPRO, totaling 100 participants. Within these samples, we used probabilistic quotas to randomly assign the subjects to the experimental treatments in a balanced way. We characterized the subjects to allow us to apply experiment design principles, such as blocking, if needed.

3.6 Choice of Design Type

The goal was to investigate whether applying SOLID design principles produces ML code that is easier to understand. The dependent variables are the level of understanding of the source code by the developer and the degree of agreement with statements related to the SOLID principles. Given this goal, we adopted a completely randomized one-factor with two treatments

design (WOHLIN et al., 2012). The design setup uses the same object (the ML code of a real ML-enabled system) with two treatments (original and restructured, applying SOLID design principles) and assigns the subjects randomly to each treatment. The experimental tasks concerned participants who analyzed the source code and answered questions concerning difficulty in understanding and answering questions about their degree of agreement with statements related to the SOLID principles.

3.6.1 Groups of participants

Participants were selected at random, by random selection of the available developers. They were separated by blocks of applied treatment.

3.6.2 Study Object

The study object of the experiment is a Source Code of ML software in the form of code snippets.

3.6.3 Factor

The factor is the Design Principle of the source code.

3.6.4 Treatments

We used two treatments. Treatment 1 was the use of SOLID Design Principles, and Treatment 2 was the not use of any Design Principle.

3.6.5 Tasks

The set of tasks is composed of Code Snippets based on real machine learning projects created in partnership with the Exacta (ExACTa, 2023) initiative. Where experiment participants need to analyze the source code and answer questions related to the perceived difficulty level of understanding the source code, which could assume the following values: 1 - Very Hard, 2 - Hard, 3 - Normal, 4 - Easy, 5 - Very Easy, and answer questions about their level of agreement with statements related to the theoretical benefits of the SOLID principles, which could assume the following values: 1- Totally disagree, 2- I disagree, 3- I neither agree nor disagree, 4- I agree, 5- Totally agree.

3.7 Instrumentation

We carefully designed and independently peer-reviewed the instrumentation, aiming to ensure that it would provide the necessary means for appropriately collecting data for the experiment. It consisted of two versions of an online questionnaire (one for each treatment), implemented using Google Forms, divided into 4 basic components: a consent form, a participant characterization form, the substantive questions related to the difficulty in understanding code snippets, and the levels of agreement with statements related to the SOLID principles, and a follow-up questionnaire. Both versions of the questionnaire are available in our online open science repository (ANONYMOUS, 2023). We detail each component of the instrumentation as follows.

3.7.1 Consent form

The consent form explains the research objectives, informs about their right to withdraw their participation at any time, and highlights the non-association of their name and e-mail address with the responses, ensuring that the research is conducted in accordance with ethical standards. It is noteworthy that all participants were volunteers.

3.7.2 Participant Characterization Form

The participant characterization form aims to collect relevant demographic information that may influence the study's results. The form includes questions related to academic background, proficiency in the English language, experience with software development, specific experience with ML software development, and levels of expertise in specific software coding-related topics, such as object-oriented programming, SOLID principles, Design Patterns, and Python.

3.7.3 Substantive questions related to the hypothesis

The substantive questions addressing the hypothesis include a set of code snippets from the real ML code. The set of ML code snippets, depending on the treatment, are the original ones or the restructured ones (applying SOLID design principles). Both instrument versions can be found in the online repository (ANONYMOUS, 2023).

Participants were required to analyze the code snippets and respond to closed-ended questions related to their perceived difficulty in understanding the code. Responses were registered using the five-point Likert scale: 1 - Very Hard, 2 - Hard, 3 - Normal, 4 - Easy, 5 - Very Easy. Additionally, participants are asked to provide feedback on their level of agreement with statements related to the SOLID principles, also on a five-point Likert scale: 1 - Totally disagree, 2 - I disagree, 3 - I neither agree nor disagree, 4 - I agree, 5 - Totally agree. For each closed-ended question, there was an open-ended question to provide justification for responses optionally.

3.7.4 Follow-up questionnaire

The follow-up questionnaire contains a closed-ended question regarding the participant's availability to answer further questions and two open-ended questions that provide an opportunity for feedback, allowing the participant to suggest improvements.

3.8 Operation and Data Collection

This section describes the operation and data collection conducted as part of the experiment.

3.8.1 Participant Selection

To conduct this experiment, it was necessary to select participants from three distinct groups: students from the University of Bari, students from the Software Engineering for Data Science course at PUC-Rio, and professionals from SERPRO. Participants were recruited voluntarily from these three sources.

3.8.1.1 University of Bari

This group is made up of students from the University of Bari Aldo Moro. They were recruited through appropriate means, such as email invitations and classroom announcements.

3.8.1.2 PUC-Rio

This group was composed of students from the Software Engineering for Data Science course at PUC-Rio, Pontifical Catholic University of Rio de Janeiro. Their participation was obtained through direct invitations and communications in the classroom.

3.8.1.3 SERPRO

Participants in this group were employees of SERPRO, a federal data processing company of Brazil. They were invited to participate in the experiment through internal company channels.

3.8.2 Online Questionnaires

The data collection instrument used was an online questionnaire created with Google Forms, available in Portuguese and English. Identical questionnaires were created for the three groups of participants, containing questions related to the dependent variables that were evaluated in this study.

3.8.3 Treatment Randomization Method

In this software engineering study, the choice of treatment applied to the participants was performed randomly. The aim was to ensure that the two treatments, Treatment 1 and Treatment 2, were impartially and randomly assigned to each participant in order to avoid any bias in data collection.

To achieve this randomization, an algorithm was implemented in JavaScript to run in the participant's browser, with the aim of randomly choosing one of the two available treatments. The algorithm followed the following steps:

1. First, it was verified that the participant's browser supported local storage to retain the treatment choice previously assigned to a participant. This was done using the following conditions.

Code 11: Local storage support verification.

```
1 if (typeof Storage !== "undefined" && localStorage.getItem('
      random_page')) {
```

2. If a participant already had a treatment choice assigned, the algorithm retrieved that choice and redirected the participant to the page corresponding to that treatment.

Code 12: Redirects to previous choice.

```
var webpage = localStorage.getItem('random_page');
document.getElementById("framey").src = webpage;
```

3. Otherwise, if the participant did not have a previous choice registered, the algorithm creates a list of web pages corresponding to the different treatments available. In this case, two web pages (Google Forms URLs) were included in the list.

Code 13: Create a page per treatment.

```
var webpages = [];
webpages.push("https://docs.google.com/forms/d/e/1
    FAIpQLSfKGte3XHdmjjcU4R5y4R7PAiXpP3eCpCocSJpTqB4v5WYMYw/
    viewform");
webpages.push("https://docs.google.com/forms/d/e/1FAIpQLSfwx-1
    IeaBEqFDRn8kRSxoopT8BH4EDdVMy1Rf3uPeUS48z7g/viewform");
```

4. Then, the algorithm randomly selects a web page from the list of available pages, using a random number generator to pick a random index from the list.

Code 14: Randomly select treatment webpage.

```
var random_page = webpages[Math.floor(Math.random() * webpages.
    length)];
```

5. The choice of treatment for the participant is recorded in local storage.

Code 15: Redirects to the treatment webpage.

```
if (typeof Storage !== "undefined") {
  localStorage.setItem('random_page', random_page);
}
```

6. Finally, the participant is redirected to the web page corresponding to the chosen treatment.

Code 16: Redirects to previous choice.

```
document.getElementById("framey").src = random_page;
```

3.8.4
Administration of Questionnaires

After the randomization of treatments, participants were redirected to the questionnaire corresponding to the assigned treatment. They were instructed to answer the questionnaire's questions based on their experiences and perceptions.

3.8.5
Data Collection

Data were collected from participant's responses to the online questionnaires. The data included participant consent, demographic information, answers to substantive questions related to the hypothesis, and information related to follow-up.

The dataset collected during the experiment is available in our online repository (ANONYMOUS, 2023).

3.9
Analysis Procedures

This section presents the statistical techniques that were applied to the data collected by the experiment.

3.9.1
Tabulation and Graphics

First, the data were tabulated by participants, including their answers about consent to participate in the research, answers about their characterization, answers about levels of understanding perceived by the developer of the presented source codes, and answers about levels of agreement with statements related to the SOLID principles, grouping by institution of participant's origin and type of treatment applied in the experiment.

After tabulation, the data were loaded into a Colab Notebook (LLC, 2023), using the Python language (FOUNDATION, 2023), with the help of the Pandas data analysis library (NUMFOCUS, 2023).

With the data already loaded into the Colab Notebook, frequency tables were created and printed for each answer on the characterization form. In order to understand the characterization of the research participants.

Soon after, the research used bar graphs to visualize the frequency distribution of responses on source code understanding and levels of agreement with statements related to the SOLID principles. These frequencies were normalized and grouped by participant's origin and type of treatment applied.

3.9.2 Descriptive Statistics

To give an overview of the overall distribution of responses provided by participants in the experiment, we calculated measures of central tendency, mean, median, and mode.

To understand the distribution of the data, we calculated two measures of dispersion: the mean absolute deviation and the standard deviation.

3.9.3 Outlier Analysis

Assuming that the dependent variables are limited to an ordinal scale, with predefined values from 1 to 5, technically, there are no outliers, as all responses are limited to this discrete interval.

3.9.4 Hypothesis Testing

How to choose between parametric tests and non-parametric tests depends on the characteristics of your data and the assumptions underlying your statistical tests. Analyzing the results of measures of central tendency and measures of dispersion, it is not possible to assume that the samples meet certain assumptions about the distribution of the data, such as the normal distribution, which led us to perform non-parametric hypothesis tests.

With regard to our samples, in addition to not being able to assume a normal distribution, they are also independent. So, the non-parametric test that proved to be the most appropriate was the Mann-Whitney test. To measure this statistical effect size, we will use Cohen's d, which is widely used and easy to interpret.

4
Experimental Study Results

In this chapter, we describe the participant characterization and present the experimental study results.

4.1
Participant Characterization

In this section, we describe the population of participants involved in our experiment. The composition of this population, including its relevant demographic data, is essential for understanding the database upon which our study is based.

4.1.1
Participants by source and treatment

The participants were randomly assigned to the two treatments: SOLID (the SOLID restructured ML code) and unstructured (the original ML code). As described, participants were selected from three sources: University of BARI, PUC-Rio, and SERPRO. The distribution of participants for each treatment and source is presented in Table 4.1. It is possible to observe that the equal likelihood of being assigned to one treatment or the other led to almost balanced distributions of participants between treatments.

Table 4.1: Participants by source and treatment.

Treatment	SOLID	Unstructured
BARI	15	17
PUC	16	16
SERPRO	17	19
Total	48	52

4.1.2
Level of education and relation to computer science

The distribution of the level of education of participants per treatment and source is shown in Table 4.2, which also shows whether participants had academic degrees related to computer science or not. It is possible to observe that the random assignment led to some differences, which we do not believe to affect the results. In particular, slightly higher educational levels can be observed for the Unstructured treatment at University of BARI and for the SOLID treatment at PUC-Rio, while the ones for the SERPRO Company are

comparable. It is also possible to observe a balance between participants with a computer science degree and from other areas. It is noteworthy, however, that the sources for subject selection may have led to a higher number of data scientists with a computer science background than in general. The data science graduate programs at both universities were part of the informatics departments, and the company was an IT company. We will further discuss this representativeness threat later.

Table 4.2: Level of education and relation to computer science.

Source	BARI		PUC		SERPRO	
Treatment	SOLID	Unstructured	SOLID	Unstructured	SOLID	Unstructured
Level of education						
Bachelor's Degree.	13	12	4	10	2	4
Specialization.	1	1	1	1	9	10
Master's Degree.	1	4	8	4	4	5
Doctoral Degree.	0	0	3	1	2	0
Computer Science	10	11	11	12	14	17
Others	5	6	5	4	3	2

4.1.3 Experience with software development

The participants' experience with software development in general, as characterized across different treatments and sources, is presented in Table 4.3. Additionally, this table also shows the participants' years of experience in software development. Similarly, Table 4.4 shows the experience with specific ML-related software development. It is possible to observe that, in terms of experience, the control and experimental groups are comparable.

Table 4.3: Experience with software development.

Source	BARI		PUC		SERPRO	
Treatment	SOLID	Unstructured	SOLID	Unstructured	SOLID	Unstructured
Experience with software development						
I have never developed software.	0	0	1	0	0	0
I have been developing for my own use.	7	12	6	7	7	9
I have been developing as team member, related to a course.	14	16	8	7	10	12
I have been developing as team member, in industry.	1	5	12	16	17	17
0-2 years	3	3	4	4	1	1
3-5 years	9	10	4	6	1	4
6-10 years	3	4	4	4	2	0
More than 10 years	0	0	4	2	13	14

Table 4.4: Experience with ML software development.

Source	BARI		PUC		SERPRO	
Treatment	SOLID	Unstructured	SOLID	Unstructured	SOLID	Unstructured
Experience with Machine Learning Software Development						
I never developed ML software.	3	4	1	0	0	1
I developed ML software for my own use.	4	6	6	8	10	7
I developed ML software as a team member, related to a course.	10	11	9	11	14	13
I developed ML software as a team member in the industry.	0	0	8	11	10	12
0-2 years	12	15	10	11	6	8
3-5 years	2	1	3	2	7	6
6-10 years	1	1	1	3	2	5
Greater than 10 years	0	0	2	0	2	0

4.1.4 Experience with topics related to software engineering

Finally, an overview of participants' background on software engineering and development topics related to this research, encompassing object-oriented programming, SOLID principles, and Python proficiency (among other collected data available in our online dataset), is presented in Table 4.5.

It is possible to observe that all of the participants had some experience with object-oriented programming and Python, with different experience levels of using SOLID principles. This data collectively provides valuable information about participants' backgrounds and competencies in software design aspects, offering context for interpreting their responses.

4.1.5 English reading and comprehension skills level

This subsection presents the English reading and comprehension skills of our participants across different treatment groups and sources. The data presented in Table 4.6 summarizes the distribution of participants based on their English language proficiency levels.

Table 4.6: English reading and comprehension skills level.

Source	BARI		PUC		SERPRO	
Treatment	SOLID	Unstructured	SOLID	Unstructured	SOLID	Unstructured
English reading and comprehension skills level						
Basic.	2	4	1	1	2	3
Intermediate.	10	7	6	2	6	7
Advanced.	3	6	9	13	9	9

This information is crucial for understanding the English language proficiency levels of our participants within each treatment and source. It provides valuable insights into the language capabilities of our study participants, which may have implications for their understanding of source code and comprehension of questionnaire questions.

Table 4.5: Experience with topics related to software engineering.

Source	BARI		PUC		SERPRO	
Treatment	SOLID	Unstructured	SOLID	Unstructured	SOLID	Unstructured
Object-oriented programming						
I studied in a classroom or in a book.	2	1	1	1	0	1
I actively practiced in a classroom project.	9	11	4	3	0	2
I used it in one project in industry.	2	2	1	7	2	4
I used it in several projects in industry.	2	3	10	5	15	12
Encapsulation						
No experience.	0	0	0	1	0	0
I studied in a classroom or in a book.	2	3	3	2	0	2
I actively practiced in a classroom project.	10	8	3	2	1	3
I used it in one project in industry.	2	3	1	6	2	6
I used it in several projects in industry.	1	3	9	5	14	8
Inheritance, Abstract Classes and Interfaces						
No experience.	0	0	0	1	0	0
I studied in a classroom or in a book.	3	2	4	1	0	1
I actively practiced in a classroom project.	9	9	2	4	0	3
I used it in one project in industry.	2	3	1	6	3	5
I used it in several projects in industry.	1	3	9	4	14	10
Clean Code						
No experience.	1	1	3	0	4	6
I studied in a classroom or in a book.	2	2	4	3	2	3
I actively practiced in a classroom project.	9	8	1	3	3	1
I used it in one project in industry.	3	3	2	6	2	6
I used it in several projects in industry.	0	3	6	4	6	3
SOLID Principles						
No experience.	2	3	4	0	6	7
I studied in a classroom or in a book.	5	4	5	5	4	4
I actively practiced in a classroom project.	8	6	2	5	2	4
I used it in one project in industry.	0	2	3	5	0	1
I used it in several projects in industry.	0	2	2	1	5	3
Design Patterns						
No experience.	0	1	3	2	1	1
I studied in a classroom or in a book.	4	6	4	4	3	2
I actively practiced in a classroom project.	8	5	1	4	3	5
I used it in one project in industry.	1	3	2	5	3	5
I used it in several projects in industry.	2	2	6	1	7	6
Domain-Driven Design (DDD)						
No experience.	6	10	6	2	7	6
I studied in a classroom or in a book.	2	2	4	9	3	5
I actively practiced in a classroom project.	6	3	1	3	1	3
I used it in one project in industry.	1	1	3	1	4	4
I used it in several projects in industry.	0	1	2	1	2	1
Python						
I studied in a classroom or in a book.	1	4	2	0	0	1
I actively practiced in a classroom project.	10	7	1	2	6	5
I used it in one project in industry.	2	3	7	2	2	2
I used it in several projects in industry.	2	3	6	12	9	11

4.2 Results

This session presents the results of questions related to our hypotheses organized by SOLID principle (MARTIN, 2000).

4.2.1 SRP - Single Responsibility Principle

4.2.1.1 Question 1

What is your perception about understanding the source code?

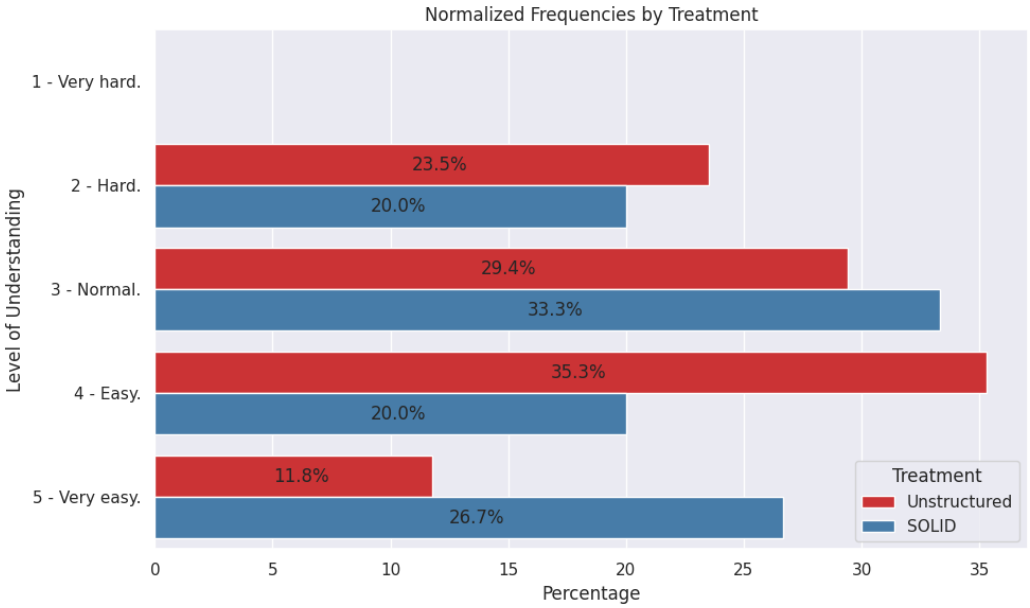

Figure 4.1: BARI - Question 1 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	3.54	3	[3]
Unstructured Code (Treatment 2)	3.36	3	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.97	1.13	
Unstructured Code (Treatment 2)	0.85	1.0	
Hypothesis Test			
p-value	0.34056528361427313		
Interpretation	Does not reject the null hypothesis: There is no evidence that the median of 'SOLID' is greater than the median of 'Unstructured'.		

Table 4.7: BARI - Question 1 - Statistics.

The results of question 1 in BARI are presented in Figure 4.1, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.7 presents measures of central tendency, measures of dispersion, and results of the hypothesis test.

Although the non-parametric test Mann-Whitney did not find statistically significant evidence to reject the null hypothesis, we can observe a noticeable difference in the perception of comprehension between Treatment 1 (SOLID) and Treatment 2 (Unstructured Code) based on participants' responses.

With Treatment 1 (SOLID), 46.7% of the participants classified comprehension as "Easy" or "Very Easy." In Treatment 2 (Unstructured Code), 46.1% of the participants classified comprehension as "Easy" or "Very Easy."

While the difference in those classifying it as "Easy" or "Very Easy" is not very large, with Treatment 1 (SOLID), a substantial portion of participants, approximately 26.7%, classified comprehension as "Very Easy." This indicates that many respondents found the SOLID code extremely easy to understand. However, with Treatment 2 (Unstructured Code), only about 11.8% of participants classified comprehension as "Very Easy." This suggests that a considerably smaller proportion of respondents found the unstructured code as easy to understand as the SOLID code.

Therefore, even though the statistical test did not show a significant difference, the qualitative analysis of the responses also suggests that Treatment 1 (SOLID) results in a better understanding of the code, as participants considered it more organized and easier to understand. As an example, a participant subjected to Treatment 1 (SOLID) commented: "The code is commented, and I have already used Python OOP." Another participant stated: "It's easy to understand all the code." On the other hand, even though finding the code easy to understand, a participant subjected to the original code, Treatment 2 (Unstructured Code), commented: "Usually, a Python developer does not know the implementation of the language in order to write code."

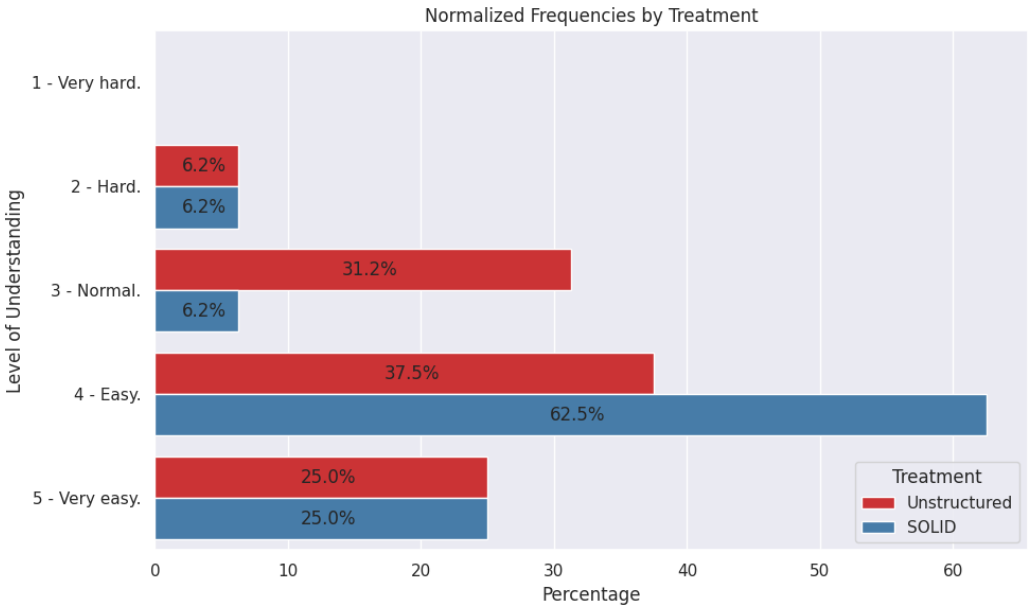

Figure 4.2: PUC - Question 1 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.07	4	[4]
Unstructured Code (Treatment 2)	3.82	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.47	0.78	
Unstructured Code (Treatment 2)	0.74	0.92	
Hypothesis Test			
p-value	0.19019767217258426		
Interpretation	Does not reject the null hypothesis: There is no evidence that the median of 'SOLID' is greater than the median of 'Unstructured'.		

Table 4.8: PUC - Question 1 - Statistics.

The results of question 1 in PUC are presented in Figure 4.2, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.8 presents measures of central tendency, measures of dispersion, and results of the hypothesis test.

Although the non-parametric Mann-Whitney test did not yield statistically significant evidence to reject the null hypothesis, we can observe that using Treatment 1 (SOLID) leads to better comprehension based on participants' responses.

In Treatment 1 (SOLID), a substantial percentage of participants, approximately 87.5%, rated the code as either "Easy" or "Very easy" to understand. Participants in this group praised the code's organization, modularity, clear variable names, and the presence of comments. For instance, one participant commented, "The code is well-commented and organized. The classes are self-explanatory".

In contrast, for Treatment 2 (Unstructured Code), only about 62.5% of participants rated the code as "Easy" or "Very easy" to understand. Respondents in this group raised concerns about code organization, the lack of modularity, and the clarity of variable names. For instance, a participant from this group commented, "I believe that the code is not well-written and difficult to understand for programmers in general.".

Therefore, even though the statistical test did not yield significant results, a qualitative analysis of participant responses strongly indicates that Treatment 1 (SOLID) leads to enhanced code comprehension. Participants consistently noted that this treatment resulted in a more organized, modular, and easily comprehensible codebase. This qualitative feedback, in conjunction with the quantitative data, underscores the advantages of Treatment 1 (SOLID) for improving code understandability.

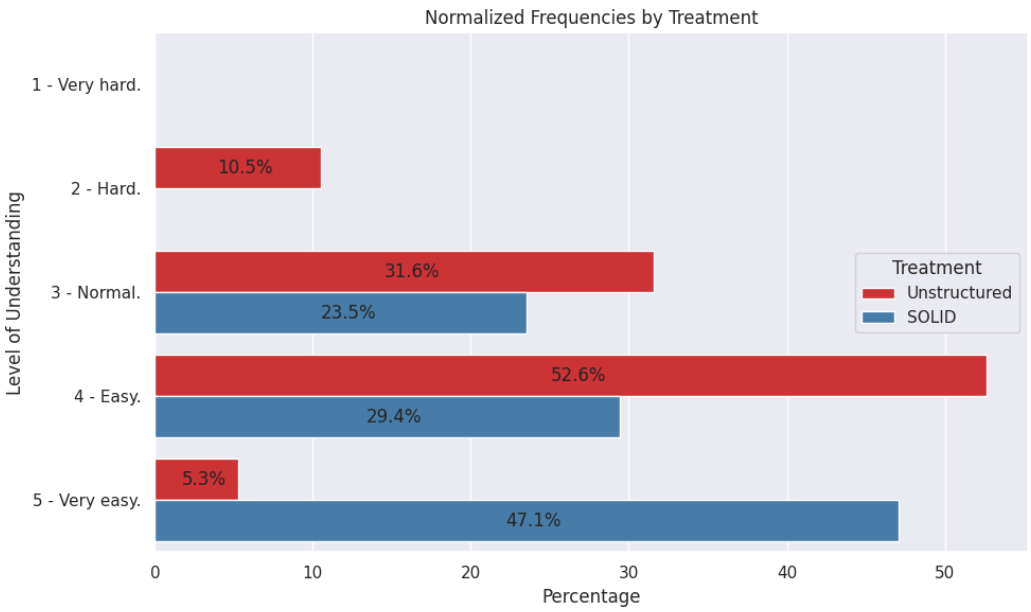

Figure 4.3: SERPRO - Question 1 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.24	4	[5]
Unstructured Code (Treatment 2)	3.53	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.72	0.84	
Unstructured Code (Treatment 2)	0.66	0.78	
Hypothesis Test			
p-value	0.00857908778636166		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.88		

Table 4.9: SERPRO - Question 1 - Statistics.

The results of question 1 in SERPRO are presented in Figure 4.3, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.9 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-value was 0.0086, which is below the alpha value of 0.1, with a large effect size of 0.88. This indicates that there is statistical evidence suggesting that Treatment 1 (SOLID) leads to a better understanding of the source code than Treatment 2 (Unstructured Code).

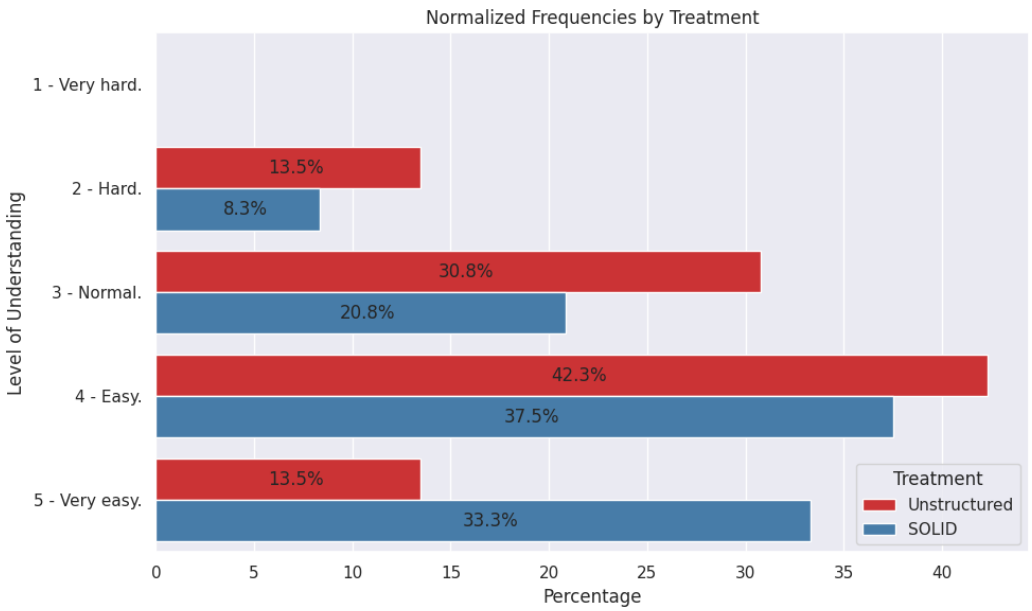

Figure 4.4: OVERALL - Question 1 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	3.96	4	[4]
Unstructured Code (Treatment 2)	3.56	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.73	0.95	
Unstructured Code (Treatment 2)	0.77	0.9	
Hypothesis Test			
p-value	0.013864774287158765		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.43		

Table 4.10: OVERALL - Question 1 - Statistics.

The results of question 1 in all groups are presented in Figure 4.4, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.10 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-value was 0.0139, which is below the alpha value of 0.1, with an effect size of 0.43, very close to being considered a medium effect. This indicates that there is statistical evidence suggesting that Treatment 1 (SOLID) leads to a better understanding of the source code than Treatment 2 (Unstructured Code).

4.2.1.2 Question 3

The responsibilities in the code above are clearly distributed and defined. What is your degree of agreement with the statement?

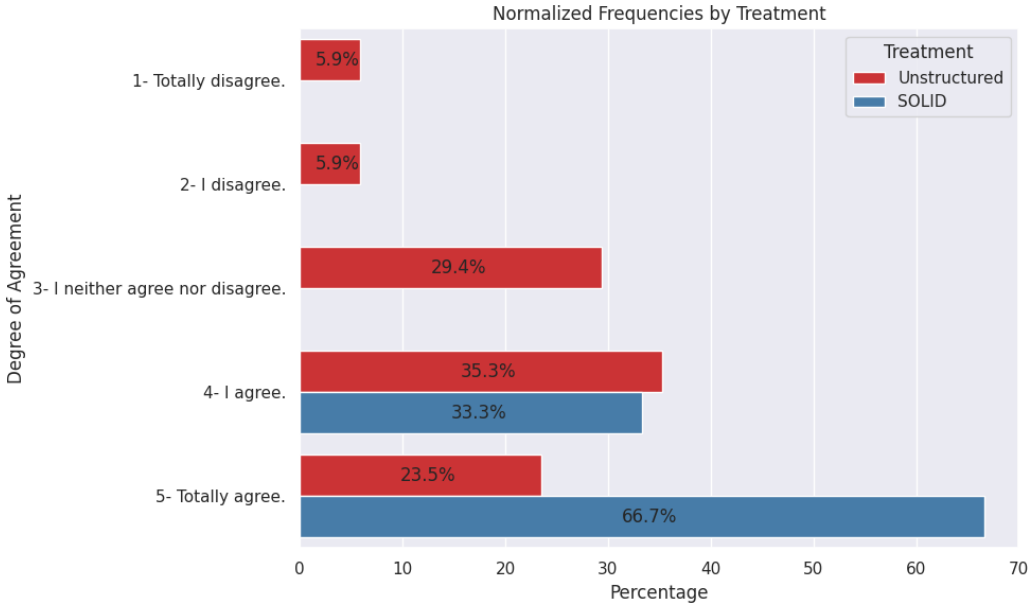

Figure 4.5: BARI - Question 3 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.67	5	[5]
Unstructured Code (Treatment 2)	3.65	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.45	0.49	
Unstructured Code (Treatment 2)	0.89	1.12	
Hypothesis Test			
p-value	0.0018036147129539043		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.15		

Table 4.11: BARI - Question 3 - Statistics.

The results of question 3 in BARI are presented in Figure 4.5, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.11 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0018, which is below the alpha value of 0.1, with a large effect size of 1.15. This indicates that there is statistical evidence to suggest that

Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

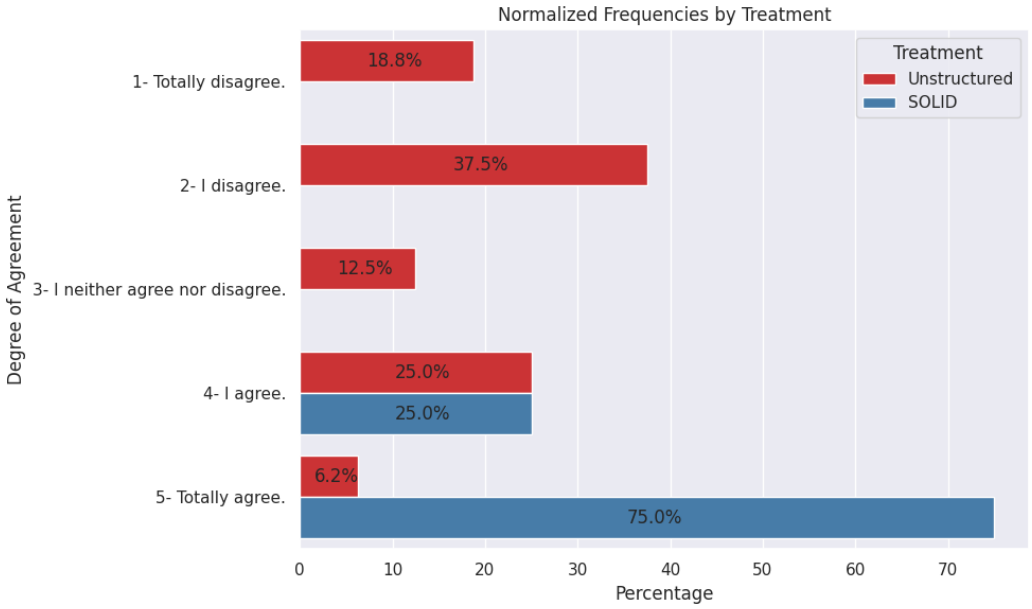

Figure 4.6: PUC - Question 3 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.75	5	[5]
Unstructured Code (Treatment 2)	2.63	2	[2]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.38	0.45	
Unstructured Code (Treatment 2)	1.08	1.26	
Hypothesis Test			
p-value	7.630706162350667e-06		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	2.25		

Table 4.12: PUC - Question 3 - Statistics.

The results of question 3 in PUC are presented in Figure 4.6, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.12 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 2.25. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

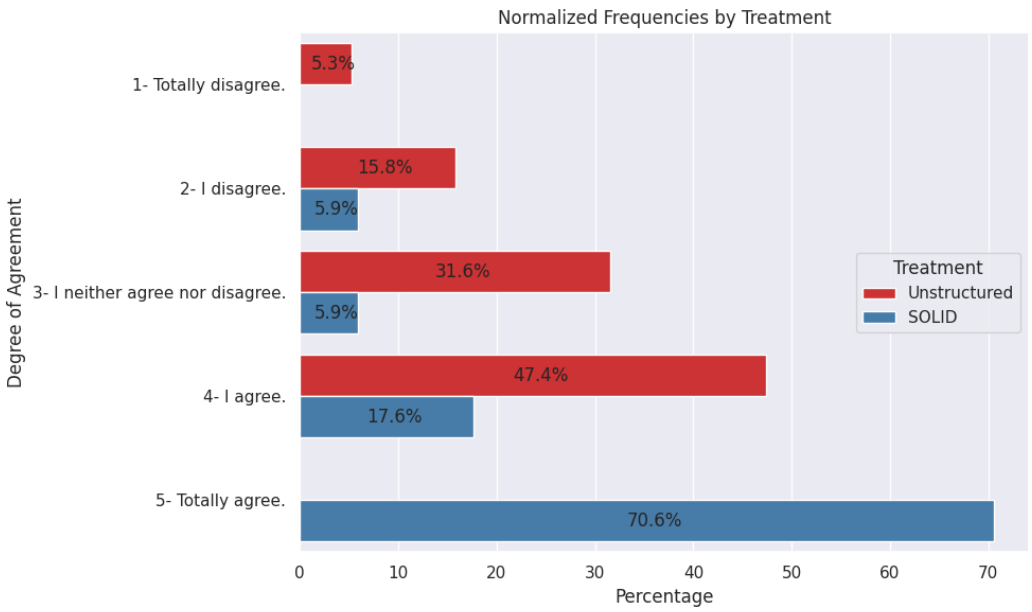

Figure 4.7: SERPRO - Question 3 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.53	5	[5]
Unstructured Code (Treatment 2)	3.22	3	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.67	0.88	
Unstructured Code (Treatment 2)	0.75	0.92	
Hypothesis Test			
p-value	4.144861139609103e-05		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.46		

Table 4.13: SERPRO - Question 3 - Statistics.

The results of question 3 in SERPRO are presented in Figure 4.7, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.13 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 1.46. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

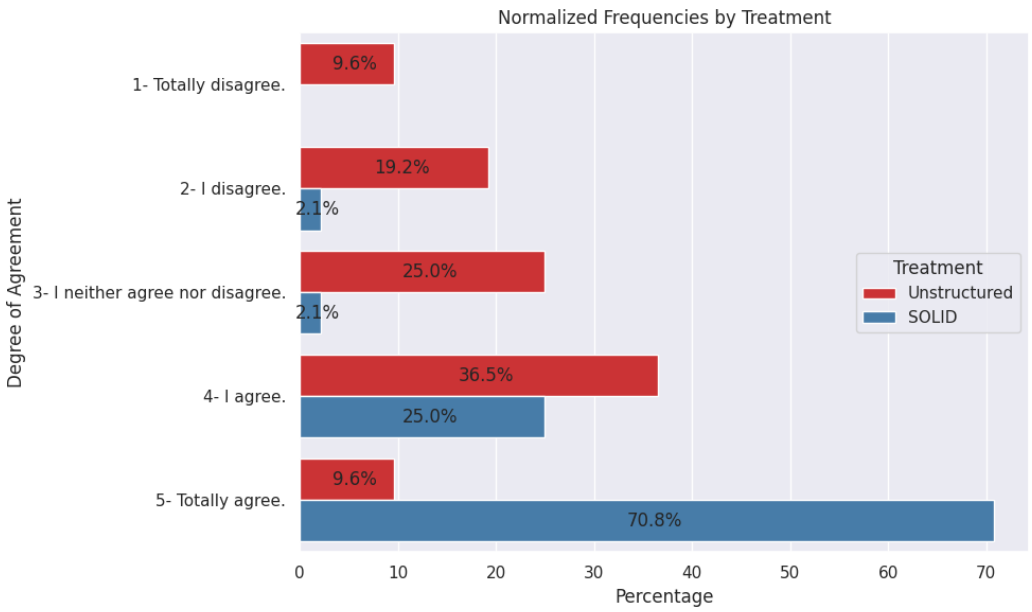

Figure 4.8: OVERALL - Question 3 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.65	5	[5]
Unstructured Code (Treatment 2)	3.18	3	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.51	0.64	
Unstructured Code (Treatment 2)	0.96	1.15	
Hypothesis Test			
p-value	1.603950527806893e-11		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.56		

Table 4.14: OVERALL - Question 3 - Statistics.

The results of question 3 in all groups are presented in Figure 4.8, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.14 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 1.56. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

4.2.1.3 Question 5

The code block above has classes structured to have a single responsibility in the software, that is, they are specialized in a single subject. What is your degree of agreement with the statement?

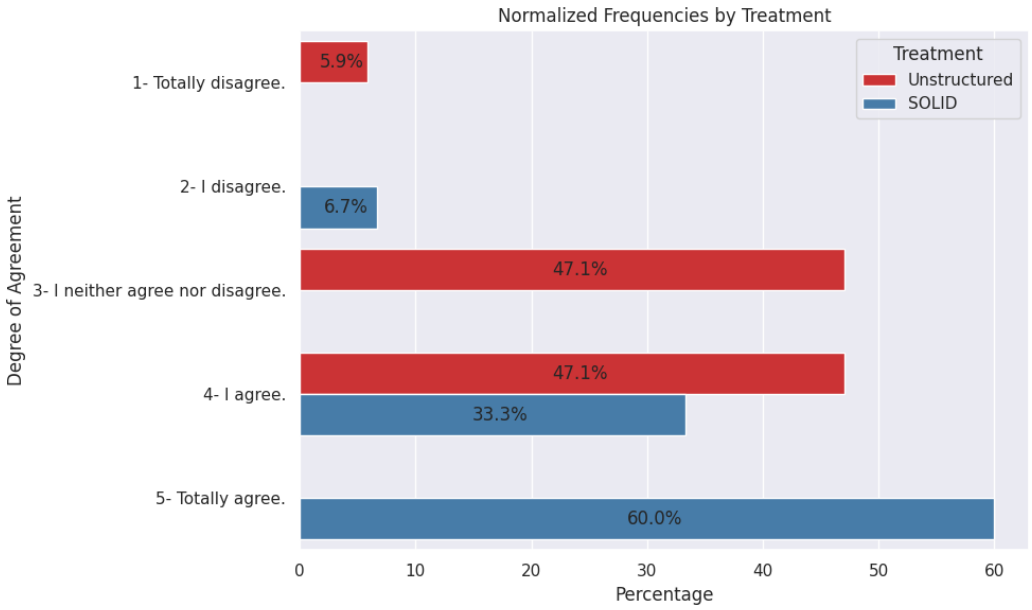

Figure 4.9: BARI - Question 5 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.47	5	[5]
Unstructured Code (Treatment 2)	3.36	3	[3 4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.64	0.84	
Unstructured Code (Treatment 2)	0.61	0.79	
Hypothesis Test			
p-value	0.00014139010604550343		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.37		

Table 4.15: BARI - Question 5 - Statistics.

The results of question 5 in BARI are presented in Figure 4.9, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.15 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 1.37. This indicates that there is statistical evidence to suggest that

Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

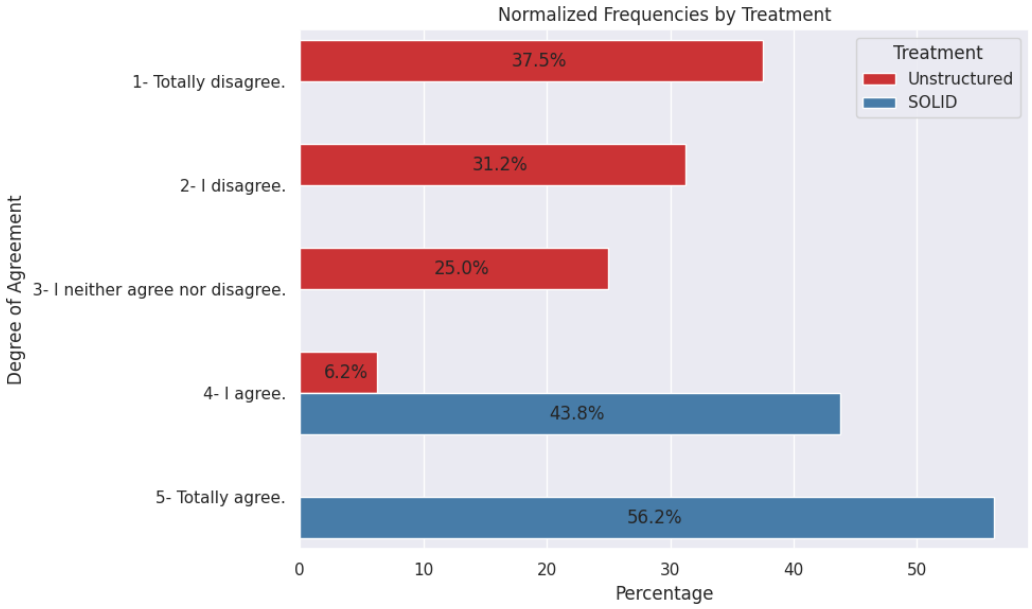

Figure 4.10: PUC - Question 5 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.57	5	[5]
Unstructured Code (Treatment 2)	2.0	2	[1]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.5	0.52	
Unstructured Code (Treatment 2)	0.75	0.97	
Hypothesis Test			
p-value	8.21212453080668e-07		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	3.31		

Table 4.16: PUC - Question 5 - Statistics.

The results of question 5 in PUC are presented in Figure 4.10, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.16 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 3.31. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

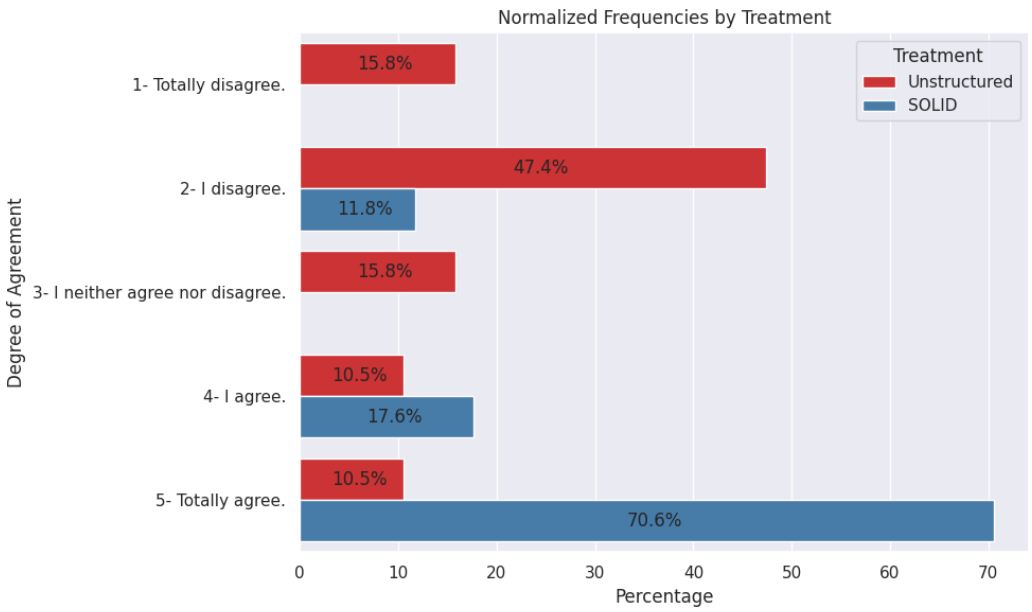

Figure 4.11: SERPRO - Question 5 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.48	5	[5]
Unstructured Code (Treatment 2)	2.53	2	[2]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.75	1.01	
Unstructured Code (Treatment 2)	0.99	1.22	
Hypothesis Test			
p-value	5.064673811903765e-05		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.72		

Table 4.17: SERPRO - Question 5 - Statistics.

The results of question 5 in SERPRO are presented in Figure 4.11, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.17 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 1.72. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

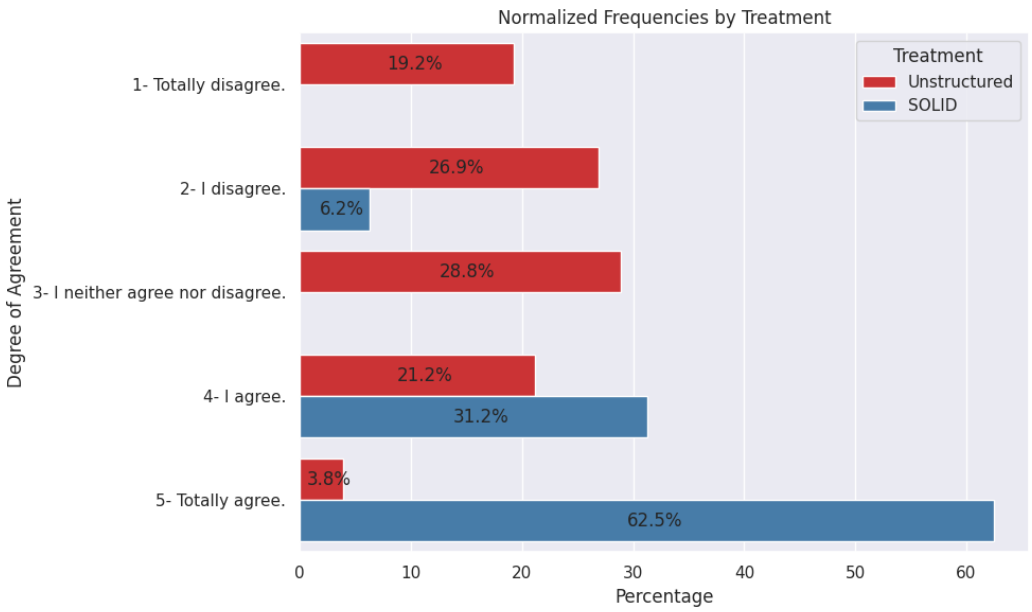

Figure 4.12: OVERALL - Question 5 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.5	5	[5]
Unstructured Code (Treatment 2)	2.64	3	[3]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.63	0.8	
Unstructured Code (Treatment 2)	0.98	1.14	
Hypothesis Test			
p-value	5.620884537172174e-13		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.88		

Table 4.18: OVERALL - Question 5 - Statistics.

The results of question 5 in all groups are presented in Figure 4.12, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.18 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 1.88. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

4.2.1.4 Question 7

The code above is structured so that your classes have a single reason for change. What is your degree of agreement with the statement?

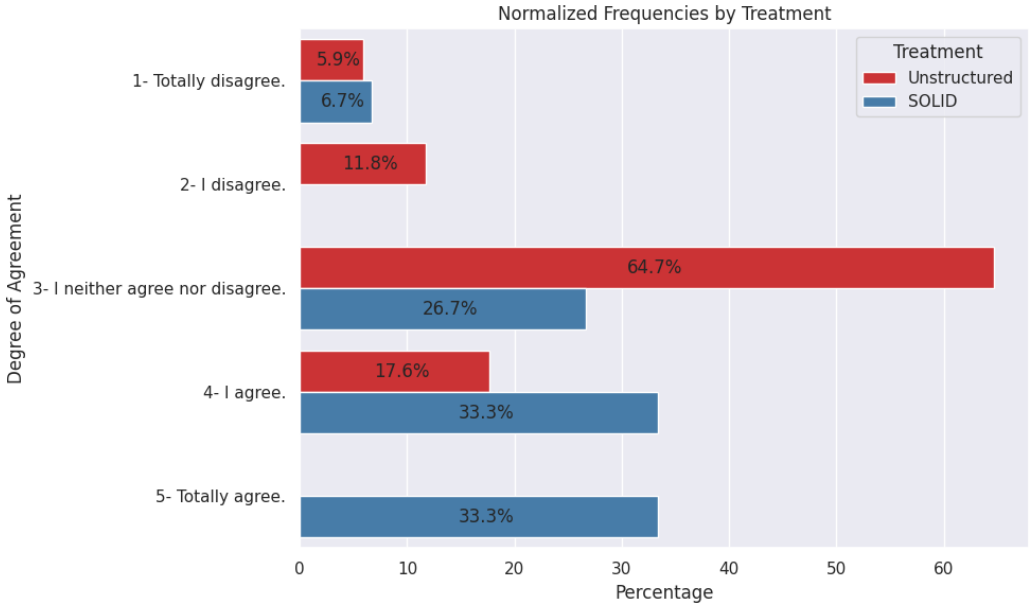

Figure 4.13: BARI - Question 7 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	3.87	4	[4 5]
Unstructured Code (Treatment 2)	2.95	3	[3]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.85	1.13	
Unstructured Code (Treatment 2)	0.45	0.75	
Hypothesis Test			
p-value	0.002710597738677753		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.98		

Table 4.19: BARI - Question 7 - Statistics.

The results of question 7 in BARI are presented in Figure 4.13, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.19 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0027, which is below the alpha value of 0.1, with a large effect size of 0.98. This indicates that there is statistical evidence to suggest that

Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

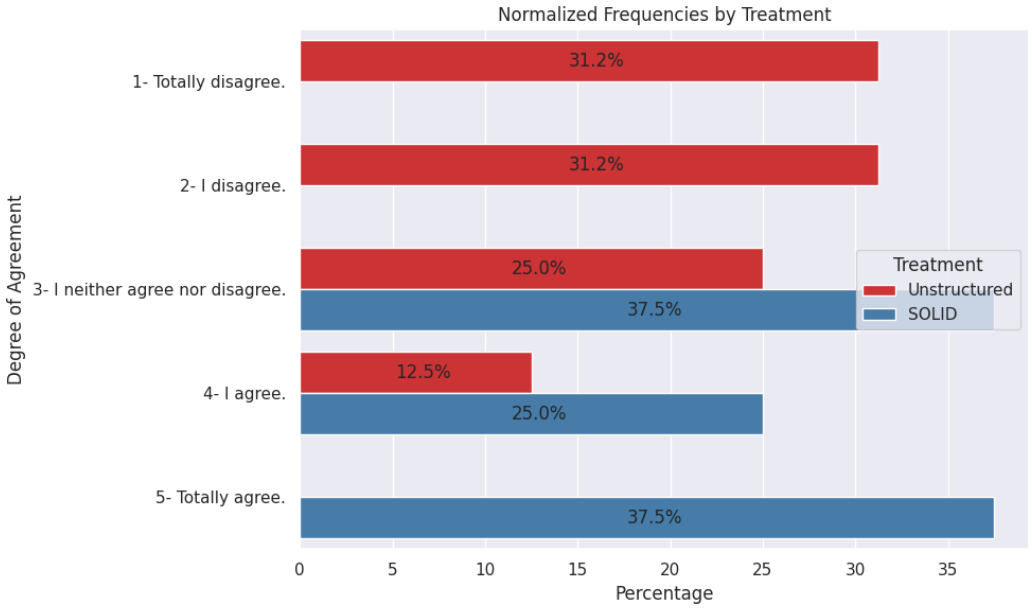

Figure 4.14: PUC - Question 7 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.0	4	[3 5]
Unstructured Code (Treatment 2)	2.19	2	[1 2]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.75	0.9	
Unstructured Code (Treatment 2)	0.86	1.05	
Hypothesis Test			
p-value	5.9474010623007696e-05		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.86		

Table 4.20: PUC - Question 7 - Statistics.

The results of question 7 in PUC are presented in Figure 4.14, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.20 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 1.86. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

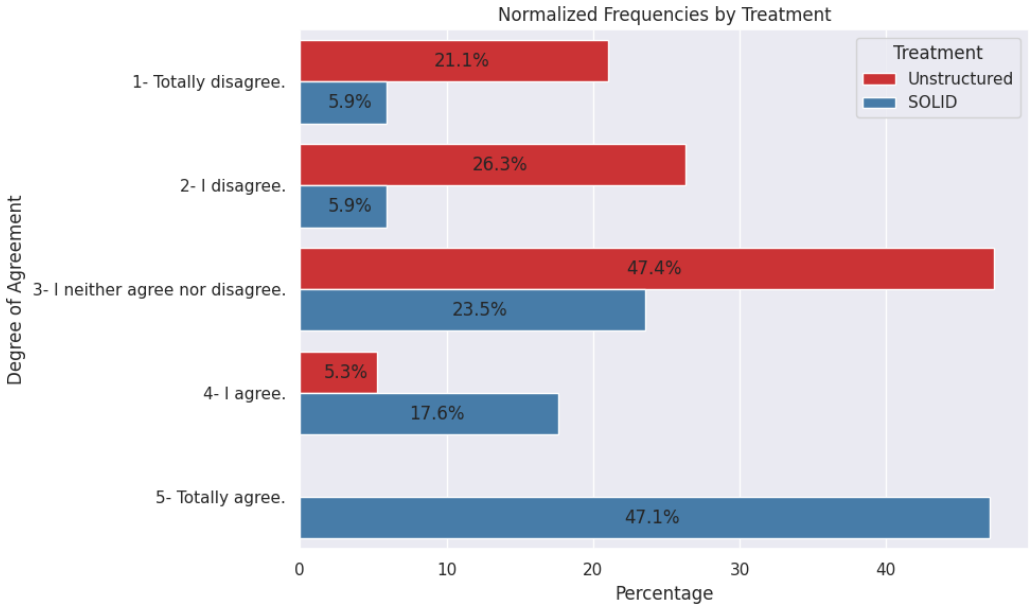

Figure 4.15: SERPRO - Question 7 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	3.95	4	[5]
Unstructured Code (Treatment 2)	2.37	3	[3]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	1.02	1.25	
Unstructured Code (Treatment 2)	0.78	0.9	
Hypothesis Test			
p-value	0.00019919425164870 31		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.46		

Table 4.21: SERPRO - Question 7 - Statistics.

The results of question 7 in SERPRO are presented in Figure 4.15, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.21 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0002, which is below the alpha value of 0.1, with a large effect size of 1.46. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

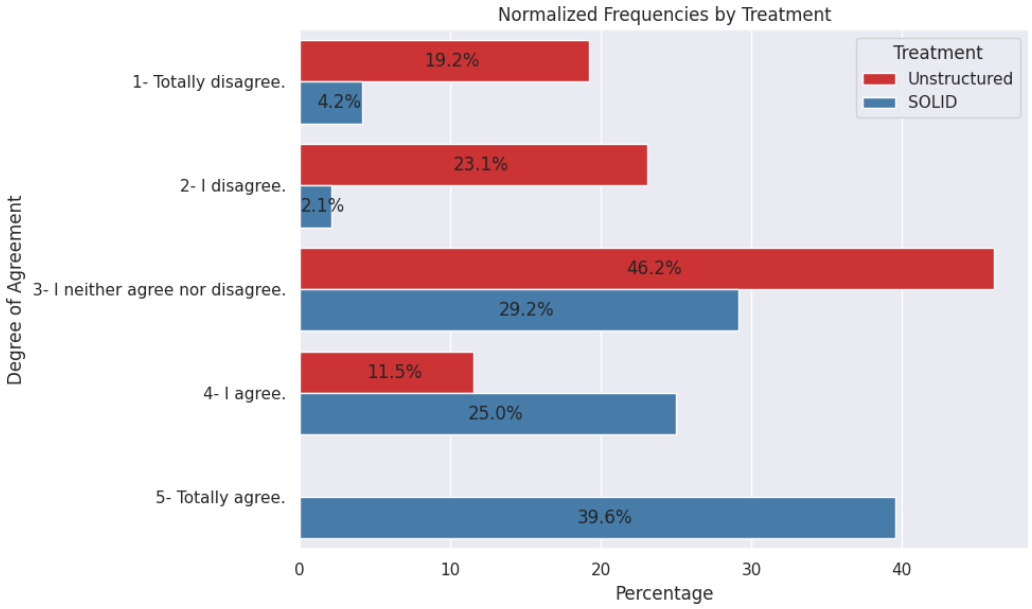

Figure 4.16: OVERALL - Question 7 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	3.94	4	[5]
Unstructured Code (Treatment 2)	2.5	3	[3]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.88	1.08	
Unstructured Code (Treatment 2)	0.81	0.94	
Hypothesis Test			
p-value	1.4343993338573635e-09		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.42		

Table 4.22: OVERALL - Question 7 - Statistics.

The results of question 7 in all groups are presented in Figure 4.16, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.22 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 1.42. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

4.2.2 OCP - Open-Closed Principle

4.2.2.1 Question 9

What is your perception about understanding the source code?

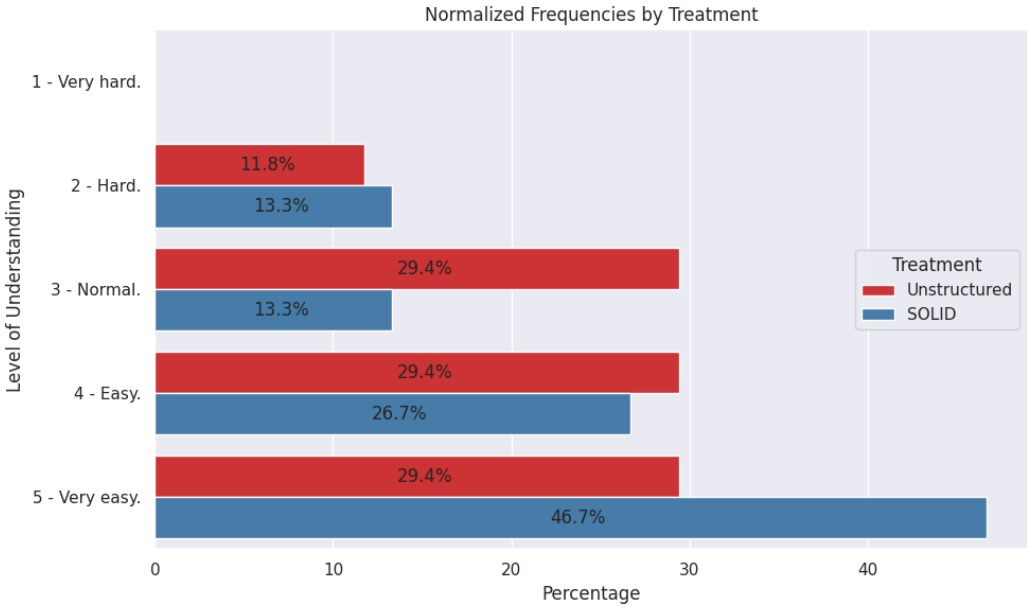

Figure 4.17: BARI - Question 9 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.07	4	[5]
Unstructured Code (Treatment 2)	3.77	4	[3 4 5]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.88	1.1	
Unstructured Code (Treatment 2)	0.87	1.04	
Hypothesis Test			
p-value	0.18700567415489328		
Interpretation	Does not reject the null hypothesis: There is no evidence that the median of 'SOLID' is greater than the median of 'Unstructured'.		

Table 4.23: BARI - Question 9 - Statistics.

The results of question 9 in BARI are presented in Figure 4.17, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.23 presents measures of central tendency, measures of dispersion, and results of the hypothesis test.

Although the result of the non-parametric hypothesis test did not show a statistically significant difference between the Treatment 1 (SOLID) and Treatment 2 (Unstructured Code) groups in relation to the perception of understanding the source code, it is still possible to argue that the use of

Treatment 1 (SOLID) leads to better understanding based on the answers and their justifications.

The majority of participants who received Treatment 1 (SOLID) classified their understanding of the code as "Easy" or "Very easy" (73.4%). On the other hand, in the group that received Treatment 2 (Unstructured Code), a smaller portion of participants classified understanding as "Easy" or "Very easy" 58.8%. Participants in Treatment 1 (SOLID) mentioned justifications that point to the clarity and understanding of the code. Some examples include "The code uses the same architecture and just adds three more models." and "The code is smaller and well written." In Treatment 2 (Unstructured Code), the justifications are not as consistent in highlighting the clarity of the code, and some mention difficulties or confusion, such as "It looks very confusing."

Although the non-parametric test did not show a significant difference in the medians, it is important to note that the distribution of responses in Treatment 1 (SOLID) is inclined towards more positive responses, while the Treatment 2 group (Unstructured Code) has a more balanced distribution between different levels of understanding.

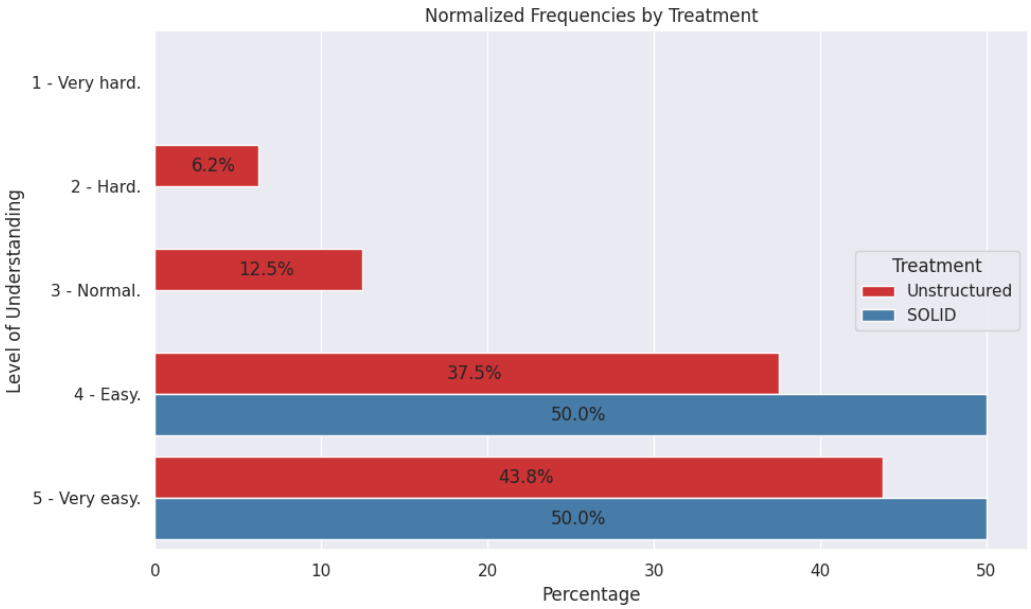

Figure 4.18: PUC - Question 9 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.5	4	[4 5]
Unstructured Code (Treatment 2)	4.19	4	[5]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.5	0.52	
Unstructured Code (Treatment 2)	0.72	0.92	
Hypothesis Test			
p-value	0.20762403448905742		
Interpretation	Does not reject the null hypothesis: There is no evidence that the median of 'SOLID' is greater than the median of 'Unstructured'.		

Table 4.24: PUC - Question 9 - Statistics.

The results of question 9 in PUC are presented in Figure 4.18, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.24 presents measures of central tendency, measures of dispersion, and results of the hypothesis test.

Even though the hypothesis test results did not demonstrate a statistically significant difference between Treatment 1 (SOLID) and Treatment 2 (Unstructured Code) groups regarding their perception of code comprehension, we can still argue that the use of Treatment 1 (SOLID) leads to a better understanding based on participant's responses and justifications.

In the group that received Treatment 1 (SOLID), 100% of the participants classified their code comprehension as "Easy" or "Very easy." They emphasized that class modularization, clarity of module responsibilities, and the use of Object-Oriented (OO) principles made the code easily understandable. For example, one participant mentioned, "The OO principles used allow us to focus on the broader view of the code based on the understanding of each module's responsibility. We can even abstract the understanding of model implementation details or preprocessing." These responses and justifications indicate a high satisfaction with code comprehension in the Treatment 1 (SOLID) group.

On the other hand, in the group that received Treatment 2 (Unstructured Code), although 81.3% of the participants classified code comprehension as "Easy" or "Very easy," some rated it as "Difficult" or "Normal." Some participants mentioned that the code was clear due to comments but not practical to read. They highlighted that the code could be better structured and more modular. Additionally, one participant noted, "The code remains clear despite being repetitive," suggesting that the code could be optimized.

While the hypothesis test did not reveal a significant difference, it is important to note that all responses in the Treatment 1 (SOLID) group

were positive, indicating a clear preference for superior comprehension. In contrast, in the Treatment 2 (Unstructured Code) group, where some responses indicated that the code was clear due to comments, it was not well-structured.

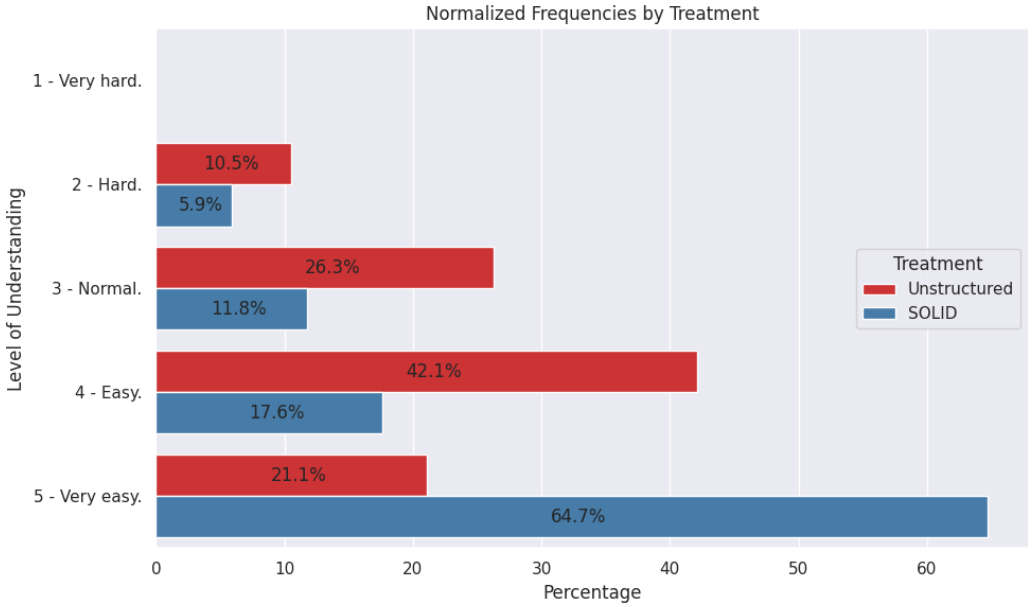

Figure 4.19: SERPRO - Question 9 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.42	5	[5]
Unstructured Code (Treatment 2)	3.74	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.77	0.94	
Unstructured Code (Treatment 2)	0.76	0.94	
Hypothesis Test			
p-value	0.011256204509341639		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.72		

Table 4.25: SERPRO - Question 9 - Statistics.

The results of question 9 in SERPRO are presented in Figure 4.19, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.25 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-value was 0.0113, which is below the alpha value of 0.1, with a medium effect size of 0.72, very close to being considered a large effect. This indicates that there is statistical evidence suggesting that Treatment 1 (SOLID) leads

to a better understanding of the source code than Treatment 2 (Unstructured Code).

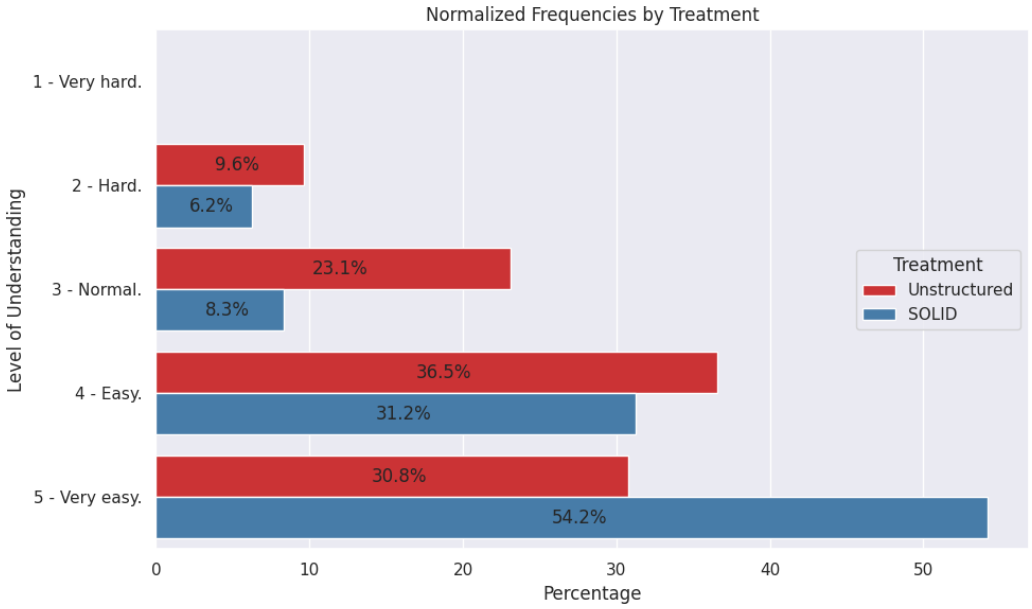

Figure 4.20: OVERALL - Question 9 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.34	5	[5]
Unstructured Code (Treatment 2)	3.89	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.73	0.89	
Unstructured Code (Treatment 2)	0.78	0.97	
Hypothesis Test			
p-value	0.005691739528320074		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.48		

Table 4.26: OVERALL - Question 9 - Statistics.

The results of question 9 in all groups are presented in Figure 4.20, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.26 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-value was 0.0057, which is below the alpha value of 0.1, with an effect size of 0.48, very close to being considered a medium effect. This indicates that there is statistical evidence suggesting that Treatment 1 (SOLID) leads to a better understanding of the source code than Treatment 2 (Unstructured Code).

4.2.2.2 Question 11

It was easy to extend the behavior of the software with the addition of new models. What is your degree of agreement with the statement?

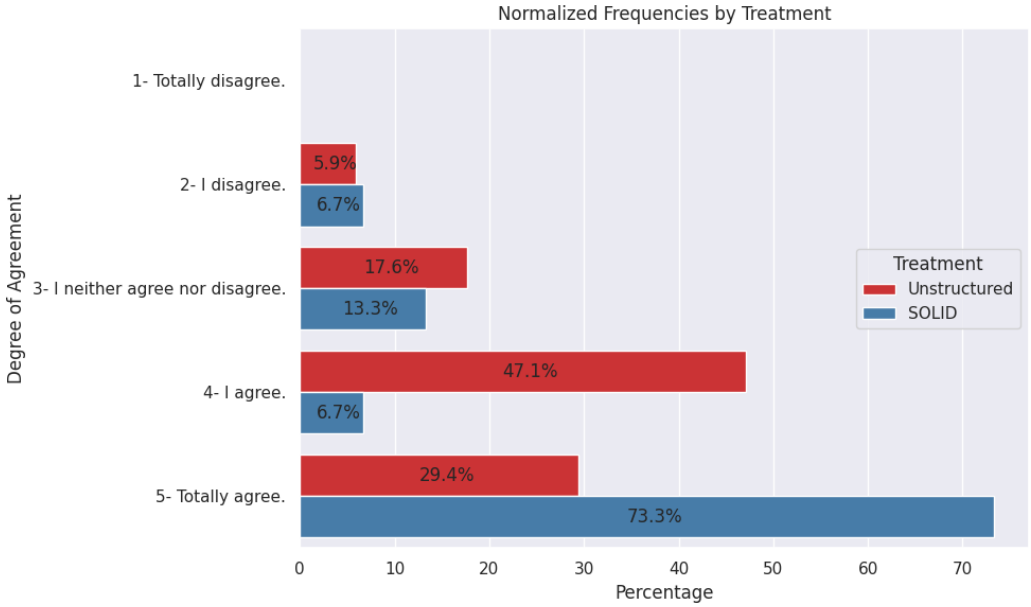

Figure 4.21: BARI - Question 11 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.47	5	[5]
Unstructured Code (Treatment 2)	4.0	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.79	1.0	
Unstructured Code (Treatment 2)	0.59	0.87	
Hypothesis Test			
p-value	0.03261401452924273		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.50		

Table 4.27: BARI - Question 11 - Statistics.

The results of question 11 in BARI are presented in Figure 4.21, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.27 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0326, which is below the alpha value of 0.1, with a medium effect size of 0.50. This indicates that there is statistical evidence to suggest

that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

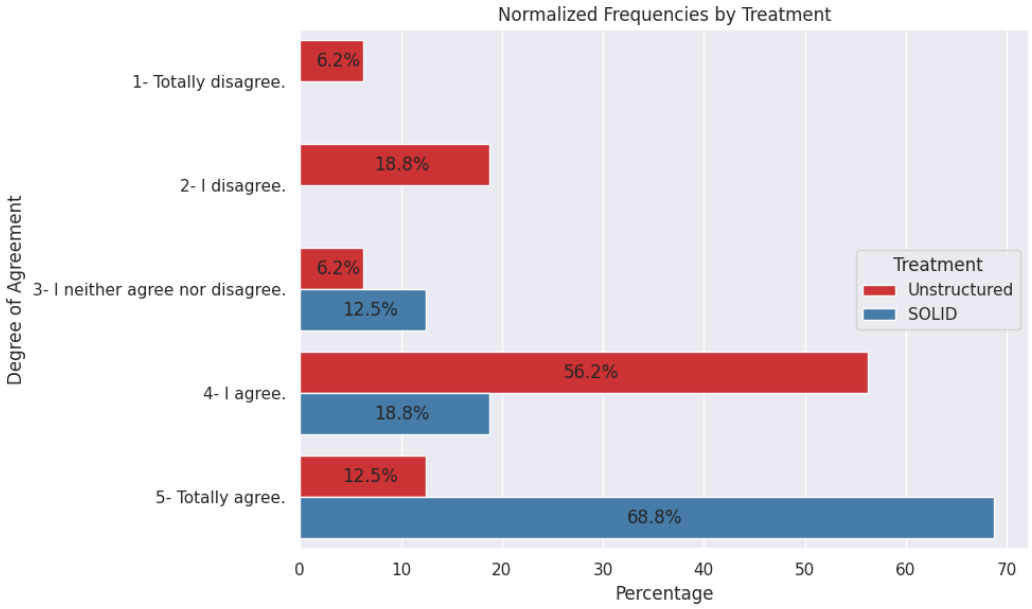

Figure 4.22: PUC - Question 11 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.57	5	[5]
Unstructured Code (Treatment 2)	3.5	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.61	0.73	
Unstructured Code (Treatment 2)	0.94	1.16	
Hypothesis Test			
p-value	0.0014693487108773985		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.10		

Table 4.28: PUC - Question 11 - Statistics.

The results of question 11 in PUC are presented in Figure 4.22, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.28 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0015, which is below the alpha value of 0.1, with a large effect size of 1.10. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

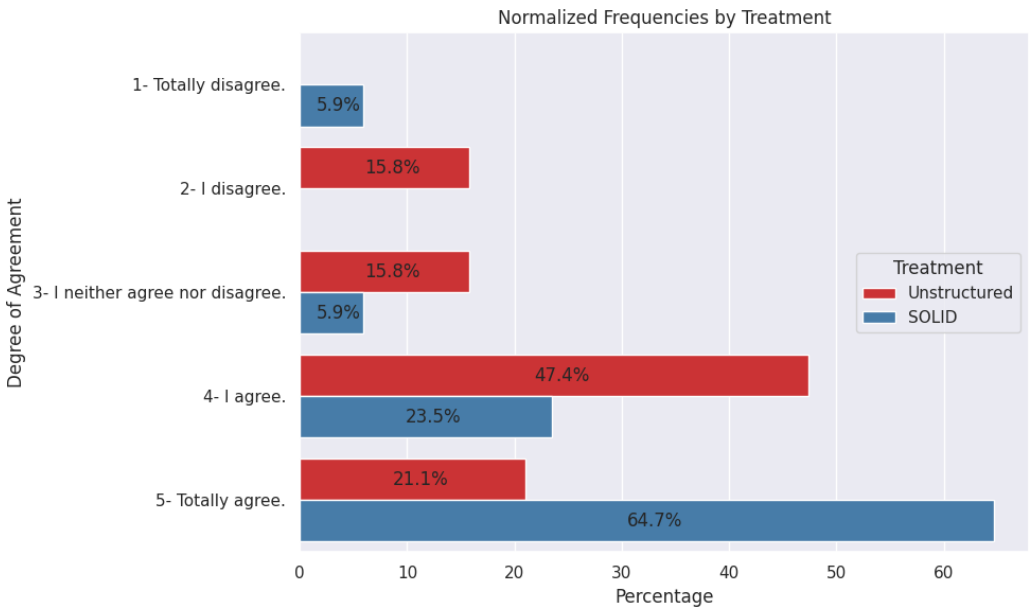

Figure 4.23: SERPRO - Question 11 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.42	5	[5]
Unstructured Code (Treatment 2)	3.74	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.77	1.07	
Unstructured Code (Treatment 2)	0.79	1.0	
Hypothesis Test			
p-value	0.00784193272120681		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.65		

Table 4.29: SERPRO - Question 11 - Statistics.

The results of question 11 in SERPRO are presented in Figure 4.23, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.29 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0078, which is below the alpha value of 0.1, with a medium effect size of 0.65. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

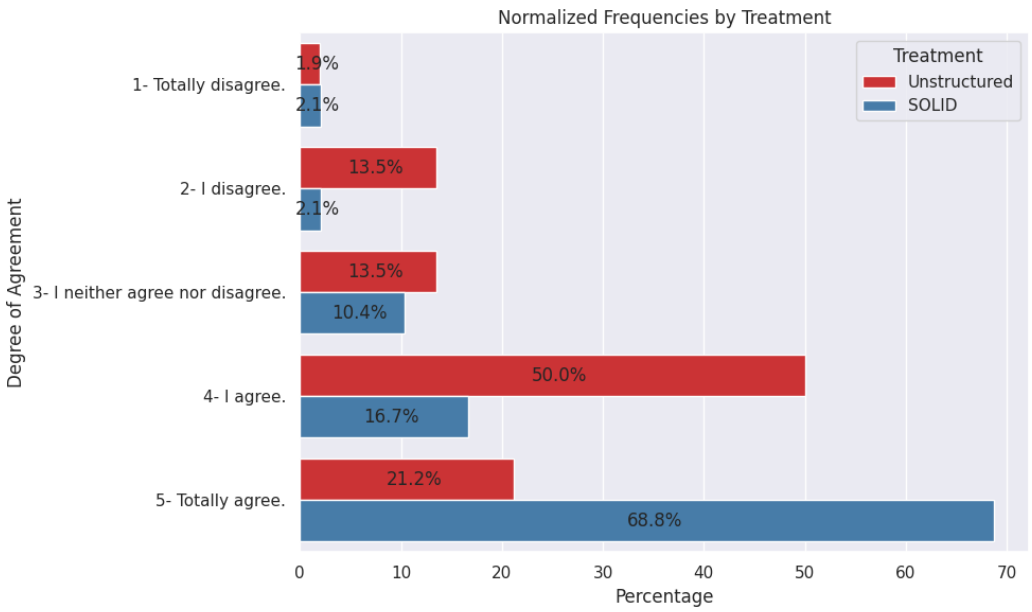

Figure 4.24: OVERALL - Question 11 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.48	5	[5]
Unstructured Code (Treatment 2)	3.75	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.72	0.93	
Unstructured Code (Treatment 2)	0.78	1.01	
Hypothesis Test			
p-value	1.1863728457030575e-05		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.75		

Table 4.30: OVERALL - Question 11 - Statistics.

The results of question 11 in all groups are presented in Figure 4.24, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.30 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with an effect size of 0.75, very close to being considered a large effect. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

4.2.2.3 Question 13

Adding new models did not imply changing pre-existing code. What is your degree of agreement with the statement?

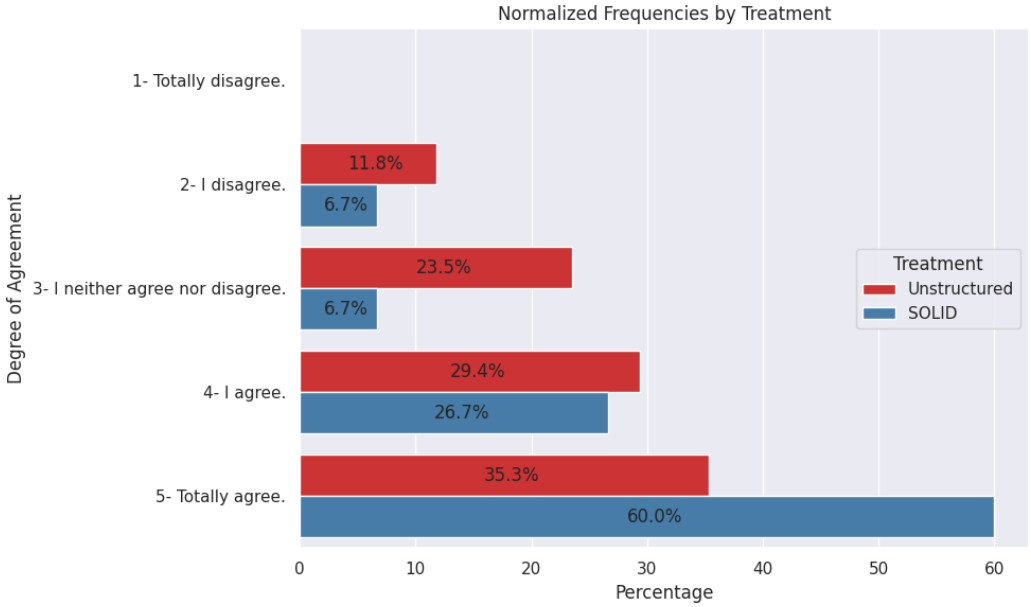

Figure 4.25: BARI - Question 13 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.41	5	[5]
Unstructured Code (Treatment 2)	3.89	4	[5]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.72	0.92	
Unstructured Code (Treatment 2)	0.86	1.06	
Hypothesis Test			
p-value	0.06717759263702512		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.52		

Table 4.31: BARI - Question 13 - Statistics.

The results of question 13 in BARI are presented in Figure 4.25, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.31 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0672, which is below the alpha value of 0.1, with a medium effect size of 0.52. This indicates that there is statistical evidence to suggest

that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

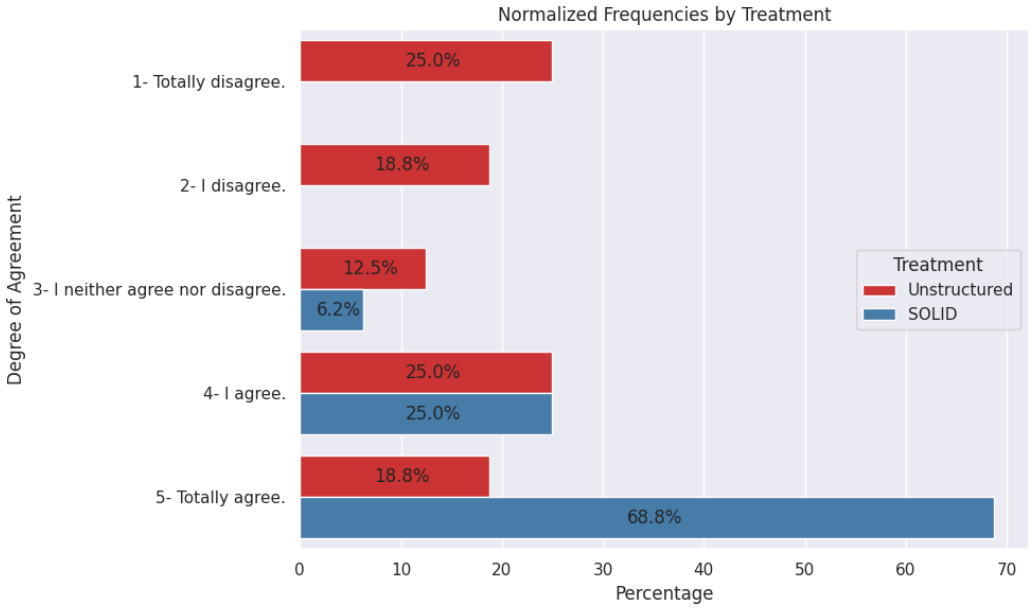

Figure 4.26: PUC - Question 13 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.63	5	[5]
Unstructured Code (Treatment 2)	2.94	3	[1 4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.52	0.62	
Unstructured Code (Treatment 2)	1.33	1.53	
Hypothesis Test			
p-value	0.00048131696999929125		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.44		

Table 4.32: PUC - Question 13 - Statistics.

The results of question 13 in PUC are presented in Figure 4.26, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.32 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0005, which is below the alpha value of 0.1, with a large effect size of 1.44. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

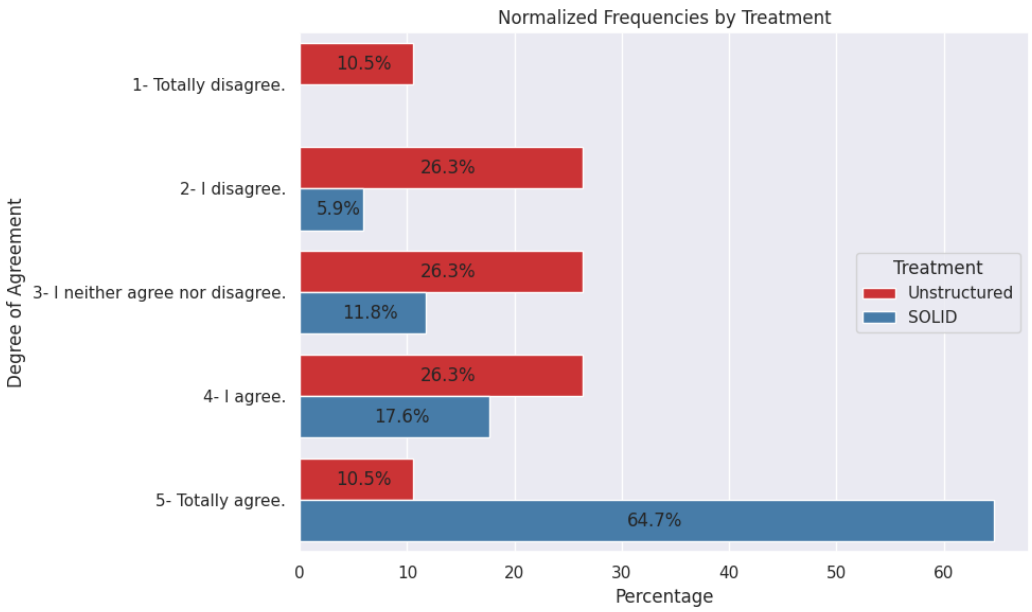

Figure 4.27: SERPRO - Question 13 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.42	5	[5]
Unstructured Code (Treatment 2)	3.0	3	[2 3 4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.77	0.94	
Unstructured Code (Treatment 2)	0.95	1.21	
Hypothesis Test			
p-value	0.00035783185499303055		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.29		

Table 4.33: SERPRO - Question 13 - Statistics.

The results of question 13 in SERPRO are presented in Figure 4.27, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.33 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0004, which is below the alpha value of 0.1, with a large effect size of 1.29. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

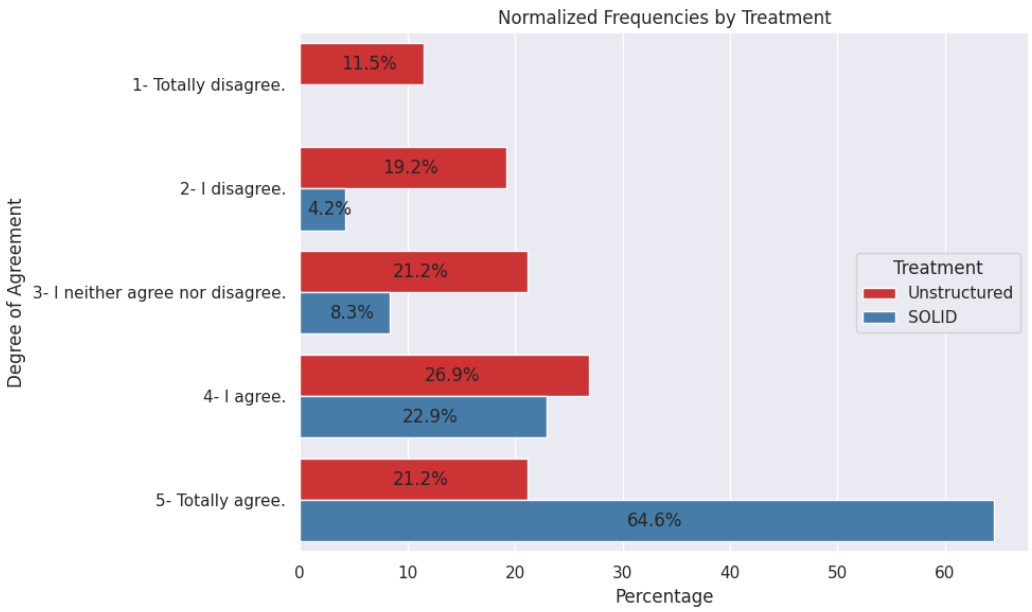

Figure 4.28: OVERALL - Question 13 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.48	5	[5]
Unstructured Code (Treatment 2)	3.27	3	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.68	0.83	
Unstructured Code (Treatment 2)	1.13	1.32	
Hypothesis Test			
p-value	4.909543919850292e-07		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.09		

Table 4.34: OVERALL - Question 13 - Statistics.

The results of question 13 in all groups are presented in Figure 4.28, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.34 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 1.09. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

4.2.3 LSP and DIP - Liskov Substitution Principle and Dependency Inversion Principle

4.2.3.1 Question 15

What is your perception about understanding the source code?

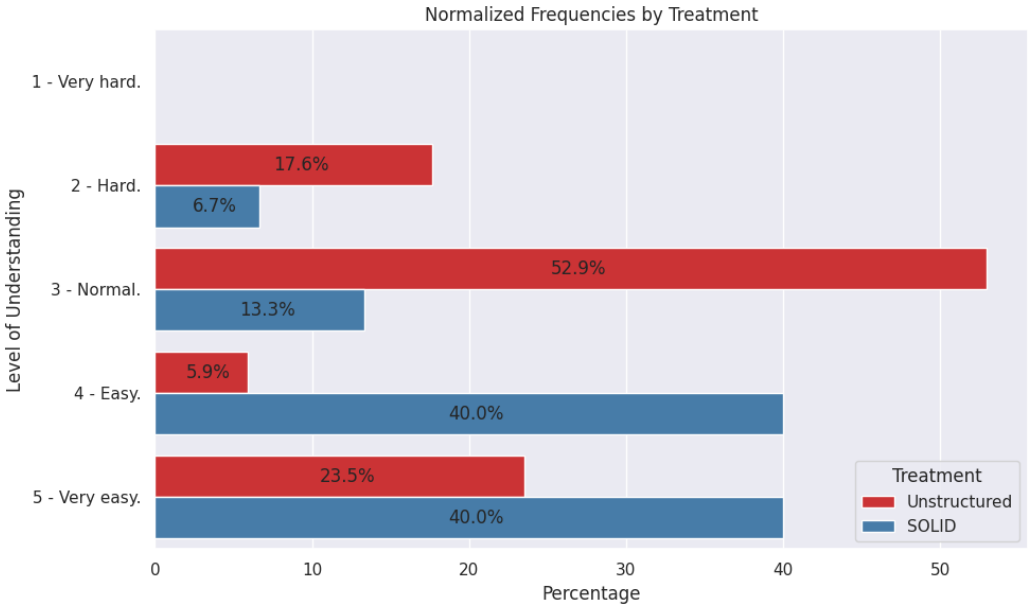

Figure 4.29: BARI - Question 15 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.14	4	[4 5]
Unstructured Code (Treatment 2)	3.36	3	[3]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.7	0.92	
Unstructured Code (Treatment 2)	0.86	1.06	
Hypothesis Test			
p-value	0.017456387154301322		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.78		

Table 4.35: BARI - Question 15 - Statistics.

The results of question 15 in BARI are presented in Figure 4.29, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.35 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-

value was 0.0175, which is below the alpha value of 0.1, with an effect size of 0.78, very close to being considered a large effect. This indicates that there is statistical evidence suggesting that Treatment 1 (SOLID) leads to a better understanding of the source code than Treatment 2 (Unstructured Code).

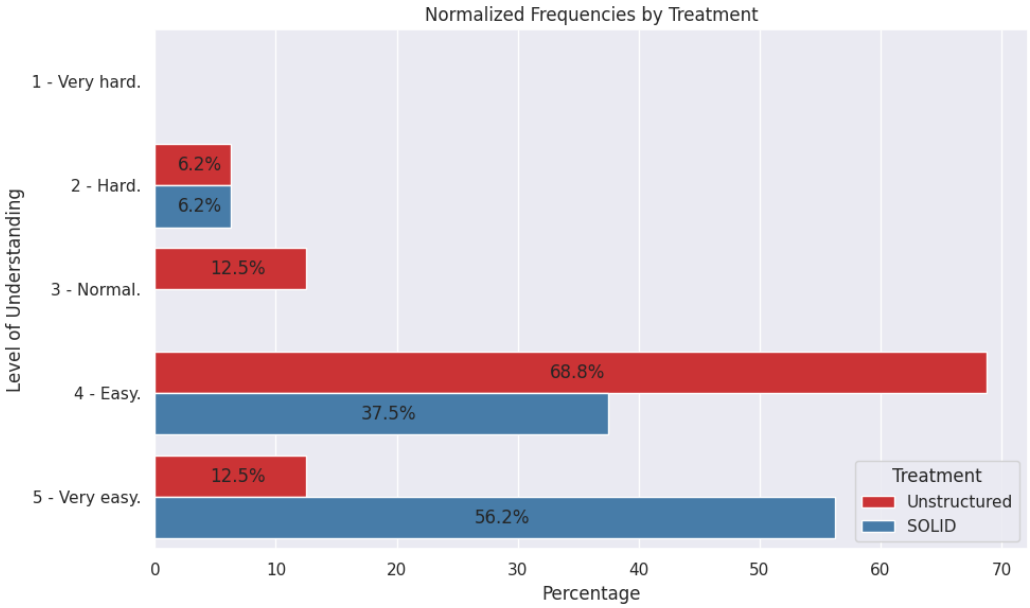

Figure 4.30: PUC - Question 15 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.44	5	[5]
Unstructured Code (Treatment 2)	3.88	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.64	0.82	
Unstructured Code (Treatment 2)	0.46	0.72	
Hypothesis Test			
p-value	0.007565774046882678		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.73		

Table 4.36: PUC - Question 15 - Statistics.

The results of question 15 in PUC are presented in Figure 4.30, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.36 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-value was 0.0076, which is below the alpha value of 0.1, with an effect size of 0.73, very close to being considered a large effect. This indicates that there

is statistical evidence suggesting that Treatment 1 (SOLID) leads to a better understanding of the source code than Treatment 2 (Unstructured Code).

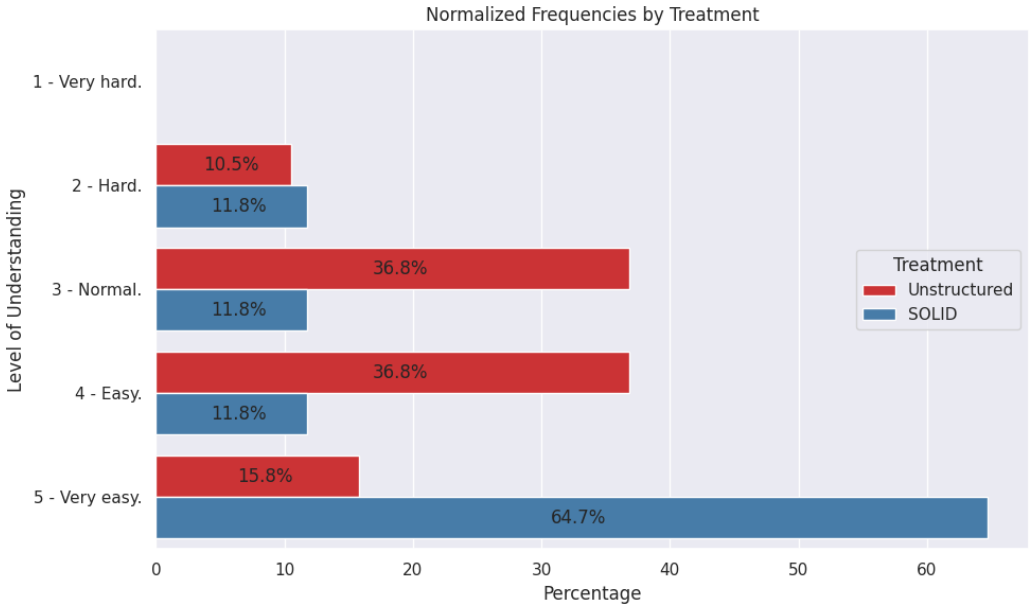

Figure 4.31: SERPRO - Question 15 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.3	5	[5]
Unstructured Code (Treatment 2)	3.58	4	[3 4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.92	1.11	
Unstructured Code (Treatment 2)	0.76	0.91	
Hypothesis Test			
p-value	0.011413403208042936		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.71		

Table 4.37: SERPRO - Question 15 - Statistics.

The results of question 15 in SERPRO are presented in Figure 4.31, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.37 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-value was 0.0114, which is below the alpha value of 0.1, with an effect size of 0.71, very close to being considered a large effect. This indicates that there is statistical evidence suggesting that Treatment 1 (SOLID) leads to a better understanding of the source code than Treatment 2 (Unstructured Code).

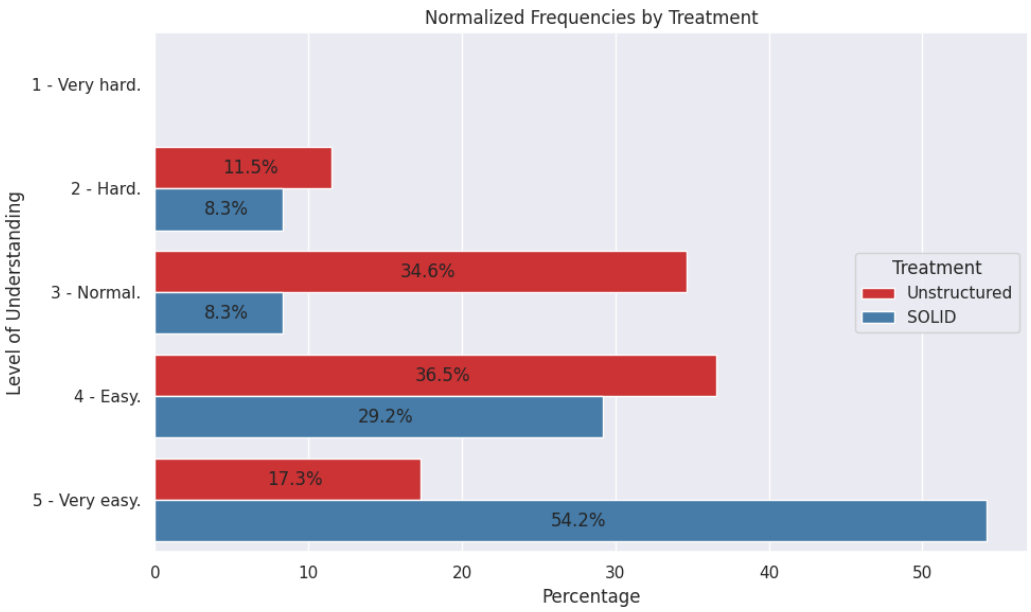

Figure 4.32: OVERALL - Question 15 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.3	5	[5]
Unstructured Code (Treatment 2)	3.6	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.77	0.95	
Unstructured Code (Treatment 2)	0.79	0.92	
Hypothesis Test			
p-value	6.784250167060853e-05		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.74		

Table 4.38: OVERALL - Question 15 - Statistics.

The results of question 15 in all groups are presented in Figure 4.32, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.38 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with an effect size of 0.74, very close to being considered a large effect. This indicates that there is statistical evidence suggesting that Treatment 1 (SOLID) leads to a better understanding of the source code than Treatment 2 (Unstructured Code).

4.2.3.2 Question 17

The models can be replaced without changing the controller. What is your degree of agreement with the statement?

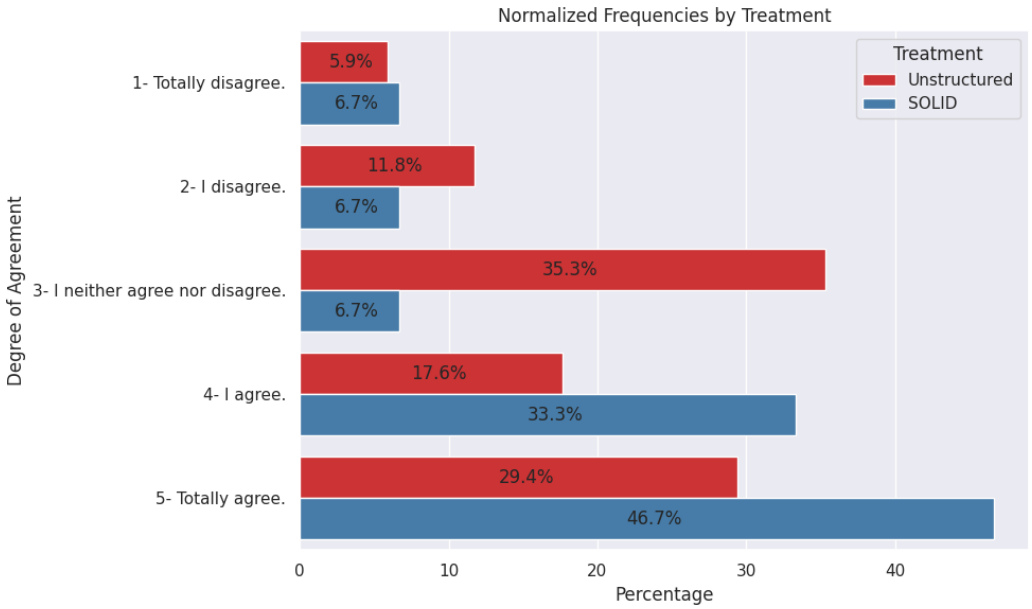

Figure 4.33: BARI - Question 17 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.07	4	[5]
Unstructured Code (Treatment 2)	3.53	3	[3]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.88	1.23	
Unstructured Code (Treatment 2)	1.04	1.24	
Hypothesis Test			
p-value	0.08730525788049082		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.43		

Table 4.39: BARI - Question 17 - Statistics.

The results of question 17 in BARI are presented in Figure 4.33, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.39 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0873, which is below the alpha value of 0.1, with an effect size of 0.43, very close to being considered a medium effect. This indicates that there

is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

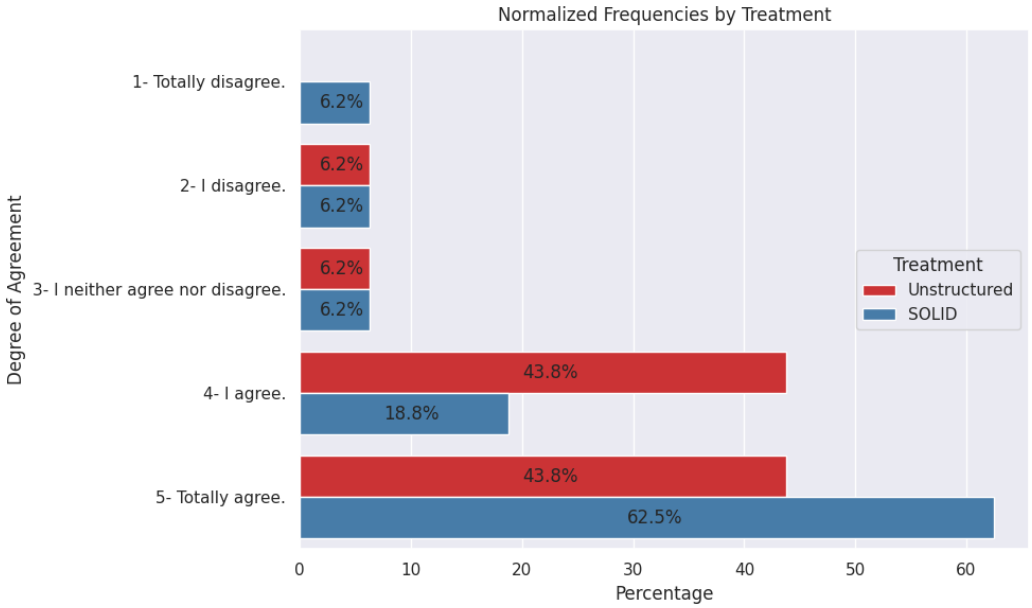

Figure 4.34: PUC - Question 17 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.25	5	[5]
Unstructured Code (Treatment 2)	4.25	4	[4 5]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.94	1.24	
Unstructured Code (Treatment 2)	0.66	0.86	
Hypothesis Test			
p-value	0.26619201215880495		
Interpretation	Does not reject the null hypothesis: There is no evidence that the median of 'SOLID' is greater than the median of 'Unstructured'.		

Table 4.40: PUC - Question 17 - Statistics.

The results of question 17 in PUC are presented in Figure 4.34, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.40 presents measures of central tendency, measures of dispersion, and results of the hypothesis test.

Although the result of the hypothesis test did not show a statistically significant difference between the Treatment 1 (SOLID) and Treatment 2 (Unstructured Code) groups regarding the statement "Models can be replaced without needing to change the controller," we can still argue that Treatment 1 (SOLID) promotes a better understanding of this capacity, based on the answers to question 17 and the justifications presented.

In the group that received Treatment 1 (SOLID), a significantly larger portion of participants (62.5%) totally agreed that models can be replaced without changing the controller. This majority of "5 - Totally Agree" responses demonstrate a clear understanding and confidence in the code's ability to support model replacement without controller interventions. Their justifications mentioned that the controller was designed to be generic, accepting models as parameters. For example, one participant mentioned that "the models are parameterized in the controller", which means that the controller is independent of the specific models and can work with different models without significant changes.

On the other hand, in the group that received Treatment 2 (Unstructured Code), although the majority (43.8%) also agreed with the statement, their agreement was less emphatic. This difference in responses suggests a less solid and confident understanding of the code's ability to support model replacement without controller changes. Their justifications were more varied. Some participants mentioned that the function that processes models is generic and does not need modifications for model switching, while others noted that flexibility depends on the evaluation metric and the behavior of the models. As an example, one participant responded, "Since what matters is the list of models, if the model requires the same metric and is of the regressor type, it would not be necessary. However, if the model to be tested has a different metric or behavior, it would be necessary to modify the function or create a new one". This suggests that, although there is agreement, participants in this group expressed a somewhat less clear and direct understanding of the code's flexibility.

Therefore, based on the high percentage of participants who totally agree with the statement, combined with justifications for their responses, it suggests that participants in Treatment 1 (SOLID) have a more solid and confident degree of agreement with the statement presented in question 17, compared to participants in Treatment 2 (Unstructured Code), despite the lack of statistically significant difference.

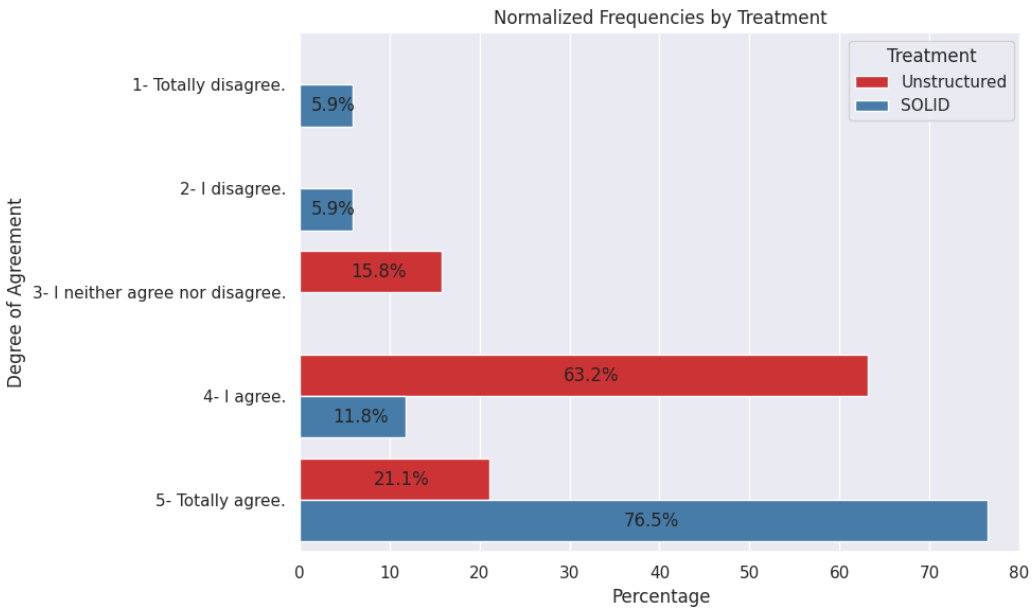

Figure 4.35: SERPRO - Question 17 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.48	5	[5]
Unstructured Code (Treatment 2)	4.06	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.81	1.18	
Unstructured Code (Treatment 2)	0.4	0.63	
Hypothesis Test			
p-value	0.003807243519790059		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.45		

Table 4.41: SERPRO - Question 17 - Statistics.

The results of question 17 in SERPRO are presented in Figure 4.35, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.41 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0038, which is below the alpha value of 0.1, with an effect size of 0.45, very close to being considered a medium effect. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

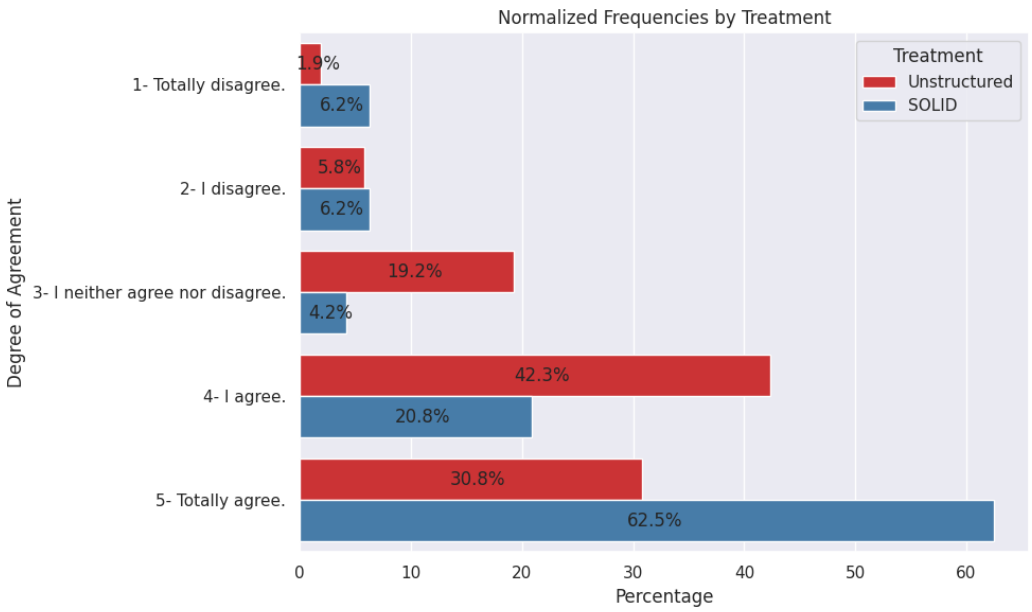

Figure 4.36: OVERALL - Question 17 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.28	5	[5]
Unstructured Code (Treatment 2)	3.95	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.92	1.2	
Unstructured Code (Treatment 2)	0.7	0.96	
Hypothesis Test			
p-value	0.004928369662340698		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.30		

Table 4.42: OVERALL - Question 17 - Statistics.

The results of question 17 in all groups are presented in Figure 4.36, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.42 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0049, which is below the alpha value of 0.1, with an effect size of 0.30. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

4.2.3.3 Question 19

Changes in a model implementation do not imply changes in the controller. What is your degree of agreement with the statement?

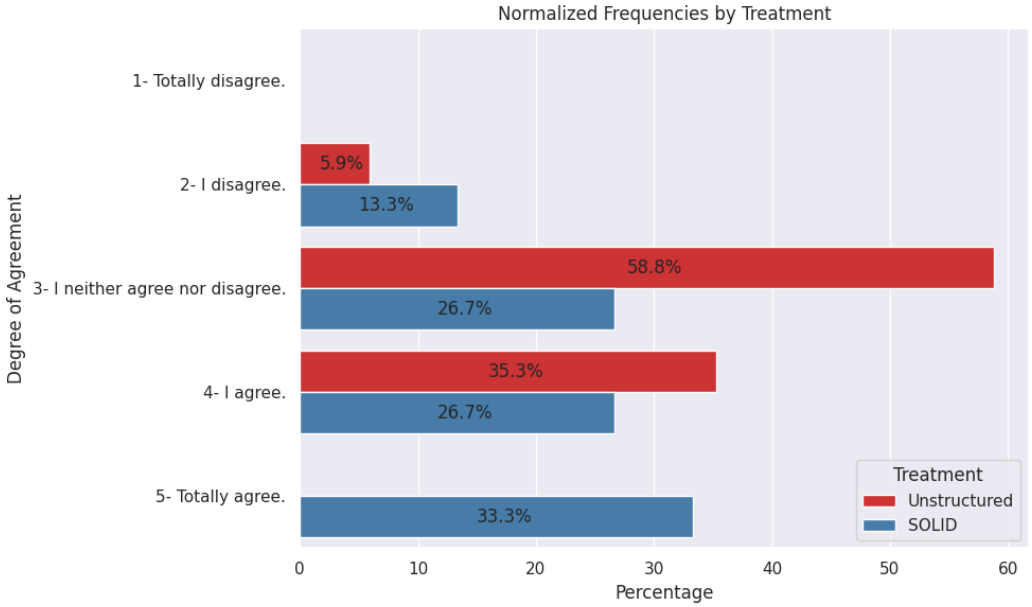

Figure 4.37: BARI - Question 19 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	3.8	4	[5]
Unstructured Code (Treatment 2)	3.3	3	[3]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.91	1.09	
Unstructured Code (Treatment 2)	0.5	0.59	
Hypothesis Test			
p-value	0.06325706533595894		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.59		

Table 4.43: BARI - Question 19 - Statistics.

The results of question 19 in BARI are presented in Figure 4.37, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.43 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0633, which is below the alpha value of 0.1, with a medium effect size of 0.59. This indicates that there is statistical evidence to suggest

that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

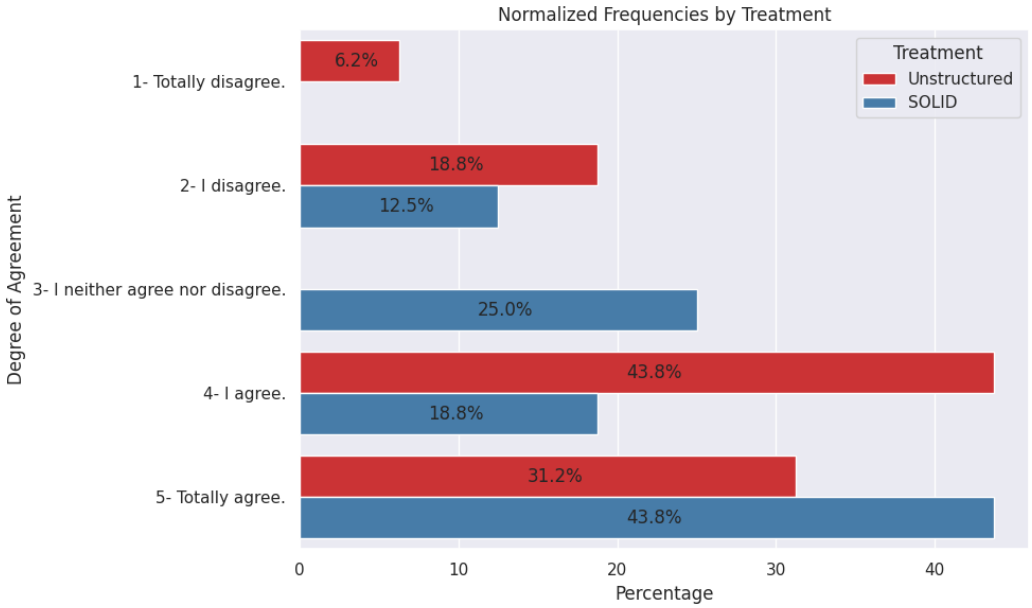

Figure 4.38: PUC - Question 19 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	3.94	4	[5]
Unstructured Code (Treatment 2)	3.75	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.96	1.13	
Unstructured Code (Treatment 2)	1.0	1.3	
Hypothesis Test			
p-value	0.35384736353087953		
Interpretation	Does not reject the null hypothesis: There is no evidence that the median of 'SOLID' is greater than the median of 'Unstructured'.		

Table 4.44: PUC - Question 19 - Statistics.

The results of question 19 in PUC are presented in Figure 4.38, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.44 presents measures of central tendency, measures of dispersion, and results of the hypothesis test.

Although the results of the hypothesis test did not show a statistically significant difference between Treatment 1 (SOLID) and Treatment 2 (Unstructured Code) regarding the statement "Changes in a model implementation do not imply changes in the controller," we can still argue that Treatment 1 (SOLID) promotes a better understanding of this capability based on the responses to question 19 and the provided justifications.

In the group that received Treatment 1 (SOLID), a significantly larger portion of participants, 43.8%, totally agreed with the statement that changes in a model implementation do not result in changes in the controller. This majority of "5- Totally agree." responses demonstrates a clear understanding and confidence in the code's ability to support model changes without requiring modifications to the controller. Their justifications mentioned that models are independent of controllers and that changes in model implementations would not significantly affect the controller's functionality. For example, one participant mentioned, "There is no need to alter the controller because each model still follows the abstraction, that of the model superclass. It always constructs the model without altering anything from the controller's perspective".

On the other hand, in the group that received Treatment 2 (Unstructured Code), although 31.2% also totally agreed with the proposed statement, their agreement was less emphatic. This difference in responses suggests a slightly less solid and confident understanding of the code's ability to support model changes without controller alterations. Some participants mentioned that the function processing of the models is specific and may require changes for different models or model changes. One participant mentioned, "As long as the Model has the same methods (fit, predict, etc.), the internal implementation can be changed." Another participant mentioned, "It changes because methods accessed by the model may have been affected or have new names or parameters."

Therefore, based on the percentage of participants who totally agree with the statement, combined with justifications for their responses, it suggests that participants in Treatment 1 (SOLID) have a more solid and confident degree of agreement with the statement presented in question 19, compared to participants in Treatment 2 (Unstructured Code), despite the lack of statistical significance.

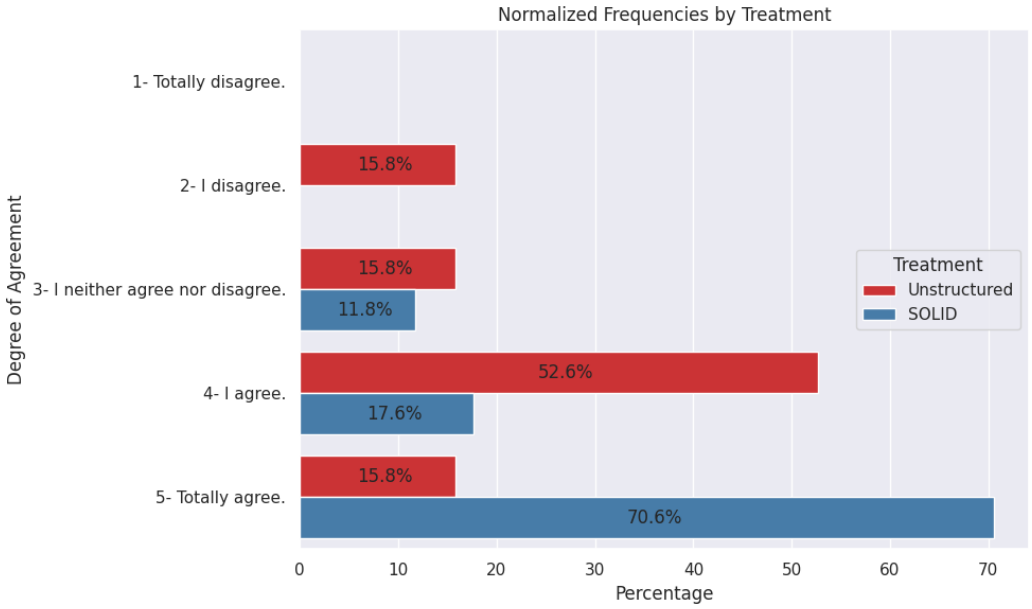

Figure 4.39: SERPRO - Question 19 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.59	5	[5]
Unstructured Code (Treatment 2)	3.69	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.59	0.72	
Unstructured Code (Treatment 2)	0.75	0.95	
Hypothesis Test			
p-value	0.001168880786491067		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.07		

Table 4.45: SERPRO - Question 19 - Statistics.

The results of question 19 in SERPRO are presented in Figure 4.39, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.45 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0012, which is below the alpha value of 0.1, with a large effect size of 1.07. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

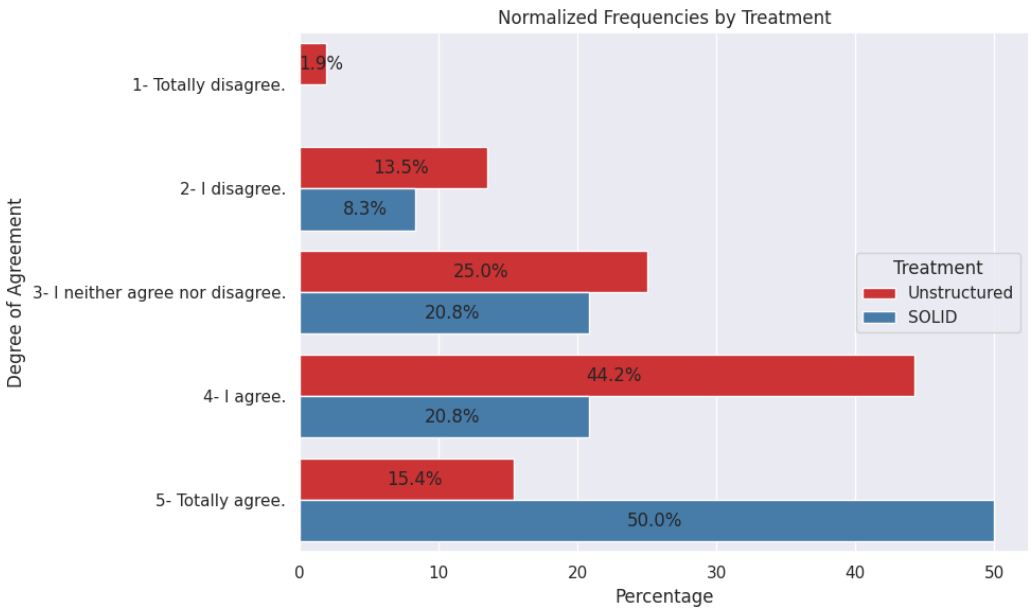

Figure 4.40: OVERALL - Question 19 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.13	4	[5]
Unstructured Code (Treatment 2)	3.58	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.88	1.03	
Unstructured Code (Treatment 2)	0.82	0.98	
Hypothesis Test			
p-value	0.0024062638472889053		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.54		

Table 4.46: OVERALL - Question 19 - Statistics.

The results of question 19 in all groups are presented in Figure 4.40, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.46 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0024, which is below the alpha value of 0.1, with a medium effect size of 0.54. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

4.2.3.4 Question 21

The controller has loose coupling with model implementations. What is your degree of agreement with the statement?

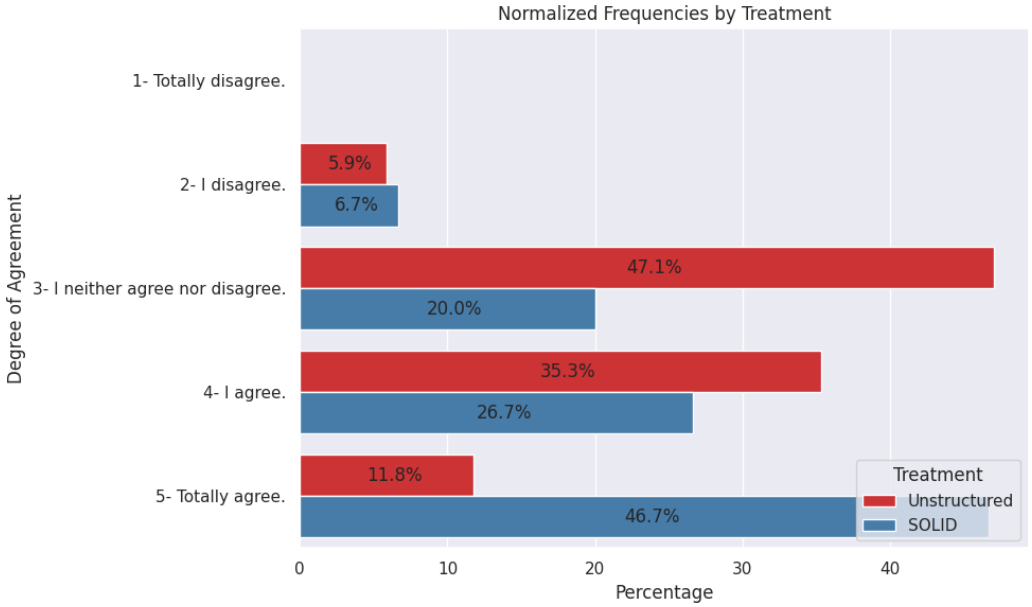

Figure 4.41: BARI - Question 21 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.14	4	[5]
Unstructured Code (Treatment 2)	3.53	3	[3]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.81	1.0	
Unstructured Code (Treatment 2)	0.68	0.8	
Hypothesis Test			
p-value	0.029838012543261735		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.67		

Table 4.47: BARI - Question 21 - Statistics.

The results of question 21 in BARI are presented in Figure 4.41, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.47 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0298, which is below the alpha value of 0.1, with a medium effect size of 0.67. This indicates that there is statistical evidence to suggest

that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

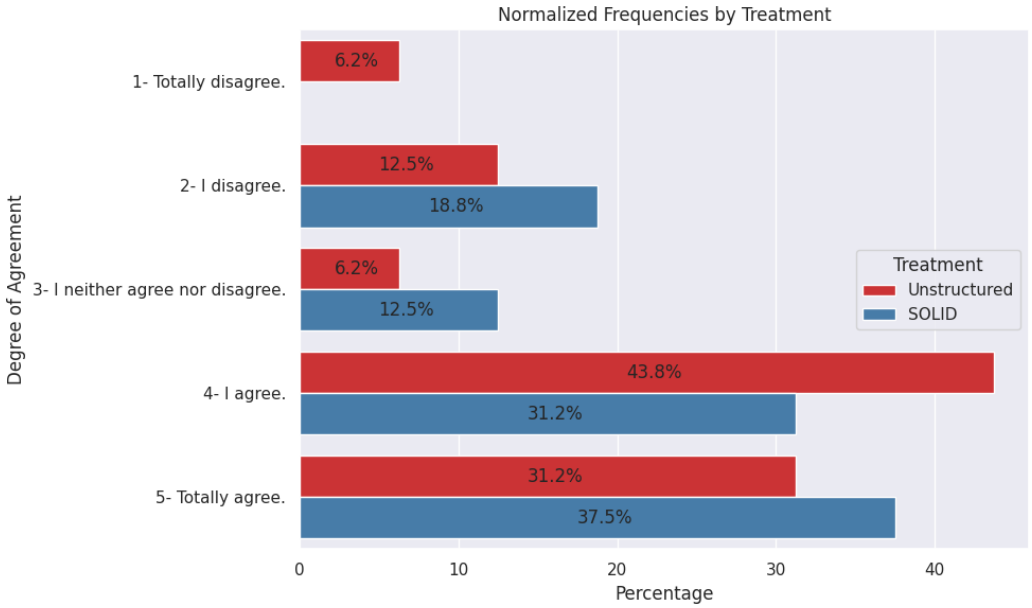

Figure 4.42: PUC - Question 21 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	3.88	4	[5]
Unstructured Code (Treatment 2)	3.82	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.93	1.15	
Unstructured Code (Treatment 2)	0.91	1.23	
Hypothesis Test			
p-value	0.4526400721768805		
Interpretation	Does not reject the null hypothesis: There is no evidence that the median of 'SOLID' is greater than the median of 'Unstructured'.		

Table 4.48: PUC - Question 21 - Statistics.

The results of question 21 in PUC are presented in Figure 4.42, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.48 presents measures of central tendency, measures of dispersion, and results of the hypothesis test.

The hypothesis test result does not reject the null hypothesis. There is no statistical evidence that participants in Treatment 1 (SOLID) agree more with the statement about low coupling than participants in Treatment 2 (Unstructured Code).

It is observed that the responses for both treatments vary significantly. In both treatments, there are participants who totally agree, agree, disagree,

and neither agree nor disagree with the statement about low coupling. This indicates that the perception of the degree of coupling is not consistent among participants.

The justifications provided by participants for their responses are also diverse. Some participants in Treatment 1 (SOLID) claim that the controller has low coupling due to its object-oriented approach and dependence only on the model's interface, while others argue that there is still some dependence on specific model implementations. One participant commented, "There is dependence only on the Model interface and not on Model implementations." Another participant disagreed with their response but commented, "The controller depends on a model that implements the expected interface (public methods)."

In Treatment 2 (Unstructured Code), the justifications also vary, with some participants emphasizing the independence of the function and others pointing out potential dependencies. One participant commented, "It has low coupling because it is independent of the models." Another participant commented, "A change in the implementation of a model can lead to a change in the processes_models function."

There is no clear consensus among participants in both treatments regarding the degree of coupling. Some totally agree that there is low coupling, while others disagree or do not have a strong opinion on the matter. This suggests that the perception of coupling can be subjective and dependent on individual interpretation.

In summary, based on the responses and justifications of participants, along with the result of the hypothesis test, it is not possible to conclusively state that Treatment 1 (SOLID) leads to a higher degree of agreement with the statement about low coupling compared to Treatment 2 (Unstructured Code).

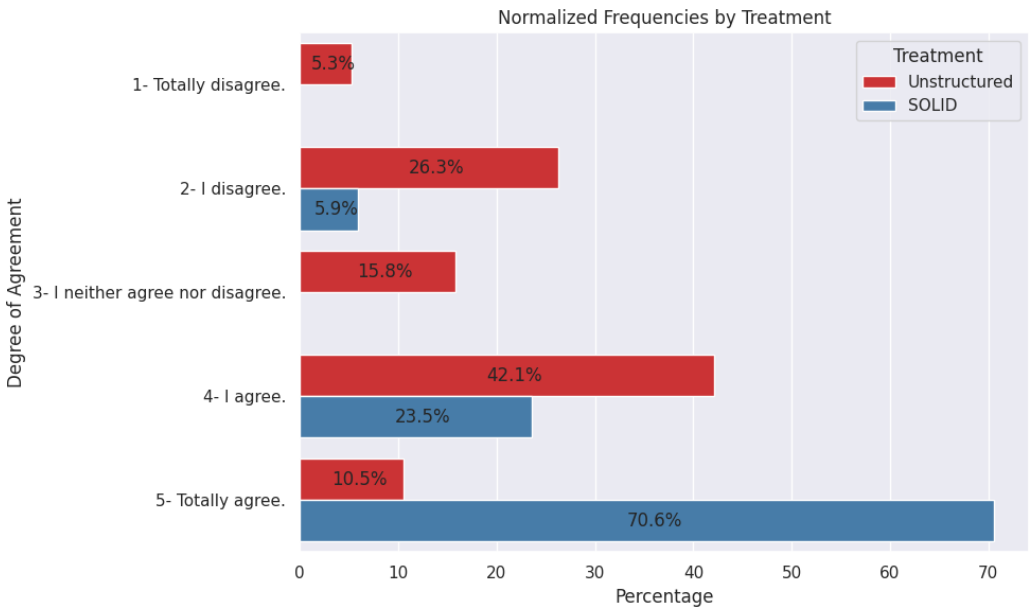

Figure 4.43: SERPRO - Question 21 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.59	5	[5]
Unstructured Code (Treatment 2)	3.27	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.59	0.8	
Unstructured Code (Treatment 2)	0.99	1.15	
Hypothesis Test			
p-value	0.0001269668037778324		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.32		

Table 4.49: SERPRO - Question 21 - Statistics.

The results of question 21 in SERPRO are presented in Figure 4.43, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.49 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 1.32. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

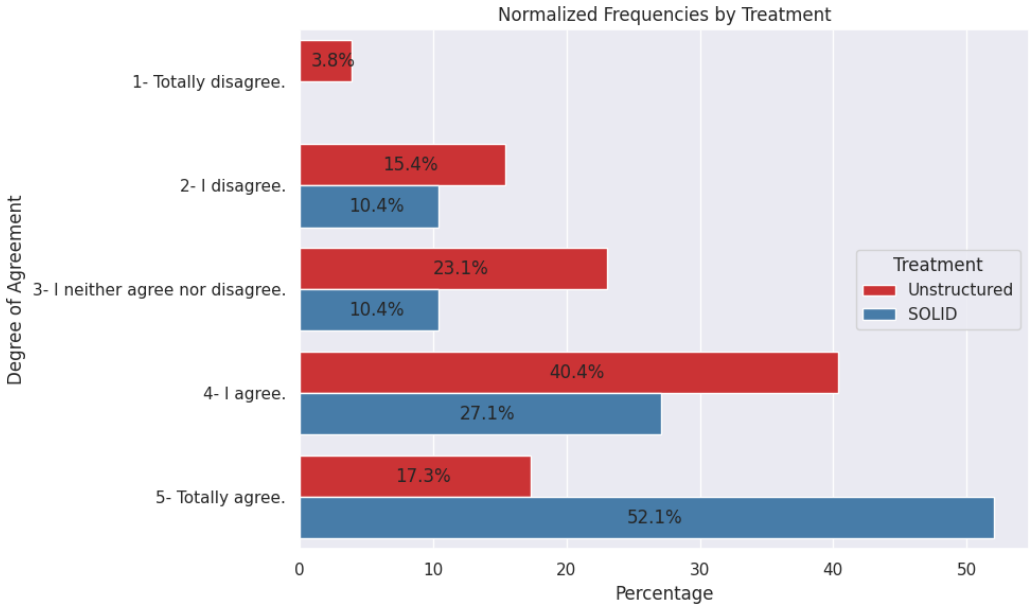

Figure 4.44: OVERALL - Question 21 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.21	5	[5]
Unstructured Code (Treatment 2)	3.52	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.83	1.01	
Unstructured Code (Treatment 2)	0.91	1.08	
Hypothesis Test			
p-value	0.0003345198533971498		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.65		

Table 4.50: OVERALL - Question 21 - Statistics.

The results of question 21 in all groups are presented in Figure 4.44, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.50 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0003, which is below the alpha value of 0.1, with a medium effect size of 0.65. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

4.2.4 ISP - Interface Segregation Principle

4.2.4.1 Question 23

What is your perception about understanding the source code?

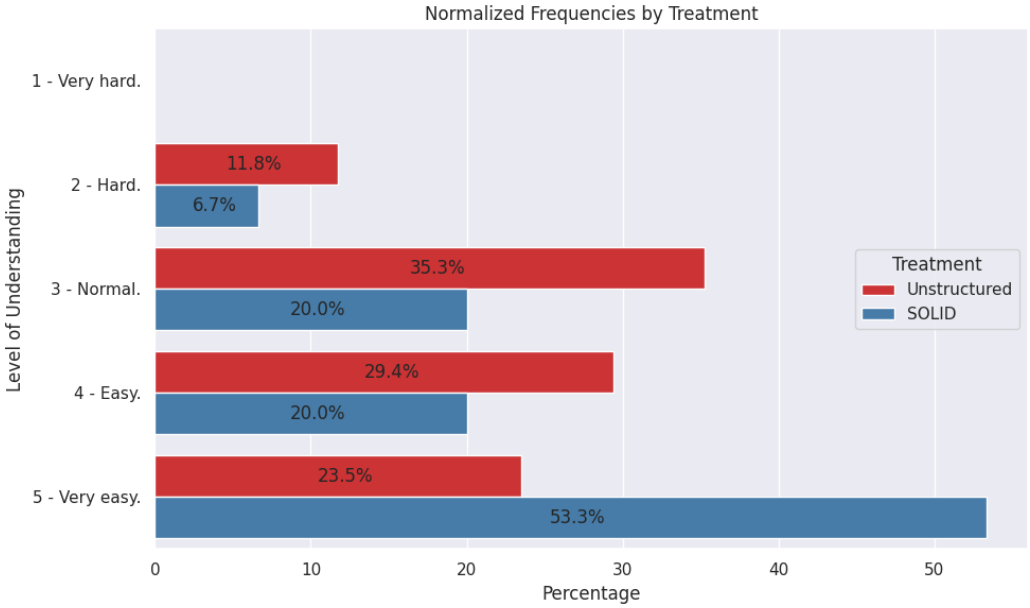

Figure 4.45: BARI - Question 23 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.2	5	[5]
Unstructured Code (Treatment 2)	3.65	4	[3]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.86	1.02	
Unstructured Code (Treatment 2)	0.85	1.0	
Hypothesis Test			
p-value	0.05889495285502934		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.55		

Table 4.51: BARI - Question 23 - Statistics.

The results of question 23 in BARI are presented in Figure 4.45, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.51 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-value was 0.0589, which is below the alpha value of 0.1, with a medium effect size of 0.55. This indicates that there is statistical evidence suggesting that Treatment 1 (SOLID) leads to a better understanding of the source code than Treatment 2 (Unstructured Code).

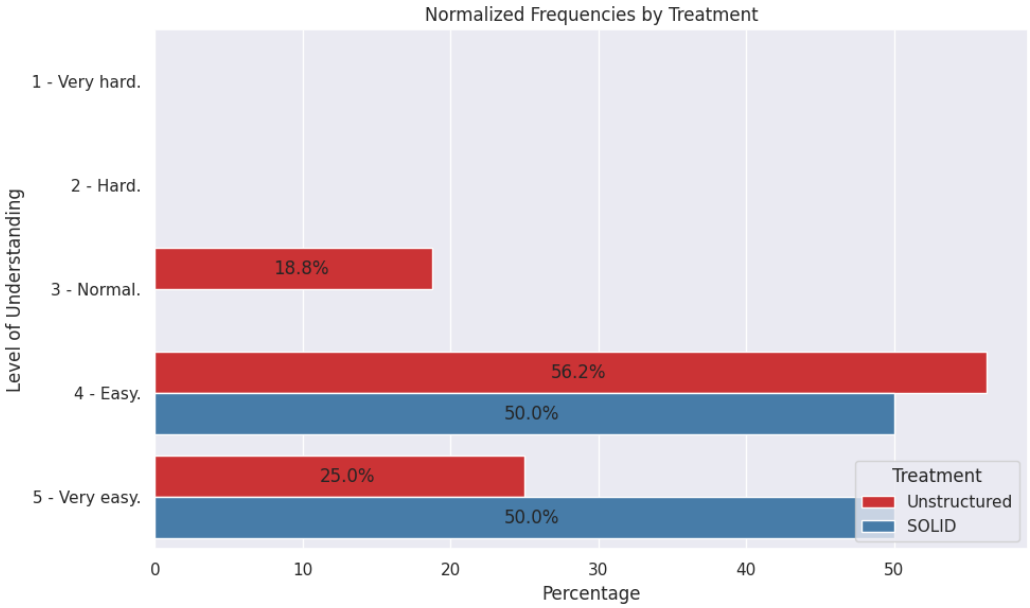

Figure 4.46: PUC - Question 23 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.5	4	[4 5]
Unstructured Code (Treatment 2)	4.07	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.5	0.52	
Unstructured Code (Treatment 2)	0.47	0.69	
Hypothesis Test			
p-value	0.033171700123354395		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.72		

Table 4.52: PUC - Question 23 - Statistics.

The results of question 23 in PUC are presented in Figure 4.46, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.52 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-value was 0.0332, which is below the alpha value of 0.1, with an effect size of 0.72, very close to being considered a large effect. This indicates that there is statistical evidence suggesting that Treatment 1 (SOLID) leads to a better understanding of the source code than Treatment 2 (Unstructured Code).

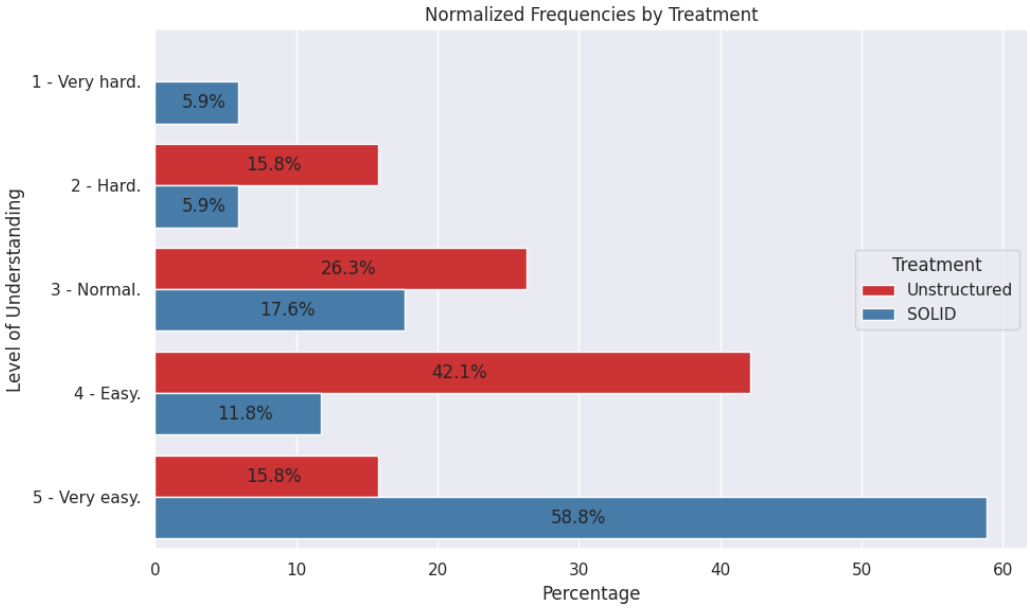

Figure 4.47: SERPRO - Question 23 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.12	5	[5]
Unstructured Code (Treatment 2)	3.58	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	1.04	1.27	
Unstructured Code (Treatment 2)	0.81	0.97	
Hypothesis Test			
p-value	0.034594182722377693		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.48		

Table 4.53: SERPRO - Question 23 - Statistics.

The results of question 23 in SERPRO are presented in Figure 4.47, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.53 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-value was 0.0346, which is below the alpha value of 0.1, with an effect size of 0.48, very close to being considered a medium effect. This indicates that there is statistical evidence suggesting that Treatment 1 (SOLID) leads to a better understanding of the source code than Treatment 2 (Unstructured Code).

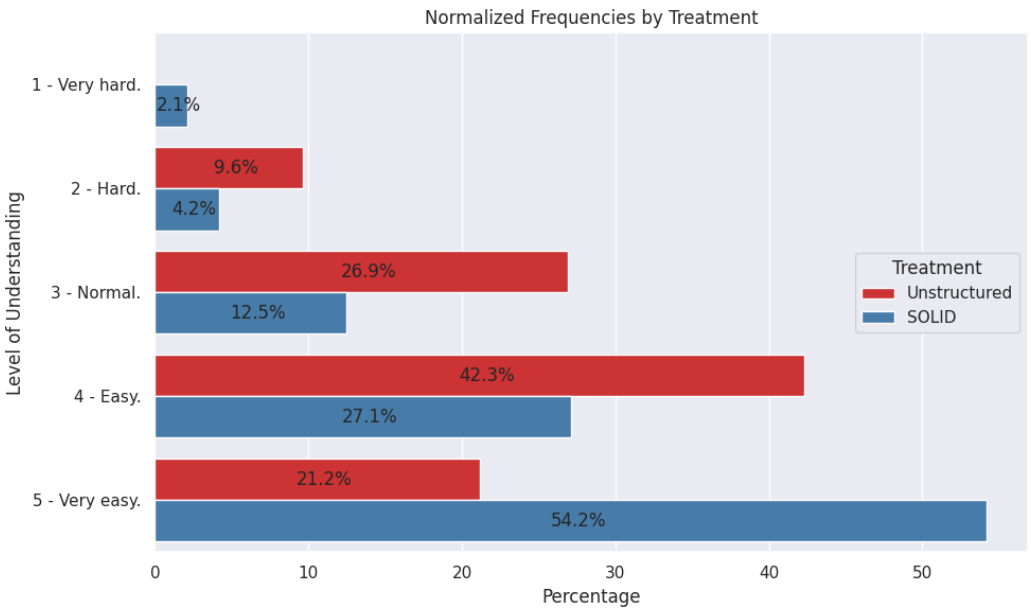

Figure 4.48: OVERALL - Question 23 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.28	5	[5]
Unstructured Code (Treatment 2)	3.75	4	[4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.79	0.99	
Unstructured Code (Treatment 2)	0.75	0.91	
Hypothesis Test			
p-value	0.000928821630030991		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.55		

Table 4.54: OVERALL - Question 23 - Statistics.

The results of question 23 in all groups are presented in Figure 4.48, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.54 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the perception of code understanding between the two groups. The calculated p-value was 0.0009, which is below the alpha value of 0.1, with a medium effect size of 0.55. This indicates that there is statistical evidence suggesting that Treatment 1 (SOLID) leads to a better understanding of the source code than Treatment 2 (Unstructured Code).

4.2.4.2 Question 25

The operations in the evaluator implementations are properly segregated for evaluation of classification and regression algorithms. What is your degree of agreement with the statement?

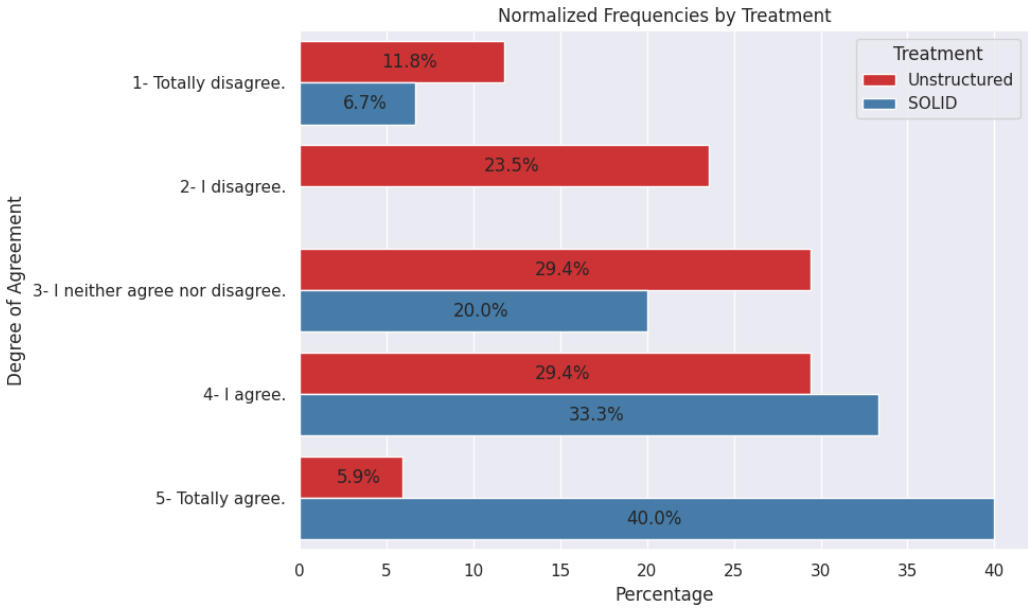

Figure 4.49: BARI - Question 25 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.0	4	[5]
Unstructured Code (Treatment 2)	2.95	3	[3 4]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.8	1.14	
Unstructured Code (Treatment 2)	0.9	1.15	
Hypothesis Test			
p-value	0.0057089239759575555		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	0.92		

Table 4.55: BARI - Question 25 - Statistics.

The results of question 25 in BARI are presented in Figure 4.49, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.55 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0057, which is below the alpha value of 0.1, with a large effect size of 0.92. This indicates that there is statistical evidence to suggest that

Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

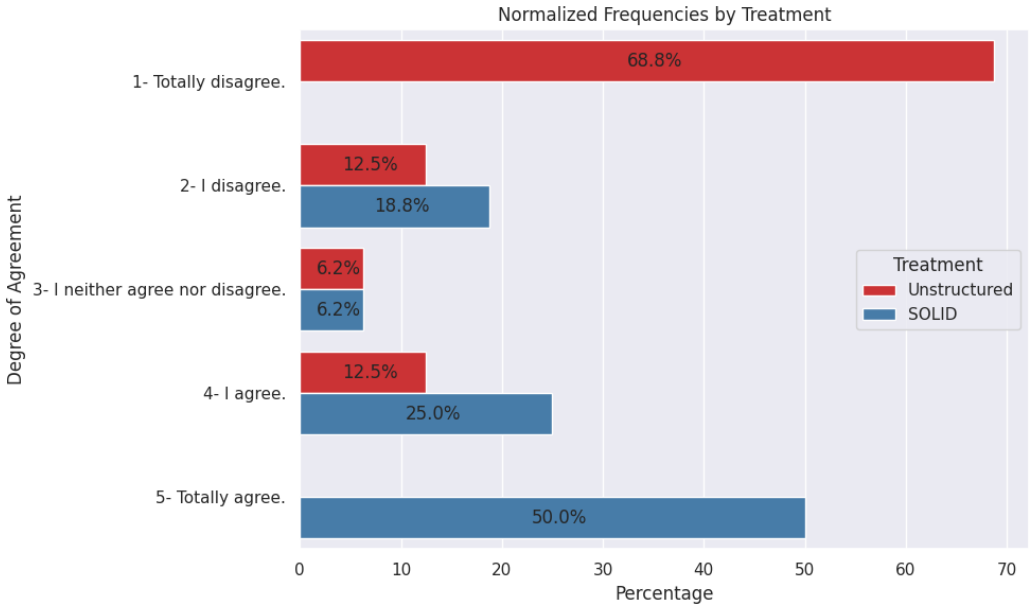

Figure 4.50: PUC - Question 25 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.07	4	[5]
Unstructured Code (Treatment 2)	1.63	1	[1]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.94	1.19	
Unstructured Code (Treatment 2)	0.86	1.09	
Hypothesis Test			
p-value	1.0654155165429655e-05		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	2.14		

Table 4.56: PUC - Question 25 - Statistics.

The results of question 25 in PUC are presented in Figure 4.50, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.56 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 2.14. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

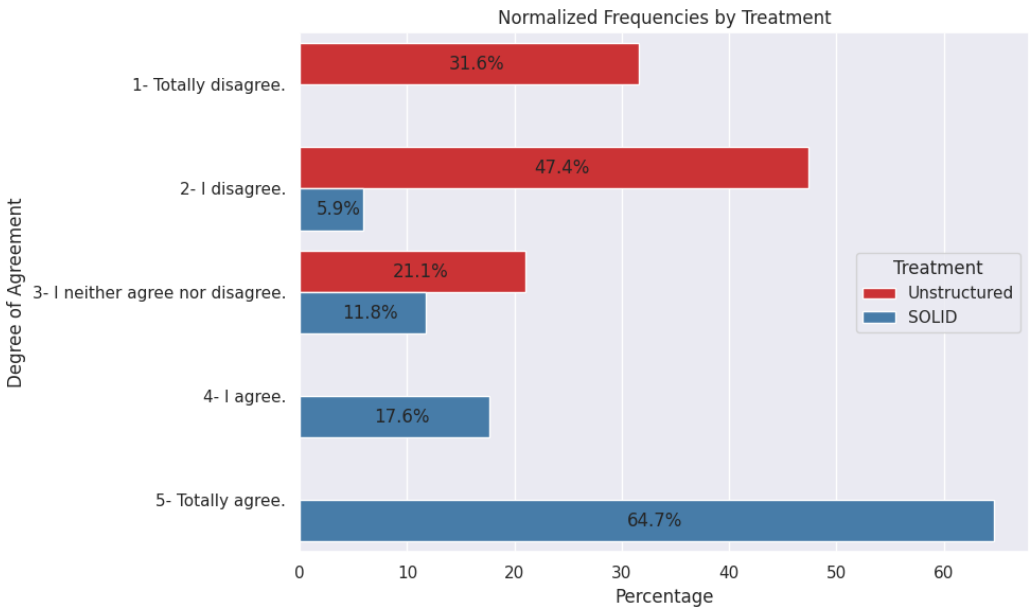

Figure 4.51: SERPRO - Question 25 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.42	5	[5]
Unstructured Code (Treatment 2)	1.9	2	[2]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.77	0.94	
Unstructured Code (Treatment 2)	0.57	0.74	
Hypothesis Test			
p-value	6.140022430043098e-07		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	3		

Table 4.57: SERPRO - Question 25 - Statistics.

The results of question 25 in SERPRO are presented in Figure 4.51, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.57 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 3. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

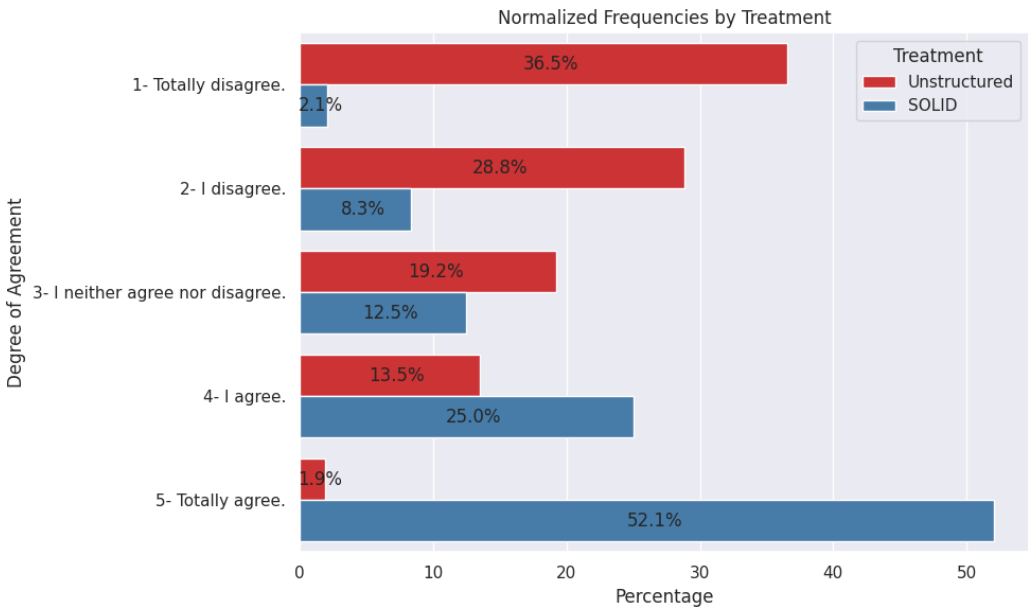

Figure 4.52: OVERALL - Question 25 - Frequencies normalized by treatment.

Measures of Central Tendency			
Treatment	**Mean**	**Median**	**Mode**
SOLID (Treatment 1)	4.17	5	[5]
Unstructured Code (Treatment 2)	2.16	2	[1]
Measures of Dispersion			
Treatment	**Mean Absolute Deviation**	**Standard Deviation**	
SOLID (Treatment 1)	0.87	1.08	
Unstructured Code (Treatment 2)	0.94	1.13	
Hypothesis Test			
p-value	6.852381839511321e-12		
Interpretation	Reject the null hypothesis: The median of "SOLID" is greater than "Unstructured Code".		
Effect Size (Cohen's d):	1.82		

Table 4.58: OVERALL - Question 25 - Statistics.

The results of question 25 in all groups are presented in Figure 4.52, where the frequencies of participants' responses are normalized by the type of treatment applied. Table 4.58 presents measures of central tendency, measures of dispersion, and hypothesis test results.

The hypothesis test shows a statistically significant difference in the degree of agreement with the statement between the two groups. The calculated p-value was 0.0001, which is below the alpha value of 0.1, with a large effect size of 1.82. This indicates that there is statistical evidence to suggest that Treatment 1 (SOLID) leads to greater agreement with the statement than Treatment 2 (Unstructured Code).

5
Discussion

We discuss the obtained results for each SOLID principle based on the results shown in Table 5.1, focusing on the observed statistical significance and effect sizes.

Table 5.1: Consolidated results.

SRP												
	BARI			PUC			SERPRO			Overall		
Question	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size
1- What is your perception about understanding the source code?	0.35	No		0.2	No		0.01	Yes	0.88	0.02	Yes	0.43
3- The responsibilities in the code above are clearly distributed and defined. What is your degree of agreement with the statement?	0.01	Yes	1.15	0.01	Yes	2.25	0.01	Yes	1.46	0.01	Yes	1.56
5- The code block above has classes structured to have a single responsibility in the software, that is, they are specialized in a single subject. What is your degree of agreement with the statement?	0.01	Yes	1.37	0.01	Yes	3.31	0.01	Yes	1.72	0.01	Yes	1.88
7- The code above is structured so that your classes have a single reason for change. What is your degree of agreement with the statement?	0.01	Yes	0.98	0.01	Yes	1.86	0.01	Yes	1.46	0.01	Yes	1.42
OCP												
	BARI			PUC			SERPRO			Overall		
Question	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size
9- What is your perception about understanding the source code?	0.19	No		0.21	No		0.02	Yes	0.72	0.01	Yes	0.48
11- It was easy to extend the behavior of the software with the addition of new models. What is your degree of agreement with the statement?	0.04	Yes	0.50	0.01	Yes	1.1	0.01	Yes	0.65	0.01	Yes	0.75
13- Adding new models did not imply changing pre-existing code. What is your degree of agreement with the statement?	0.07	Yes	0.52	0.01	Yes	1.44	0.01	Yes	1.29	0.01	Yes	0.82
LSP and DIP												
	BARI			PUC			SERPRO			Overall		
Question	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size
15- What is your perception about understanding the source code?	0.02	Yes	0.78	0.01	Yes	0.73	0.02	Yes	0.71	0.01	Yes	0.74
17- The models can be replaced without changing the controller. What is your degree of agreement with the statement?	0.09	Yes	0.43	0.27	No		0.01	Yes	0.45	0.01	Yes	0.3
19- Changes in a model implementation do not imply changes in the controller. What is your degree of agreement with the statement?	0.07	Yes	0.59	0.36	No		0.01	Yes	1.07	0.01	Yes	0.54
21- The controller has loose coupling with model implementations. What is your degree of agreement with the statement?	0.03	Yes	0.67	0.46	No		0.01	Yes	1.32	0.01	Yes	0.65
ISP												
	BARI			PUC			SERPRO			Overall		
Question	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size	p-value	Reject H0	Effect Size
23- What is your perception about understanding the source code?	0.06	Yes	0.55	0.04	Yes	0.72	0.04	Yes	0.48	0.01	Yes	0.55
25- The operations in the evaluator implementations are properly segregated for evaluation of classification and regression algorithms. What is your degree of agreement with the statement?	0.01	Yes	0.92	0.01	Yes	2.14	0.01	Yes	3	0.01	Yes	1.82

5.1
SRP - Single Responsibility Principle

In the context of the SRP principle, the results show that the null hypothesis (H0) could not be rejected for the perception of the difficulty of understanding the source code (Question 1) for the trials at the University of

Bari and PUC-Rio but could be rejected for the SERPRO trial and considering the aggregated results. Still, it is possible to observe a slightly higher perceived ease of understanding for the SOLID treatment in all trials (*cf.* Figures 4.1 and 4.2). Not achieving statistically significant differences for the University of Bari and PUC-Rio could be due to the slightly smaller sample size.

While a complete qualitative analysis of the open-ended questions is not the main scope of this dissertation, we used this data aiming at better understanding situations in which HO could not be rejected. At BARI, three participants of the SOLID treatment considered the code hard to understand. Out of these three, two explained their answers. One mentioned “I’m quite new in the ML field, i’ve just studied some concepts for my personal projects," while the other one justified “I didn’t understand the part of features and labels because I have no solid base of ML." Hence, their difficulties were apparently more closely related to specific ML-related implementations than to the code structure. At PUC-Rio, only one participant of the SOLID treatment considered the code hard to understand, justifying it with “lack of practice with object-oriented programming". Another important aspect is that the original code was relatively small, potentially making the benefits of distributing responsibilities clearer for the subsequent maintenance scenarios. Indeed, many subjects of the Unstructured treatment also found the code easy to understand. The limited sample size within a single trial can have isolated cases of confounding factors leading to non-statistically significant results in some cases.

The SRP questions regarding clearly distributed and defined responsibilities (Question 3), single responsibilities (Question 5), and single reasons for change (Question 7) allowed rejecting H0, with significantly low p-values and high effect sizes. These results indicate that when it comes to specific aspects related to SRP, a significant and positive impact was perceived by the participants.

Finding 1: Applying the SRP principle favors ML code understanding and leads to the perception of benefits related to having clearly distributed and defined ML code responsibilities, favoring future maintenance by having single responsibilities and single reasons for change.

5.2 OCP - Open-Closed Principle

For OCP, the null hypothesis (H0) could also not be rejected concerning the overall perception of the difficulty of understanding the related source code

(Question 9) for the University of Bari and PUC-Rio. Nevertheless, a slightly higher perceived ease of understanding was observed for the SOLID treatment for all trials, even for the University of Bari and PUC-Rio (the difference can be observed in the graphs provided in the online material (ANONYMOUS, 2023)). Not achieving statistically significant differences could be due to the smaller sample size. Indeed, it was possible to reject H0 for the SERPRO trial and for the aggregated results.

Looking at the qualitative data at the University of Bari, two participants of the SOLID treatment considered the code difficult to understand, potentially leading to the lack of statistical significance in the differences. Out of these two, one provided an explanation "I don't know these functions well", apparently referring to difficulties related to the new specific regression ML algorithms added as part of the OCP scenario and not to the code structure. At PUC-Rio, all participants perceived the code as easy or very easy to understand. Still, the difference favoring the SOLID treatment for PUC-Rio was not statistically significant.

The OCP questions regarding ease of extension (Question 11) and not implying changing existing code (Question 13) allowed rejecting H0, with significantly low p-values and medium to high effect sizes. This indicates that the application of OCP had a positive impact on the ease of extending the software without modifying existing code.

> **Finding 2:** Applying the OCP principle favors ML code understanding and leads to the perception of benefits related to facilitating ML code extensions without substantially changing existing code.

5.3 LSP and DIP - Liskov Substitution Principle and Dependency Inversion Principle

For the LSP and DIP principles, the null hypothesis (H0) could be rejected with statistical significance and medium to high effect sizes concerning the overall perception of the difficulty of understanding the related source code (Question 15), indicating that the application of the principles made it easier to understand the ML code.

The specific questions related to model substitution (Questions 17 and 19) and low coupling (Question 21) also suggest that these principles had a significant and positive impact on the participants' perception regarding these aspects. An exception was observed for the trial at PUC-Rio, where these questions, while having answers slightly favoring the SOLID treatment

(ANONYMOUS, 2023), did not allow rejecting H0. Digging into the qualitative data allowed us to observe that for Question 17, out of the participants of the SOLID treatment at PUC-Rio, only one participant strongly disagreed, and one disagreed that models could be replaced without changing the controller, potentially leading to the lack of statistically significant differences within this trial. In their justifications, both argued that the controller had to be modified to support a list of models. Hence, they slightly misinterpreted the question, which referred to the ability to replace models within the list of models handled by the improved controller. This same misunderstanding led the one who strongly disagreed to also disagree with Questions 19 and 21, with which the participant who disagreed with Question 17 agreed, observing that changes in the implementation of a model would not affect the controller.

Finding 3: Applying the LSP and DIP principles favors ML code understanding and leads to the perception of benefits related to the flexibility of substituting ML code elements and favoring low coupling.

5.4 ISP - Interface Segregation Principle

For the ISP principle, the results show that the null hypothesis (H0) could be rejected with statistical significance and medium to high effect sizes for all questions related to this principle. This indicates that the application of ISP had a significant impact on the ease of understanding the ML source code (Question 23). Furthermore, it indicated that the ISP had a significantly positive impact on the participants' perception of proper segregation of operations, in this case, related to establishing separate evaluation interfaces for classification and regression algorithms.

Finding 4: Applying the ISP principle favors ML code understanding and leads to the perception of proper segregation of interface operations, not forcing ML code to depend upon methods that it does not use.

5.4.1 Implications

In summary, the consolidated analysis of the results with the combination of all trials indicates that the application of SOLID principles has a significant positive impact on ML code understanding and leads to the perception of

several other benefits related to the ML system's maintainability. These results provide empirical evidence allowing us to assert that the application of these principles is beneficial for improving ML code understanding and maintainability.

Implications for researchers include having evidence to ground themselves when referring to the need and benefits of applying software engineering design best practices within the ML context. Of course, this study can also foster additional experimental replications, potentially varying the experimental objects and involving larger and more diverse samples. Furthermore, additional studies are needed to investigate the effects of applying many other software engineering practices within the ML context.

Implications for data science practitioners include the need to familiarize themselves with software engineering design best practices in order to write more robust ML code. For educators, the results indicate that, as society becomes more and more dependent on ML-enabled systems, data science curricula should include software engineering skills to ensure that these systems and their ML components are built with quality and that they are easy to maintain.

6
Threats to validity

In this chapter, we describe the threats to validity faced by this research and the mitigation actions taken to control them within our possibilities. We organize the threats according to the categories described by Wohlin *et al.* (WOHLIN et al., 2012).

6.1
Internal Validity

Threats to internal validity are trial-related influences that can affect the independent variable with respect to causality (WOHLIN et al., 2012). The instrument was applied without any assistance from the researchers. Hence, participants were not monitored during their activities, and some of them could have conducted the task with less attention or within environments facing interruptions, which may have affected their code understanding. Participants were volunteers, and we had no way to control this threat. However, all participants completed the tasks until the end and provided justification for at least some of the open-ended questions, which leads us to assume that they participated with attention.

Furthermore, before running the experimental trials, we conducted a pilot study with four participants, which allowed us to observe an average effort of 20 to 30 minutes. The pilot allowed us to improve the instrument with some simplifications (removing two questions that were considered redundant) and minor adjustments (*e.g.*, adjusting the size of the code, some snippets to improve readability, and adding definitions for some concepts). We considered the effort reasonable for retaining the attention of the participants.

6.2
External Validity

Threats to external validity are conditions that limit our ability to generalize the results of our experiment to industrial practice (WOHLIN et al., 2012). Two threats to external validity involve the representativeness of our subjects and the representativeness of our experimental object (i.e., the ML code used as a basis for the experiment) and tasks.

With respect to the subjects, we aimed to mitigate this threat by carefully characterizing them, allowing us to discuss their representativeness. Indeed, while we had data scientists with varying levels of experience in all three

samples, 75% of our sample received education within the area of computer science. This does not properly reflect the observation of widely varying educational backgrounds reported in the literature (AHO et al., 2020; KIM et al., 2016). An explanation for this is that the data science graduate programs at both universities were part of the informatics departments and that the company was an IT company. Nevertheless, applying blocking allowed us to observe that an improvement in the ease of ML code understanding when SOLID principles are applied could also be observed for data scientists without a computer science degree. Due to the small sample size within this group not favoring significance testing or effect size calculations, we did not include this (rather weak) analysis in the dissertation. Indeed, as commonly done in controlled experiments in software engineering, we relied on convenience sampling to recruit the best sample we were able to in order to make this investigation happen and carefully describe its limitations. To further address this threat, we call for additional external replications with more diverse subjects.

Regarding the experimental object, to improve representativeness, we used the real ML code of a solution deployed at an industrial partner of PUC-Rio. This ML code was developed using Jupyter notebooks and without applying SOLID principles. For the SOLID treatment, to avoid any confounding factors, we strictly redesigned the code by applying the principles. It is noteworthy that the authors reviewed the code and that they include experts in software design. To assess the code in a maintenance context, we also elicited typical evolution scenarios that would make sense for the ML context based on our experience with delivering ML-enabled systems to industrial partners. These scenarios involved implementing several ML algorithms, assessing a list of ML models, and assessing regression and classification algorithms. All of these are commonly discussed tasks within the data science domain. Despite the care, we also call for additional replications involving other experimental objects and tasks.

6.3 Construct Validity

Construct validity concerns how well the treatments and outcome measurements of the experimental design reflect the causes and effects being assessed. We used real ML code as the basis for designing the experimental object. For one treatment, we used the ML code as it was, while for the other, we strictly applied the SOLID principles. Hence, this was the only difference in the treatments, and therefore, we believe that using the SOLID principles

(or not) properly reflects the cause for observed differences (measured using the exact same questions and scales).

To avoid hypothesis guessing, we only informed participants that the study aimed to understand the impact of design principles on understanding machine learning code. They did not know which treatment they were (randomly) assigned to. Also, their participation was on a volunteer basis, and we informed them that the research would be conducted anonymously, avoiding the evaluation apprehension threat.

To assess whether confounding factors would appear during the experimental tasks, we conducted a pilot study, which allowed us to improve the instrument further. We used the random assignment experimental design practice and characterized participants to control other potential confounding factors, such as background and experience. Nevertheless, we still observed some isolated confounding factors taking place, like a few participants not properly understanding some of the questions. Based on the qualitative analyses of the open-ended justifications, we observed that these were rare and isolated cases, not severely affecting the overall results. The instrument for each treatment and all the collected data are available online (ANONYMOUS, 2023).

6.4 Conclusion Validity

Conclusion validity refers to the degree to which conclusions about the relationship among variables based on the data are correct from a statistical point of view (WOHLIN et al., 2012). Conclusion validity can be affected by the sample size. We recruited 100 data scientists from 3 different origins: two universities and one company. For the inferential statistics, we conservatively employed the Mann-Whitney test with an alpha value of 0.1. Mann-Whitney is a non-parametric statistical test that doesn't pose assumptions on the data distribution, and that can be safely applied to ordinal scales. Furthermore, we observed mainly comparable scenarios for the different experimental trials, improving our confidence in the results.

7
Conclusion

7.1
Contributions

ML projects present unique challenges in terms of code understanding. Among the primary factors contributing to these challenges are the inherently experimental nature of ML and the variety of educational backgrounds of data scientists. Despite these challenges and the importance of code understanding for effective maintenance, we found no studies related to ML code comprehension in the literature.

In this dissertation, we take a step towards addressing this gap, investigating the impact of SOLID design principles on ML code understanding by data scientists. We conducted a controlled experiment involving 100 data scientists from three different organizations. The control group was presented with ML code from a real industrial setting that did not incorporate SOLID principles. The experimental group was presented with that same ML code restructured by applying the SOLID principles. Subsequently, the data scientists were tasked with analyzing the code and filling out a questionnaire that included both closed-ended and open-ended questions related to their understanding of the code and their agreement with statements related to typical implications of applying the SOLID principles.

The results indicate that the adoption of each of the five SOLID design principles can significantly facilitate ML code understanding. Moreover, the application of the principles was also perceived to lead to expected benefits related to applying the SOLID principles. These benefits include having clearly defined ML code responsibilities, facilitating ML code extensions without substantially changing existing code, enabling substituting ML code elements, favoring low coupling, and proper segregation of interfaces.

Our results highlight the relevance of SOLID principles in machine learning Software projects and suggest further research to enhance the relationship between data science and traditional Software Engineering principles and best practices. In light of these findings, we propose the dissemination of software engineering design principles within the data science community, advocating for their consideration as a means to enhance the maintainability of ML code.

Additionally, exploring code understanding in machine learning has implications for the broader field of data science. As we observed, data

scientists often come from diverse educational backgrounds without formal software engineering training. Our research emphasizes the need for data scientists and machine learning professionals to recognize the importance of code understanding and the potential benefits of adhering to SOLID Design Principles in their projects. These principles can serve as a bridge between data science and software engineering, enhancing the effectiveness of interdisciplinary teams.

7.2 Limitations

While this research has provided valuable insights into the impact of SOLID principles on code understanding in machine learning projects, it is important to acknowledge the limitations that should be considered when interpreting the results and applying the findings. These limitations include:

- **Sample Size:** Sample size is a typical limitation in experimental studies in software engineering. We managed to involve 100 participants in the experiment. Although participants were recruited from different groups (students and professionals), it is important to note that generalizing the results to a broader population may still have some risks. Future studies with additional samples could provide a more comprehensive view of the effects of SOLID principles in different machine learning software development contexts.
- **Experiment Context:** The experiment was conducted in a controlled environment where participants were asked to analyze and evaluate code in accordance with SOLID principles guidelines. While this method has its advantages, it may not fully reflect the complexities of real-world software development in machine learning projects. Therefore, applying the results to practical situations requires careful consideration of the nuances of real development.
- **Subjective Nature of Participant Responses:** Participant responses regarding code understanding and agreement with SOLID principles statements are subjective. Each individual's perception may vary and can be influenced by individual factors. While measures were taken to minimize bias, the subjectivity of responses should be considered when interpreting the results.
- **Focus on SOLID principles:** This research specifically focused on SOLID principles of software design. Other software engineering practices and factors that may affect code understanding in machine learning

projects were not addressed. Therefore, the results should not be seen as a comprehensive assessment of all aspects of software development in this context.

- **Exploratory Nature of the Study**: This research is evaluative but not conclusive. It provides preliminary evidence of the impact of SOLID principles on code understanding in machine learning projects, but additional studies are needed to validate and deepen these findings.

In summary, this research offers valuable insights, but it's important to acknowledge its limitations for a proper interpretation of the results. Considering these limitations, the results of this study can serve as a starting point for future research and software development practices in machine learning projects.

7.3 Future work

This master's dissertation paves the way for several areas of research concerning the understanding of code in machine learning projects. Based on the findings and limitations identified in this research, the following suggestions are proposed for future work:

- **Experiment Replication and Expansion:** Conduct a replication of the experiment with a larger and more diverse sample of participants, including developers with different levels of experience and contexts of machine learning applications. This would help validate and deepen the conclusions obtained in this dissertation.
- **Longitudinal Study:** Carry out a longitudinal study to assess how the impact of SOLID principles on code understanding evolves over time in machine learning projects. This would allow for an analysis of whether the initial improvements are sustained throughout the project's lifecycle.
- **Code Complexity Variation:** Investigate how code complexity affects the effectiveness of SOLID principles in machine learning projects. It would be interesting to evaluate whether the observed benefits are more pronounced in high-complexity projects.
- **Comparison with Other Software Engineering Practices:** Conduct comparative studies to assess how SOLID principles compare to other software engineering practices in terms of their impact on code understanding in machine learning projects. This could include the analysis of best practices specific to machine learning projects.

- **Development of Support Tools:** Develop code analysis and design tools that assist developers in effectively applying SOLID principles in machine learning projects. These tools could provide real-time feedback and guidance on how to improve the code.
- **Cost-Benefit Analysis:** Perform a cost-benefit analysis to evaluate whether the application of SOLID principles in machine learning projects is worthwhile in terms of development effort versus improvements in code maintainability and understanding.
- **Application in Different machine learning Domains:** Extend the research to assess the impact of SOLID principles in different domains of machine learning applications, such as computer vision, natural language processing, and others.

These suggestions represent a direction for future research related to code understanding in machine learning projects.

8
Bibliography

AHO, T. et al. Demystifying data science projects: A look on the people and process of data science today. In: SPRINGER. **Product-Focused Software Process Improvement: 21st International Conference, PROFES 2020, Turin, Italy, November 25–27, 2020, Proceedings 21**. [S.l.], 2020. p. 153–167.

ANONYMOUS. **Artifacts: Investigating the Impact of SOLID Design Principles on Machine Learning Code Understanding**. 2023. <https://zenodo.org/doi/10.5281/zenodo.7396502>.

BASILI CALDIERA, H. D. The goal question metric approach. **Encyclopedia of software engineering**, p. 528–532, 1994.

BRÄUER, J. et al. Measuring object-oriented design principles: The results of focus group-based research. **Journal of Systems and Software**, Elsevier, v. 140, p. 74–90, 2018.

CORBI, T. A. Program understanding: Challenge for the 1990s. **IBM Systems Journal**, IBM, v. 28, n. 2, p. 294–306, 1989.

ExACTa. **About the ExACTa**. 2023. <https://exacta.inf.puc-rio.br/#:~:text=ExACTa%20%C3%A9%20uma%20iniciativa%20da,Otimiza%C3%A7%C3%A3o%20e%20Intera%C3%A7%C3%A3o%20Humano%20Computador.> Accessed: 2023-05-02.

FJELDSTAD, R. K. Application program maintenance study. **Report to Our Respondents, Proceedings GUIDE**, v. 48, 1983.

FOUNDATION, P. S. **Python**. 2023. <https://www.python.org/>.

GAMMA, E. et al. **Design patterns: elements of reusable object-oriented software**. [S.l.]: Pearson Deutschland GmbH, 1995.

GONÇALES, L. J. et al. Evaluation of machine learning techniques to classify code comprehension based on developers' eeg data. In: **Proceedings of the 19th Brazilian Symposium on Human Factors in Computing Systems**. [S.l.: s.n.], 2020. p. 1–10.

HARRIS, H.; MURPHY, S.; VAISMAN, M. **Analyzing the analyzers: An introspective survey of data scientists and their work**. [S.l.]: " O'Reilly Media, Inc.", 2013.

HUANG, Y. et al. Does your code need comment? **Software: Practice and Experience**, Wiley Online Library, v. 50, n. 3, p. 227–245, 2020.

KALINOWSKI, M. et al. **Engenharia de software para ciência de dados: um guia de boas práticas com ênfase na construção de sistemas de machine learning**. [S.l.]: Casa do Código, 2023.

Ken Thompson. **Unix philosophy**. 2022. <https://en.wikipedia.org/wiki/Unix_philosophy>. Accessed: 2022-12-11.

KIM, M. et al. The emerging role of data scientists on software development teams. In: IEEE. **2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)**. [S.l.], 2016. p. 96–107.

KIM, M. et al. Data scientists in software teams: State of the art and challenges. **IEEE Transactions on Software Engineering**, IEEE, v. 44, n. 11, p. 1024–1038, 2017.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. Citeseer, 2007.

KO, A. J. et al. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. **IEEE Transactions on software engineering**, IEEE, v. 32, n. 12, p. 971–987, 2006.

LARMAN, C. et al. **Applying UML and patterns**. [S.l.]: Prentice Hall Upper Saddle River, 1998. v. 2.

LEONARDO, R. Pico: model for clinical questions. **Evid Based Med Pract**, v. 3, n. 115, p. 2, 2018.

LLC, G. **Cobab**. 2023. <https://colab.google/>.

MARTIN, R. C. Principles of ood. **Online]. Tersedia: butunclebob. com/ArticleS. UncleBob. PrinciplesOfOod.[Diakses pada Juli 2018]**, 1995.

MARTIN, R. C. Design principles and design patterns. **Object Mentor**, v. 1, n. 34, p. 597, 2000.

MARTIN, R. C. **Agile software development: principles, patterns, and practices**. [S.l.]: Prentice Hall PTR, 2003.

MARTIN, R. C. **Clean code: a handbook of agile software craftsmanship**. [S.l.]: Pearson Education, 2009.

MAY, T. **The new know: innovation powered by analytics**. [S.l.]: John Wiley & Sons, 2009.

MINELLI, R.; MOCCI, A.; LANZA, M. I know what you did last summer-an investigation of how developers spend their time. In: IEEE. **2015 IEEE 23rd International Conference on Program Comprehension**. [S.l.], 2015. p. 25–35.

MOURÃO, E. et al. On the performance of hybrid search strategies for systematic literature reviews in software engineering. **Information and Software Technology**, Elsevier, v. 123, p. 106294, 2020.

MUSIC, A. **SOLID Principles for Data Science and Machine Learning – Improve your coding by sticking to 5 simple principles**. 2023. <https://anelmusic13.medium.com/solid-principles-for-data-scientist-adbc9edb6151>. Accessed: 2023-10-31.

NUMFOCUS, I. **Pandas**. 2023. <https://pandas.pydata.org/>.

OORT, B. V. et al. The prevalence of code smells in machine learning projects. In: IEEE. **2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)**. [S.l.], 2021. p. 1–8.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and software technology**, Elsevier, v. 64, p. 1–18, 2015.

PIMENTEL, J. F. et al. A large-scale study about quality and reproducibility of jupyter notebooks. In: IEEE. **2019 IEEE/ACM 16th international conference on mining software repositories (MSR)**. [S.l.], 2019. p. 507–517.

PIMENTEL, J. F. et al. Understanding and improving the quality and reproducibility of jupyter notebooks. **Empirical Software Engineering**, Springer, v. 26, n. 4, p. 65, 2021.

SHALABY, M. et al. Automatic algorithm recognition of source-code using machine learning. In: IEEE. **2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)**. [S.l.], 2017. p. 170–177.

SHEN, H. Interactive notebooks: Sharing the code. **Nature**, Nature Publishing Group UK London, v. 515, n. 7525, p. 152–152, 2014.

SOULY, A. **Towards Data Science: 5 Principles to write SOLID Code**. 2023. <https://towardsdatascience.com/5-principles-to-write-solid-code-examples-in-python-9062272e6bdc>. Accessed: 2023-10-31.

TANG, Y. et al. An empirical study of refactorings and technical debt in machine learning systems. In: IEEE. **2021 IEEE/ACM 43rd international conference on software engineering (ICSE)**. [S.l.], 2021. p. 238–250.

UMANEO. **The SOLID Principles Applied to Machine Learning**. 2023. <https://www.umaneo.com/post/the-solid-principles-applied-to-machine-learning>. Accessed: 2023-10-31.

VELARDO, V. **Online Course: Uncle Bob SOLID Principles for Machine Learning Engineers**. 2023. <https://github.com/musikalkemist/solidforml/tree/main/solidforml>. Accessed: 2023-10-31.

VISWANATHAN, S.; KUMAR, M. A.; SOMAN, K. A comparative analysis of machine comprehension using deep learning models in code-mixed hindi language. In: **Recent Advances in Computational Intelligence**. [S.l.]: Springer, 2019. p. 315–339.

WIERINGA, R. et al. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. **Requirements engineering**, Springer, v. 11, n. 1, p. 102–107, 2006.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: **Proceedings of the 18th international conference on evaluation and assessment in software engineering**. [S.l.: s.n.], 2014. p. 1–10.

WOHLIN, C. et al. **Experimentation in software engineering**. [S.l.]: Springer Science & Business Media, 2012.

WYRICH, M.; BOGNER, J.; WAGNER, S. 40 years of designing code comprehension experiments: A systematic mapping study. **arXiv preprint arXiv:2206.11102**, 2022.

XIA, X. et al. Measuring program comprehension: A large-scale field study with professionals. **IEEE Transactions on Software Engineering**, IEEE, v. 44, n. 10, p. 951–976, 2017.

ZELKOWITZ, M. V.; SHAW, A. C.; GANNON, J. D. **Principles of software engineering and design**. [S.l.]: Prentice Hall Professional Technical Reference, 1979.