UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Eliisabet Kaasik

Software Testing Lab Package: Metamorphic Testing

Bachelor's Thesis (9 ECTS)

Supervisor:
Dietmar Pfahl, PhD

Tartu 2025

Software Testing Lab Package: Metamorphic Testing

Abstract:

As the use of artificial intelligence continues to rise, teaching testing methods to verify such complex applications becomes increasingly important. This thesis aimed to develop a lab package on metamorphic testing for the Software Testing course at the University of Tartu. The thesis includes an overview of the course and the metamorphic testing method, a description of the development process of the materials and the implementation of the lab session, as well as an analysis of the feedback received. The lab took place in the spring semester of 2025 with six student groups, and feedback was collected through a survey. Results show that students found the lab clear, engaging, and relevant to industry needs, with minor improvements needed in the setup instructions.

Keywords: Software testing, metamorphic testing, machine learning, computer science education

CERCS: P175 Informatics, systems theory

Tarkvara testimise praktikumimaterjalid: metamorfne testimine

Lühikokkuvõte:

Tehisintellekti populaarsuse kiire kasvu tõttu muutub neid kasutavate süsteemide testimismeetodite õpetamine järjest olulisemaks. Töö eesmärgina koostati Tartu Ülikooli kursuse „Tarkvara testimine” jaoks metamorfse testimise praktikumimaterjalid. Lõputöö sisaldab ülevaadet kursusest ja metamorfse testimise meetodist, kirjeldust materjalide loomisprotsessist ja praktikumi läbiviimisest ning analüüsi kogutud tagasisidest. Praktikum viidi läbi 2025. aasta kevadsemestril kuues praktikumirühmas ning tagasiside koguti küsitlusena. Tulemused näitasid, et õpilaste arvates oli praktikum selge, kaasahaarav ja asjakohane ning juhistes on vaja teha vaid väiksemaid parandusi.

Võtmesõnad: Tarkvara testimine, metamorfne testimine, masinõpe, informaatika haridus

CERCS: P175 Informaatika, süsteemiteooria

Contents

1. Introduction

Software testing is essential to software development, ensuring systems operate correctly and reliably under various conditions. As systems grow more complex, due in part to advancements in machine learning (ML) and artificial intelligence (AI), conventional approaches to testing are often no longer sufficient. In many cases, verifying a system's output becomes extremely challenging or even impossible because of the oracle problem [1]. Metamorphic testing (MT), introduced by Chen et al. [2], addresses this issue by verifying properties of software behaviour through relationships between multiple program executions rather than relying on a known correct output.

Despite its promise and growing relevance in software testing, MT is not yet widely covered at the undergraduate level in computer science education. As the use of AI continues to rise, it is increasingly important that computer science courses integrate AI-relevant topics, not only in specialised courses but throughout the curriculum. The goal of this work is to design, implement, and evaluate a lab package that teaches computer science students to test AI-based software systems using metamorphic testing. The lab introduces students to the concept and application of MT through two practical tasks covering its main use cases: testing scientific computation software and machine learning models. By including metamorphic testing in the software testing course, this work prepares students for the reality of testing complex AI-related systems and directly addresses the educational gap in modern software testing.

This thesis contains four main chapters. The first chapter covers the background information about the course, an overview of metamorphic testing and a view into how metamorphic testing is taught in universities. The second chapter gives insight into the lab design process: a summary of the tasks, how they were designed, what materials the students receive, the grading guide and lab schedule. The third chapter covers the lab execution, including direct observations and responses from the instructors of the unobserved labs. The last chapter summarises and analyses the feedback collected from the students after the lab and the immediate improvements made to the materials based on the feedback and future development opportunities.

2. Background

This chapter gives an overview of the course contents and metamorphic testing.

2.1 Information About the Course

Software Testing (LTAT.05.006) [3] is a course offered in the spring semester by the University of Tartu Computer Science curriculum as a part of the Software Development Speciality Module. The purpose of the course is to introduce students to different testing strategies to ensure software quality. The topics covered vary from basic testing strategies, such as black-box testing and debugging, to more advanced techniques like mutation testing.

The metamorphic testing lab, created as a part of this thesis, replaces last year's 7th lab, "Scriptless GUI Testing with TESTAR". This change was made to introduce students to increasingly important concepts in the context of AI-based systems. As artificial intelligence continues to play a greater role in software development, it is necessary for students to become familiar with testing methods, like metamorphic testing, that are valuable in verifying such complex applications.

2.2 Metamorphic Testing

In software development, testing is one of the most critical processes to ensure a system behaves as expected. Traditional testing methods often rely on the presence of a test oracle[1], which plays a key role in verifying software behaviour. Metamorphic Testing (MT) was first introduced by Chen et al. in 1998 [2] as a new approach to software testing in cases where a test oracle is unavailable or unreliable.

2.2.1 Metamorphic Testing as a Solution to the Oracle Problem

One of the main motivations behind metamorphic testing is its potential to address this issue. According to Zheng et al. [1], there are many cases where the oracle is either unknown or impractical to apply, making it challenging to determine whether the output of the system under test (SUT) is correct. This leads to what is known as the *oracle problem*.

[1]A *test oracle* is a mechanism or procedure used to determine whether the output of a program is correct or incorrect [4].

Segura et al. [5] describe the usefulness of metamorphic testing in addressing this problem. According to them, metamorphic testing is an effective method for verifying the correctness of so-called "untestable" programs, where errors cannot be easily detected by simply checking the outputs, as there are no test oracles. They explain how the MT strategy differs from other approaches: instead of focusing on individual outputs, MT examines the results of multiple program executions by checking whether the inputs and outputs satisfy certain properties. These properties are called metamorphic relations (MRs). An example of such a relation is provided in the next section.

2.2.2 Applications of Metamorphic Testing

Metamorphic testing has applications in many fields where more standard testing methods are insufficient. One usage of metamorphic testing lies in verifying scientific computations, where complicated calculations make determining the correct result difficult. Lin et al. [6] provided the following example in their article:

When testing a program that calculates the sine function, knowing the correct output for an individual input can be challenging. For instance, the exact value of $\sin(12)$ may depend on how floating-point arithmetic is implemented.

To address this, one can use the mathematical property that:

$$\sin(x) = \sin(\pi - x)$$

This means $\sin(12)$ should have the same value as $\sin(\pi - 12)$. If the outputs for these inputs differ, it indicates a program fault. This trigonometric identity, $\sin(x) = \sin(\pi - x)$, is an example of a metamorphic relation, which forms the foundation of the entire concept of metamorphic testing. A visualisation of this metamorphic relation is shown in Figure 1

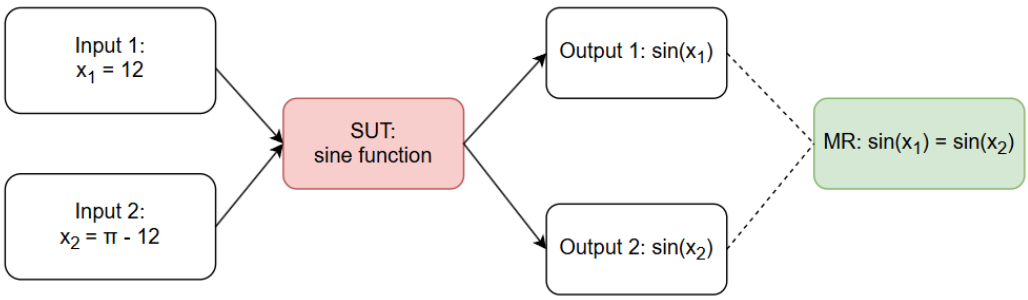


Figure 1. Illustration of a metamorphic relation, adapted from an example in Lin et al. [6].

Another notable usage for MT is in machine learning algorithms. Saha and Kanewala [7] described in their research non-testable programs, such as classifier models, whose correctness often cannot be determined using more traditional techniques. According to them, since classifiers are not always perfectly accurate, an incorrect result may not necessarily indicate a program fault, which makes testing machine learning models complex.

One of the main challenges in testing ML models, specifically image classification models, is their nondeterministic[2] behaviour. As Cooper et al. noted [9], machine learning systems inherently exhibit nondeterminism due to their reliance on stochastic (i.e., random) processes, making it hard to ensure consistent decision-making across different executions of the same algorithm. This differs from deterministic programs, such as calculators, which always produce the same output if the input is unchanged. Since software testing courses typically only cover testing methods for evaluating deterministic programs, it is beneficial for students to learn alternative approaches such as metamorphic testing.

2.3 Teaching Metamorphic Testing

While still not widely taught, metamorphic testing is increasingly appearing in software testing curricula, especially at the Bachelor's level. For example, the Singapore University of Technology and Design's course Software Testing and Verification [10] introduces metamorphic testing near the end of the syllabus as part of its advanced testing methodologies. The course covers topics such as testing without oracles and evaluating ML-based systems, positioning metamorphic testing alongside techniques like fuzzing and differential testing. Similarly, Idaho State University's (ISU) course Software Testing and Quality Assurance [11] highlights metamorphic testing with other advanced techniques such as mutation testing and test automation.

The inclusion of metamorphic testing in university courses reflects its growing relevance in software testing. One industry observer, Michael Stahl [12], stated that while MT has previously been absent from most of the materials for basic or advanced software testing, it is now gaining popularity seemingly due to its utility for testing AI-based applications. This sentiment is also supported by a growing trend in publications on the subject, as summarised by Segura et al. [13], who analysed 119 papers on metamorphic testing and highlighted its increasing prominence in scientific literature.

[2]A computation is nondeterministic if a given input can result in different outputs [8].

Previous teaching experiences, summarised by Chen et al. [14], have shown that students generally understand metamorphic testing concepts easily and can apply them in practice, which makes MT a suitable topic for higher education in software testing. In particular, the task of identifying metamorphic relations has been found to increase student engagement and creativity in testing exercises. However, due to the relative novelty of metamorphic testing, challenges such as the lack of high-quality learning materials remain. At the University of Tartu, metamorphic testing was previously not covered as a lab topic in the Bachelor's level Software Testing course, which has created the need for suitable lab materials, addressed in this thesis.

3. Lab Design

This chapter outlines the content and structure of the metamorphic testing lab, giving an overview of the materials and the processes behind conceptualising the tasks.

3.1 Lab Overview

The purpose of this lab is to provide students with a practical understanding of metamorphic testing and its usefulness by applying it to realistic scenarios relevant to autonomous vehicles. Students must identify the most reliable software for two critical functions of a self-driving car: braking distance calculation and traffic sign recognition. The tasks were developed to illustrate the strengths of metamorphic testing in complex systems. Moreover, the two tasks were chosen to reflect the main applications of metamorphic testing: testing scientific computations and machine learning.

The braking distance calculator task was developed because it represents a system based on physical and mathematical relationships, which can easily be expressed as metamorphic relations. This follows Lin et al.'s demonstration that metamorphic testing is well-suited to scientific calculation programs [6]. It is an appropriate introduction to metamorphic testing since the students can use real-world experience and intuition to decide whether the output changes as expected when the input parameters, like speed or slope, are altered. This is a straightforward example of how metamorphic testing can be applied in scientific computing. Alternative tasks, like testing basic math operations, would be too simple and fail to demonstrate the benefits of this testing method.

The traffic sign classification task was selected to introduce students to the application of metamorphic testing in machine learning systems. Because image classifiers rarely have a reliable oracle, this task tackles the wider oracle problem that Saha and Kanewala highlight as prevalent in ML systems [7]. It presents a realistic and relevant challenge, particularly suited to demonstrate how intuitive metamorphic relations can be used to assess the reliability of complex black-box models. By constructing the task within a familiar context, students can better appreciate the practical value of MT. The task highlights how MRs, in this case, modifying the test images, can serve as functional substitutes for test oracles in ML applications.

3.2 Lab and Homework Tasks

The lab session is intended to introduce students to the homework tasks and set up the programs to complete the assignment successfully. In the first task, students set up the braking distance

calculators and run an initial test based on a given metamorphic relation. Once they understand the concept, they continue the homework by identifying and testing additional MRs to find the best calculator out of the four provided. In the second task, students set up the testing environment for the traffic sign classifiers, review the example relation, and then proceed to modify the data and evaluate the models as part of their homework.

3.2.1 Task 1: Braking Distance Calculators

Braking distance refers to the distance a vehicle travels from the moment brakes are applied until it comes to a complete stop. The distance depends on several factors, including vehicle speed, road conditions, and reaction time before pressing the brake. The braking distance calculation formula (Equation 1) was adapted from Omni Calculator's Stopping Distance Calculator[3] and modified to add an additional parameter for braking strength:

$$s = \frac{0.278 \cdot t \cdot v + \frac{v^2}{254 \cdot (\mu + G)}}{B} \tag{1}$$

This formula was chosen due to the large number of parameters (see Table 1), which makes the task more difficult and allows students to think of more MRs.

Table 1. Stopping distance formula parameters.

Variable	Description
s	Total stopping distance (meters)
t	Reaction time (seconds)
v	Speed of the car (km/h)
G	Road grade (slope), ranging from -1.0 to 1.0
μ	Coefficient of friction (0 to 1.0)
B	Braking strength (0 to 1, where 1 is full braking)

The calculators used in this lab are implemented as Java classes; the classes share a common interface to ensure consistency (see Appendix I). There are four calculators in total, three of which contain errors, and one that is correct. Since students must find faults in calculations based on metamorphic relations, not runtime errors or similar bugs, input validation is implemented.

[3]Omni Calculator, available at: https://www.omnicalculator.com/physics/stopping-distance

The calculators are combined into a single black-box application to ensure that students do not have direct access to the formulas within each calculator, which would make finding bugs easier for them. Instead, they interact with the system through a standardised interface, where they have to provide input parameters and analyse the outputs.

Students have two ways of interacting with the system. They can either use the main file, which provides a command-line interface (CLI) for inputting parameters and viewing results, or directly execute test cases through the provided JUnit test suite. The CLI allows users to explore different input scenarios manually and understand the impact of the variables on braking distance. The test suite is used to create tests with the previously identified MRs and test all calculators automatically.

Metamorphic testing is useful for this task since it allows verification of correctness without requiring previously calculated expected outputs, which are not available. Instead, MRs define how the braking distance changes if an input variable is changed. The expected MRs are based on common sense and general world experience. Therefore, students are not expected to have a deep knowledge of physics to identify these relations.

For this task, the example metamorphic relation given to the students is as follows: *If the coefficient of friction decreases (for example, from a dry to a wet surface), the stopping distance should increase.* With this relation, they can test the calculators by reducing the friction parameter while leaving the other parameters unchanged and observing if the output changes accordingly. If the calculated distance decreases, the students should notice this as an indication of a bug. Their goal is to test all of the calculators and find the one without bugs.

3.2.2 Task 2: Traffic Sign Classifiers

In Task 2, the students have to evaluate the performance of four convolutional neural network (CNN) models for traffic sign recognition by applying metamorphic testing. CNNs have consistently shown high performance in image classification tasks [15], including object recognition. For this reason, CNNs were chosen for this lab's traffic sign recognition task, as they provide a widely used approach for solving actual image classification problems.

The models used in this task were individually trained as part of this project to recognise and classify traffic signs into 58 categories. They are intended to be tested across three markets with different environmental challenges: Sweden, California, and Kenya. The objective is to assign a model for each market, with one model being unused.

The models were trained on an open-source dataset from Kaggle: Traffic Sign Dataset - Classification[4]. This dataset was chosen due to its high usability rating on Kaggle, which means it is simple to use and updated frequently. It contains 58 classes, with an average of 73 images per class (see Figure 2). The dataset includes a ”labels.csv” file that provides labels for each class, allowing the categorisation of traffic signs. The data is split into training and test folders.

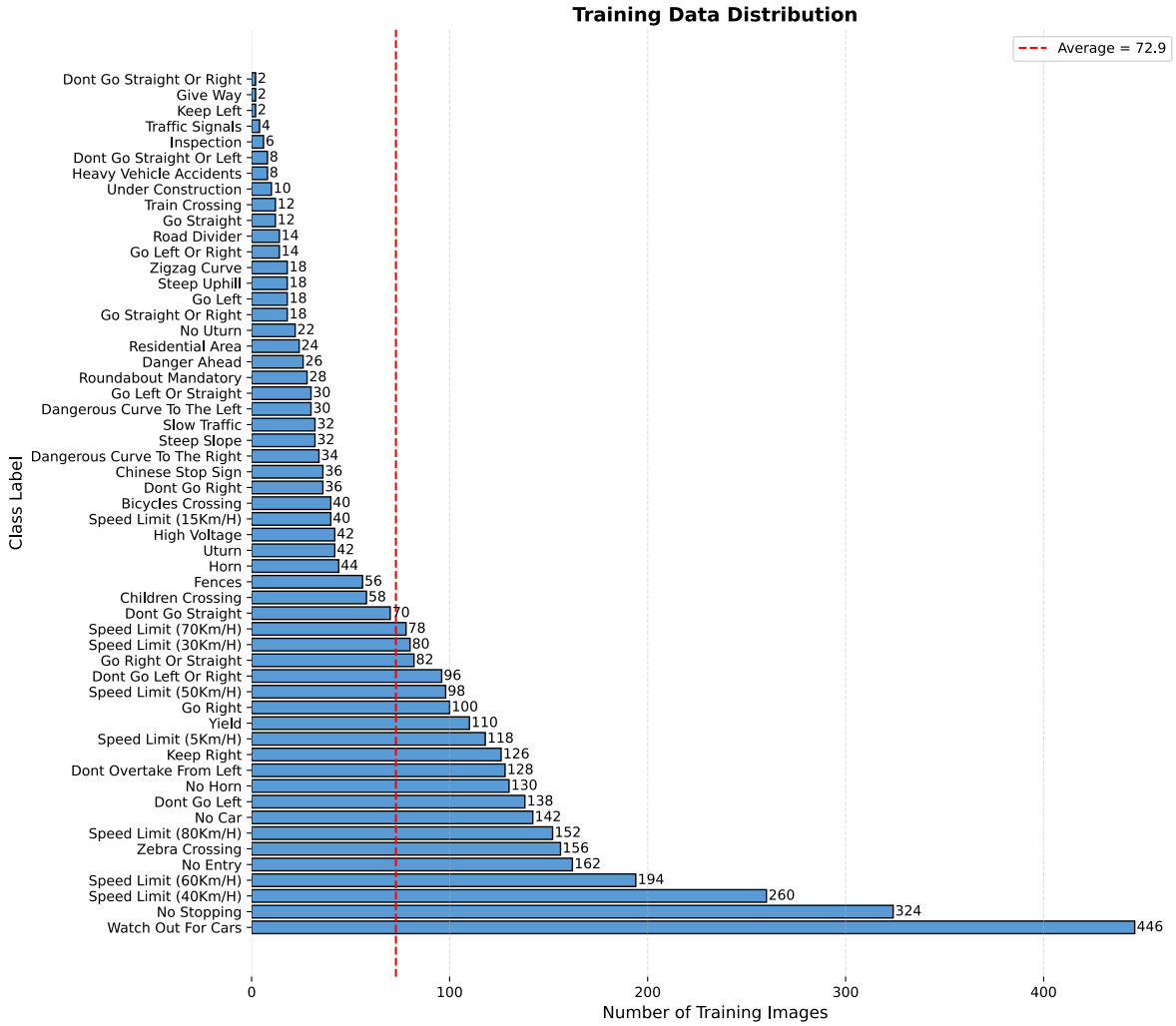


Figure 2. Training data distribution.

A total of four different models were trained for this task. All models underwent the same preprocessing steps: resizing images to a fixed dimension, normalising pixel values to a range of 0 to 1, and converting them into arrays for input into neural networks. The code used to train

[4]Kaggle, Traffic Sign Dataset: https://www.kaggle.com/datasets/sagnik1511/traffic-sign-dataset-classification

the models is available in Appendix I. To improve generalisation and prevent overfitting, the data was augmented slightly using rotations, flipping, contrast adjustments, and other standard data augmentation techniques. The models were trained using a convolutional neural network architecture built with TensorFlow[5] and Keras[6]. The training process used the Adam optimiser and categorical cross-entropy loss, a common setup for multi-class classification tasks, which is therefore suitable for these models. Additionally, hyperparameter tuning was performed to improve model performance. This included testing different sizes for the connected layers (1024, 512 and 256 units), dropout rates (0.3, 0.4 and 0.5) and learning rates (0.001 and 0.0001). The final configuration was 512 and 256 units in the dense layers, a learning rate of 0.001, and a dropout rate of 0.5, which consistently delivered the best validation accuracy without overfitting.

While all models followed the same general training process, models for California, Sweden, and Kenya had some custom modifications to make them more robust to the specific conditions. The modifications involved adding noise and blur to the training data, as well as darkening and brightening the images, based on which model is meant for which environment. The four models have achieved similar F1 scores[7] of approximately 0.85 when tested on unaltered data. This performance is appropriate for this task since the goal was to make acceptable yet imperfect models to allow students to find faults and inconsistencies. However, these scores drop to varying degrees depending on the type of modification applied to the test data, with each model exhibiting different levels of robustness to specific transformations.

MT is used in evaluating the robustness and reliability of the traffic sign classifiers by making slight transformations on the test data. Instead of comparing predictions against predetermined values, metamorphic testing assesses how model behaviour changes when input data is modified according to the previously determined MRs. Slightly altering the input data, for example, rotating the image, should still result in the same predicted label.

To complete this task, the students have to apply metamorphic testing by understanding the market conditions and creating methods to alter the provided test data accordingly. For example, in Sweden, with long, dark winters, they are expected to darken the test data and observe

[5]TensorFlow is an open-source ML framework designed for deep learning applications: https://www.tensorflow.org/
[6]Keras is a high-level API integrated with TensorFlow for training neural networks: https://keras.io/
[7]The F1 score is used to evaluate classification performance, ranging from 0 (worst) to 1 (best) https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

how the predictions of the models change. They then have to compare the consistency of the classifications on altered and unaltered data. If a model's predicted labels differ too greatly between the data, then that indicates a flaw.

3.3 Lab Materials

The lab materials consist of two parts (see Appendix II): lab and homework instructions and a ZIP file containing the SUTs. There are separate instructions for students and lab supervisors.

The PDF file containing the instructions for the students gives an overview of metamorphic testing, the contents of the lab, and the homework tasks. It also includes step-by-step instructions on how to set up both of the programs in the ZIP file and run the first sample tests. The instructions also contain the grading guide, a small part of the solution as an example, and helpful tips to assist students in completing the homework.

The aforementioned ZIP file contains the two systems that students have to test. For the first task, there is a folder named "task1," which contains the program for the first assignment. This includes a JAR file consisting of four braking distance calculators, three of which are faulty and one that is correct. Additionally, there is a test suite named "Testing.java," where students will add their own tests throughout the homework. The materials for the second task are in the folder "task2", which contains four traffic sign classification models, testing data (images of traffic signs), and a testing file. The students' task is to test all the models by modifying the test data, with an example provided in the testing file, reporting the consistency of the predicted labels.

The TA materials are enhanced versions of the materials provided to students, which are available in Appendix I. The lab instructions include additional information about the SUTs, written in red text, correct solutions, and other necessary details, such as more specific grading instructions. Moreover, they have access to the training files through GitHub for each model for future development.

3.4 Lab Schedule

The lab session (including completing the homework) is expected to take a total of 8 academic hours, with 2 hours (90 minutes) allotted for conducting the lab session and the remaining time saved for finishing the homework. The homework can be completed in pairs, halving the time. The schedule for this lab is planned as follows:

- **10 minutes** – Downloading the lab materials.

- **10 minutes** – Introduction to the lab tasks.
- **25 minutes** – Setting up the program for the first task, introducing the task, and explaining the sample solution for finding the first metamorphic relation.
- **30 minutes** – Setting up for the second task, explaining the task, and analysing the first metamorphic relation.
- **15 minutes** – Explaining the further work, i.e., the content of the homework.

If conducting the lab takes less time than planned, the remainder of the lesson can be used to start on the homework tasks.

3.5 Grading

The maximum amount of points for this lab is 9, divided between the two tasks (see Table 2). Task 1 is worth 4 points since it is slightly easier than Task 2. Task 2 is worth 5 points. The grading criteria for the tasks are as follows:

Table 2. Grading breakdown for Lab 7 tasks.

Task	Max Points	Objective
T1.1	2	Correctly identifying four additional metamorphic relations and writing down the corresponding mathematical functions.
T1.2	1	Developing a test suite, with 0.25 points for each test that successfully reveals a bug.
T1.3	0.5	Creating a table reporting the identified bugs.
T1.4	0.5	Choosing the best calculator and providing a reasonable explanation for the selection.
T2.1	3	Modifying the dataset with five metamorphic relations, providing sample images of the altered data, and explaining the reasoning behind each modification.
T2.2	1	Submitting the result tables.
T2.3	1	Analysing the results and correctly assigning the models to their respective markets.

The grading structure rewards technical implementation, critical thinking, and examination of results. The highest weight is given to defining and explaining metamorphic relations, as the

main goal of the tasks is to understand how metamorphic testing works in practice. Furthermore, the grading encourages students to document and analyse their findings. This is especially necessary for Task 2 since it has many valid ways of solving it, allowing students to get points even when their work and results differ from the sample solution.

With the lab tasks, materials, and grading established, the following chapter reports how the lab was executed.

4. Lab Execution

This chapter describes the execution of the metamorphic testing lab sessions conducted on April 8 and 9, 2025, at the University of Tartu's Delta Centre. Each session was 90 minutes long, and students were divided into six groups based on the schedule. This chapter first covers the observed sessions (Groups 1, 3, and 5), followed by a summary of unobserved sessions based on TA feedback.

4.1 Observed Lab Sessions

The first session was attended by 10 students. It began with an introduction to metamorphic testing, a walkthrough of the lab materials, and an explanation of the two main tasks. Students were then guided through the setup process for both assignments. The most common setup issues were related to incompatible software versions. Many students still had Java 11 installed from the previous lab, while Java 19 or newer was required. Although this requirement was specified in the setup guide, several students found it difficult to install the correct version. For the second task, some students did not have Python installed, and others had Python version 3.13, which is not supported by TensorFlow and led to errors. These problems were resolved during the session, and by the end, all students had completed the setup. Many began working on the assignment, and a few nearly completed Task 1 during the session.

The Group 3 session had 17 students in attendance. This group did not experience significant issues when setting up Task 1, but Python-related issues similar to those experienced by Group 1 appeared during the setup for Task 2. Overall, the setup was quicker compared to the earlier session, since the solutions to problems from the previous lab were known. Students moved efficiently through Task 1 and generally found it simple and intuitive; they found most MRs during the lab session. Task 2 was perceived as more difficult, requiring more abstract thinking and experimentation with image manipulation. The TA provided hints to help clarify the goals of the second task. Several students started making the expected modifications to the test data, demonstrating an understanding of the goal of the task. Some students started modifying the "Testing.py" file in other ways to understand the code better, which was not necessary, but displayed an interest in the task.

After the session, the lab materials were revised to clarify the problem with Python versions encountered in Task 2. A message was also sent to both instructors and students to explain how

to resolve the setup issues, particularly the incompatibility between TensorFlow and Python 3.13, and the requirement for Java 19 or newer.

The Group 5 session on April 9th had 10 students. Students were encouraged to read the instructions beforehand, unlike previous labs, where the students exclusively relied on the TA's slides. The TA demonstrated the setup process for Task 1. As a result, there were almost no issues with the Java setup. Task 2 presented similar Python-related problems as before, mainly missing installations or incompatible versions, which were again solved in the lab. A few students expressed confusion about the F1 score column in the reporting table found in the homework instructions, Appendix B. This element could be removed from the instructions in future iterations to prevent misunderstandings. Additionally, one student asked whether darkening and brightening the images constituted the same metamorphic relation. Since they should count as separate ones, the task description or lab instructors could clarify this more explicitly.

4.2 Unobserved Lab Sessions

Feedback was also collected from the TAs who supervised the unobserved lab groups. According to the TA reports, Lab 7 ran smoothly without noteworthy issues.

The TAs generally agreed that the setup process was easier than previous labs, where students struggled significantly. Although Lab 7 introduced Python into the Software Testing course for the first time, students successfully followed the instructions to create a virtual environment and install the required packages. None of the students faced any significant setup problems, and they were able to complete the setup during the lab session and immediately started solving the tasks. One TA noted that Lab 7's format differed from previous labs, and the introduction of an AI-related task improved its appeal to students. They showed interest in knowing how AI models work and the approaches for evaluating a pre-trained model. The TA claimed that Lab 7 was their favourite out of all the labs offered that semester.

4.3 Summary

Overall, the sessions were effective in introducing students to the principles of metamorphic testing. Despite initial technical difficulties, most students were able to begin or even complete Task 1 during the lab. Many of the problems encountered were recurring and could have been prevented with more explicit setup instructions. In the observed labs, students did not leave before the end of the session, staying to get help with the setup or to continue working on their

homework tasks. In both the observed and unobserved sessions, students actively participated in both tasks, which suggests that the lab successfully engaged the students.

5. Student Feedback

This chapter provides an overview of the feedback collected from students after the homework deadline. The section explains how the feedback was collected, presents and analyses the results, and provides a summary of implemented and future improvements.

5.1 Collecting the Feedback

The following week, after the homework deadline passed, an anonymous feedback form was sent out to the students. The lab instructors gave the students a few minutes to complete the form at the beginning of the labs. However, students who did not attend the lab still had the opportunity to fill it out independently.

The feedback form consisted of 10 questions. Several were standard questions reused from previous years to evaluate the suitability of the materials, while others were designed specifically for this lab. The complete list of questions can be found in Appendix III.

Most of the questions were written as statements, with students asked to respond using a four-point Likert scale with the options *agree*, *somewhat agree*, *somewhat disagree*, or *disagree* to avoid neutral answers. This approach was chosen because it can lead to more informative feedback, as it reduces the number of ambiguous answers that are difficult to interpret in analysis. In addition, the form included one question about the amount of time it took to complete the homework, one about the difficulty of the lab and an open-ended question requesting any additional feedback, with example prompts to guide responses. Furthermore, to support the thesis goal of covering more AI-related content in the course, one question asked if students believe that testing AI should be a larger part of the course, to understand their interest in this topic.

5.2 Feedback Results

In total, 37 students completed the questionnaire. The responses to the quantitative and qualitative feedback questions will be analysed separately.

5.2.1 Quantitative Results

Figure 3 illustrates a summary of responses to all agree/disagree questions. Overall, 95% of respondents either agreed or somewhat agreed that the lab and homework goals were clearly communicated. Similarly, most students agreed that the instructions were clear and easy to follow, with only four people disagreeing. The technical setup process was reported as the

most challenging aspect of the lab, for which 10 responders disagreed with the statement that the setup process was easy. After completing the lab, students generally found the concept of metamorphic testing understandable, with 92% of respondents selecting agree or somewhat agree and no students selecting disagree. Most students agreed when asked whether the lab was relevant for work in the software industry, while only 2 students disagreed or somewhat disagreed.

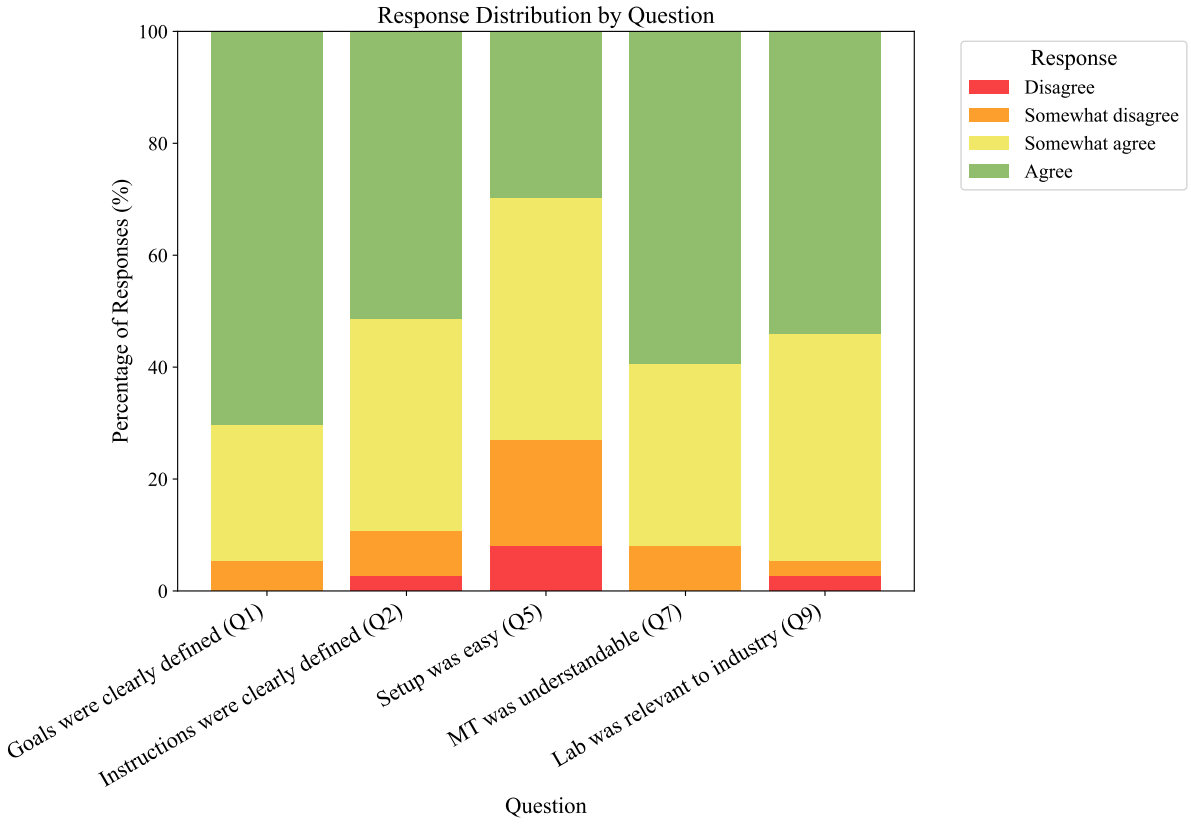


Figure 3. Response distribution for agree/disagree questions.

Regarding difficulty, 51% of students (see Figure 4) found the lab to be about the same level of difficulty as previous assignments. The remaining responses were evenly split between those who thought it was harder and those who thought it was easier than the previous homework assignments.

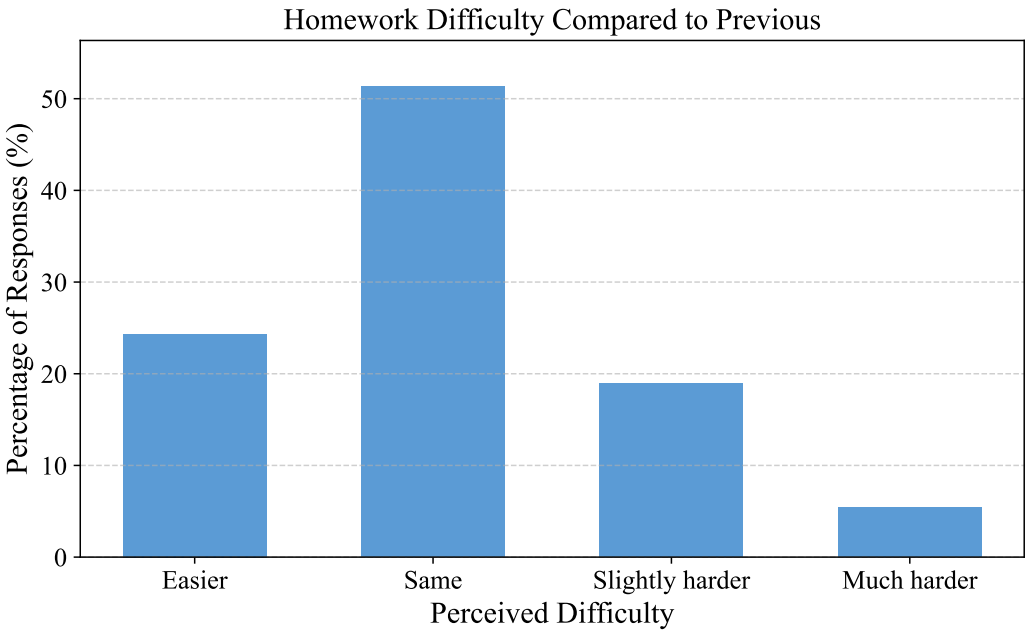


Figure 4. Perceived difficulty compared to previous assignments.

The time required to complete the lab varied slightly (see Figure 5). The majority reported completing it within 2 to 5 hours, which is the preferred time, since completing the homework in pairs should take around 8 hours. The second most common selection was 5-10 hours; only a few people took less than two or more than 10 hours to complete the homework.

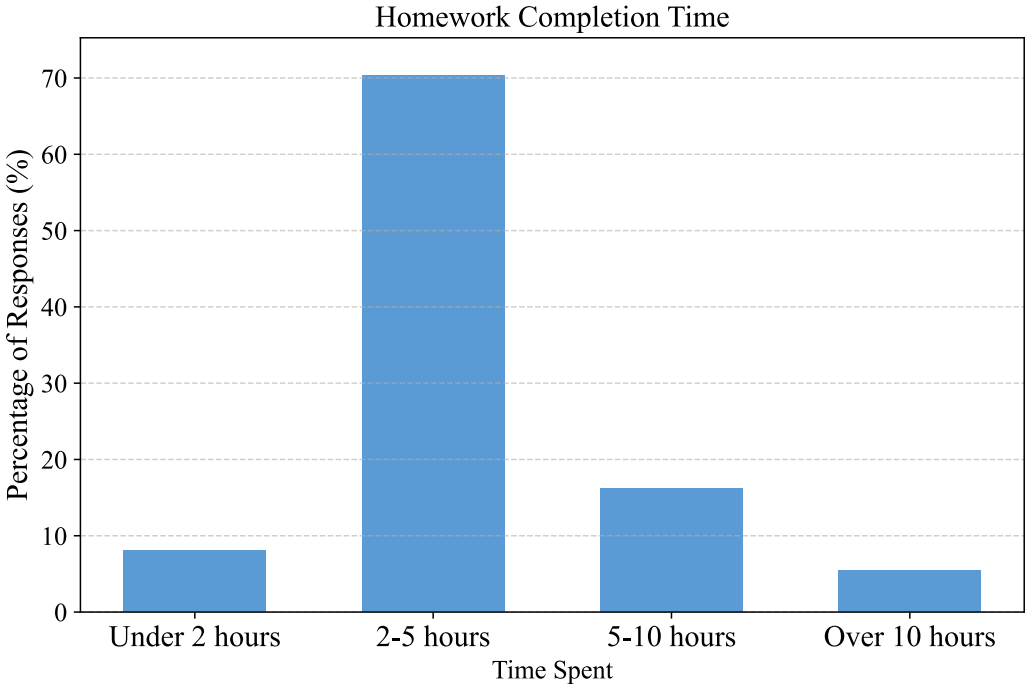


Figure 5. Time spent completing the homework.

Lastly, a majority (70%) thought that testing AI and machine learning systems should have a greater role in the Software Testing course, which aligns with the thesis goal of increasing the inclusion of these topics.

5.2.2 Qualitative Results

In addition to scaled responses, students were asked to provide open-ended feedback through two questions.

The first of these questions was to specify what was unclear about the lab instructions if the respondent selected ”disagree” or ”somewhat disagree” to Q2. While only four people disagreed with the statement, 16 answered the question. There were eight options to choose from, as well as the possibility to write in their own option. Most of the complaints were about the environment setup being complicated or the technical instructions being lacking. Four students mentioned that the instructions were hard to navigate, and a few mentioned that there was too much information at once.

The second qualitative question was fully open-ended to gather any additional feedback students had about the lab. A few guiding questions were provided to help students think of comments to add. These included prompts such as what they found most challenging, what could be improved in the lab materials or structure, and whether they would recommend keeping the lab in future versions of the course. Most of the responses were to one or multiple of these prompts.

5.3 Feedback Analysis

The quantitative feedback responses show that students generally understood the lab’s objectives and tasks, meaning that the learning outcomes were communicated effectively. The balance of perceived difficulty and the distribution of time spent on the homework suggest that the lab was appropriately planned both in terms of complexity and duration. The even split between students who found the lab easier or harder than previous assignments is likely due to the difference in overall course performance and academic ability rather than a flaw in the task design. The data also reinforces the relevance of the lab’s focus area. This feedback confirms that students are interested in learning about techniques relevant to AI-based systems, supporting the value of including metamorphic testing in the course.

The feedback from the open-ended question was very positive. A total of 13 students explicitly mentioned that the lab should be kept in future versions of the course for different reasons, such as the increasing relevance of learning about AI testing and the homework providing a clear

understanding of metamorphic testing. Many students mentioned that the instructions were clearer than in the previous labs, and one noted that this lab felt the closest to testing real-world applications instead of toy programs. A prominent theme in the responses was that the lab was more fun and interesting when compared to the previous labs.

Despite the overall positive feedback, some suggestions for improvement were brought up. The setup challenges identified in the results stand out as the main issue. While not universal, the number of students who reported difficulties indicates that this aspect requires improvement. The main criticisms were about the setup instructions, primarily concerning the second task. Some students suggested a troubleshooting section in the instructions to remedy common setup issues. Additionally, five respondents pointed out confusion regarding the F1 score column in the report table. It was unclear whether they were expected to calculate these scores themselves. Overall, there was not much confusion about actually solving the tasks, which suggests that the idea of the lab was clear to most of the students.

Nonetheless, some limitations on the feedback exist. Only 37 out of roughly 150 active students completed the survey, meaning the results are based on a small, self-selected sample. This might over-represent motivated students, who likely did better on the homework. In the future, the feedback could be made mandatory or incentivised with bonus points to increase the response rates. However, it is still valuable, since the quantitative responses outlined a rough trend of the lab being positively received, and the qualitative responses gave valuable ideas for improvements for future iterations.

5.4 Immediate Improvements

Based on the feedback and observations made in the labs, a few refinements were made to the lab instructions. The changes are visible in Appendix IV in Table 3 with modified/added text marked in blue. For the first task, the setup instructions were revised to explicitly mention that students should check that they do not have an outdated Java version configured from previous labs.

However, the main improvements were made to the second task, which had the most problems. Firstly, the setup instructions were made more detailed and structured. The improved instructions now mention that Python must be installed. Version compatibility is emphasised, and a direct download link to a compatible Python version is provided. Secondly, a troubleshooting section was added to the end of the instructions, listing some common errors that could be encountered

while setting up the lab and the solutions to fix them. Lastly, the F1 column from the model report table in Appendix B was removed to avoid confusion during the homework task.

The updated instructions and lab format will continue to be used in next year's course. However, it is recommended that the setup process and dependencies be checked annually so that all Python libraries and frameworks remain compatible.

5.5 Future Developments

Although this lab has successfully introduced students to metamorphic testing through practical tasks, a few limitations can be addressed by developing this lab further.

Firstly, Task 1, which focused on braking distance calculators, was designed to be relatively straightforward. While this simplicity helped students focus on understanding the basic principles of metamorphic testing, it also limited the number of possible metamorphic relations they could discover, since there could only be one MR per parameter. Future lab versions could involve a more complex application, allowing students to identify more metamorphic relations and make the exercise longer. A possibility would be to extend the task into multiple calculators, for example, combining braking distance calculations with a crash risk analyser based on several additional parameters. This would make the first task more challenging and help students practice defining metamorphic relations.

Similarly, the models in Task 2 were intentionally kept simple to ensure that they were functional and reliable to use in the homework. As a result, the machine learning models do not fully reflect the complexity commonly encountered in real-world AI testing scenarios. To make the task richer and more realistic, an object detection algorithm could be trained to locate and classify a traffic light in an image, which makes it a more complex algorithm with a larger number of possible MRs. Furthermore, in the future, this lab structure could be adapted to create differently themed labs focused on various types of AI systems. For example, a future version could involve a language model that performs sentiment analysis on customer reviews, where students could define metamorphic relations such as paraphrasing or synonym substitution.

Such developments would not only add to the lab's educational value but also ensure its relevance as AI systems and testing become more complicated.

6. Conclusion

This thesis presents the development of a new lab package designed to introduce students to the concept and applications of metamorphic testing. It addresses the growing need to expose students to testing methods specifically suited for complex, nondeterministic systems, primarily due to the growing role of artificial intelligence and machine learning.

To achieve this goal, a lab package was designed and implemented for the University of Tartu Software Testing course. The lab tasks covered the main use cases of metamorphic testing to provide students with practical experience in applying metamorphic relations where traditional oracles are unavailable. Two tasks were developed: testing braking distance calculators and evaluating traffic sign classifiers. In addition to creating realistic tasks and comprehensive materials, the effectiveness of the lab was assessed through student feedback collected after the session.

The feedback results indicated that metamorphic testing can be taught at the undergraduate level in an understandable and engaging way for the students. The lab sessions were conducted successfully, with the students participating actively in both tasks. Despite initial technical challenges, especially regarding the setup of Python environments, these issues were resolved during the sessions, and improvements were made to the materials based on observations and feedback. Feedback collected after the homework deadline showed strong student support for the lab. Most students found the tasks understandable and relevant to industry, and many expressed that AI-related testing topics should play a greater role in the course. Student interest and feedback, together with instructors' comments, showed that the lab was successful and should be kept in future versions of this course.

Overall, this thesis improves software testing education by integrating AI-relevant testing techniques into a core undergraduate course. The results showed that practical assignments can effectively teach complex concepts such as metamorphic testing. Future development of the lab could be expanded to include a broader range of AI systems, providing students with more realistic testing challenges.

References

[1] Zheng Z., Ren D., Liu H., and Chen T. Y. Metamorphic fault tolerance: addressing the oracle problem of reliability assurance for contemporary software systems. *Computer* 57.7 (July 2024), pp. 77–86. DOI: 10.1109/MC.2024.3390759.

[2] Chen T. Y., Cheung S.-C., and Yiu S.-M. Metamorphic testing: a new approach for generating next test cases. *arXiv* 2002.12543 (2020). DOI: 10.48550/arXiv.2002.12543.

[3] University of Tartu, Institute of Computer Science. SIS Software Testing (6 EAP) LTAT.05.006 course description. 2024. https://ois2.ut.ee/#/courses/LTAT.05.006/version/8e67f0aa-e4ae-fa4f-8b72-37e4c0fd608f/details (12/07/2024).

[4] Barr E. T., Harman M., McMinn P., Shahbaz M., and Yoo S. The oracle problem in software testing: a survey. *IEEE Transactions on Software Engineering* 41.5 (May 2015), pp. 507–525. DOI: 10.1109/TSE.2014.2372785.

[5] Segura S., Towey D., Zhou Z., and Chen T. Y. Metamorphic testing: testing the untestable. *IEEE Software* 37.3 (May 2020), pp. 46–53. DOI: 10.1109/MS.2018.2875968.

[6] Lin X., Simon M., Niu N., Carver J., and Rouson D. Exploratory metamorphic testing for scientific software. *Computing in Science Engineering* 22.2 (Mar. 2020), pp. 78–87. DOI: 10.1109/MCSE.2018.2880577.

[7] Saha P. and Kanewala U. Fault-detection effectiveness of metamorphic relations developed for testing supervised classifiers. *2019 IEEE International Conference on Artificial Intelligence Testing (AITest)*. San Francisco, CA: IEEE, Apr. 2019, pp. 157–164. DOI: 10.1109/AITest.2019.00019.

[8] Glossary of Machine Learning Terms. OpenTrain AI. 2024. https://opentrain.ai/glossary (04/10/2025).

[9] Cooper A. F., Frankle J., and De Sa C. Non-Determinism and the Lawlessness of Machine Learning Code (2022). DOI: 10.1145/3511265.3550446.

[10] 50.053 Software Testing and Verification. Online resource. Singapore University of Technology and Design. 2025. https://www.sutd.edu.sg/course/50-053-software-testing-and-verification/ (04/12/2025).

[11] Computer Science (CS) Graduate Courses — Academic Catalog 2025–26. Idaho State University. 2025. https://coursecat.isu.edu/graduate/allcourses/cs/ (03/06/2025).

[12] Stahl M. Metamorphic Testing. StickyMinds. 2021. https://www.stickyminds.com/article/metamorphic-testing (03/06/2025).

[13] Segura S., Fraser G., Sánchez A. B., and Ruiz-Cortés A. A Survey on Metamorphic Testing. *IEEE Transactions on Software Engineering* 42 (Sept. 2016), pp. 1–1. DOI: 10.1109/TSE.2016.2532875.

[14] Chen T. Y., Kuo F.-C., Liu H., Poon P.-L., Towey D., Tse T. H., and Zhou Z. Q. Metamorphic testing: a review of challenges and opportunities. *ACM Computing Surveys* 51.1 (Jan. 2019), pp. 1–27. DOI: 10.1145/3143561.

[15] Sharma N., Jain V., and Mishra A. An analysis of convolutional neural networks for image classification. *Procedia Computer Science* 132 (2018), pp. 377–384. DOI: 10.1016/j.procs.2018.05.198.

Appendices

I GitHub Repository

The lab materials and code developed as part of this thesis are available in the following GitHub repository: https://github.com/EliisabetK/Thesis-Materials. The repository is private, and access must be requested from the author.

Repository overview:

- Lab Materials/: Materials used in the original lab session on April 8-9:
  - Student and TA instructions
  - Lab session slides
  - ZIP archive used during the session
- Improved Materials/: Updated student and TA instructions based on the feedback collected after the lab, intended for future sessions.
- Task1/ — Braking Distance Calculator (Java):
  - src/main/java/: Braking distance implementations, CLI, interface
  - src/test/java/: Test suite template and completed test suite for reference
- Task2/ — Traffic Sign Classifier (Python):
  - data/: Training and test datasets, label files
  - models/: Pre-trained classifiers
  - testing/:
    - MT_testing.py: Completed metamorphic test suite
    - students_file.py: Template test file for students
    - testing.py: F1 score calculations
  - training/: Training scripts for all four models, each labeled accordingly

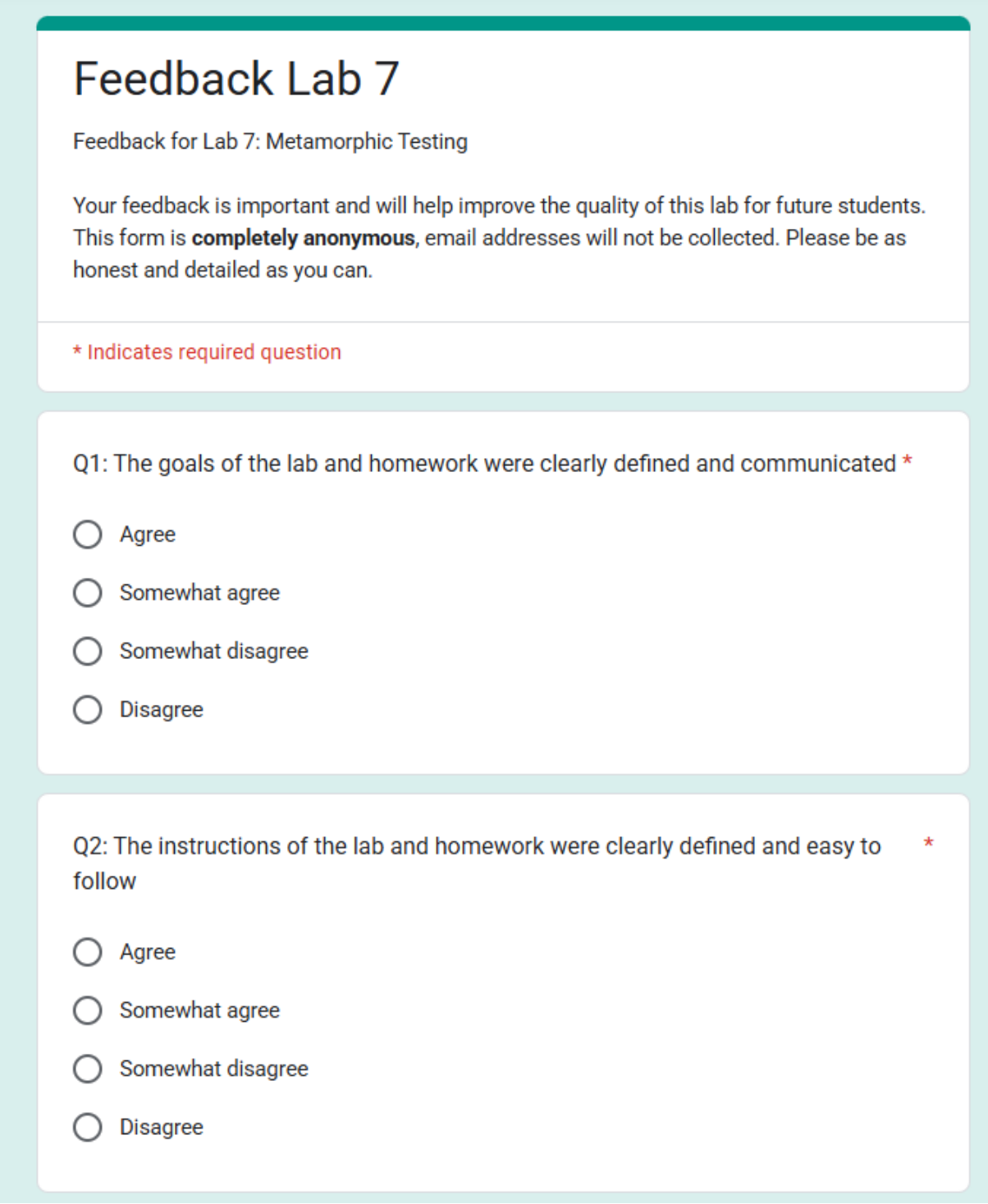
II Student Lab Materials

Lab materials as listed on the course site, including lab instructions and a ZIP file containing the task code:

- Lab 7 Instructions (PDF)
- Lab 7 SUT (ZIP)

III Feedback Form

Feedback form sent to the students after the homework deadline.



Feedback Lab 7

Feedback for Lab 7: Metamorphic Testing

Your feedback is important and will help improve the quality of this lab for future students. This form is **completely anonymous**, email addresses will not be collected. Please be as honest and detailed as you can.

* Indicates required question

Q1: The goals of the lab and homework were clearly defined and communicated *

- Agree
- Somewhat agree
- Somewhat disagree
- Disagree

Q2: The instructions of the lab and homework were clearly defined and easy to follow *

- Agree
- Somewhat agree
- Somewhat disagree
- Disagree

Figure 6. Student feedback questionnaire used after the lab session (page 1 of 4).

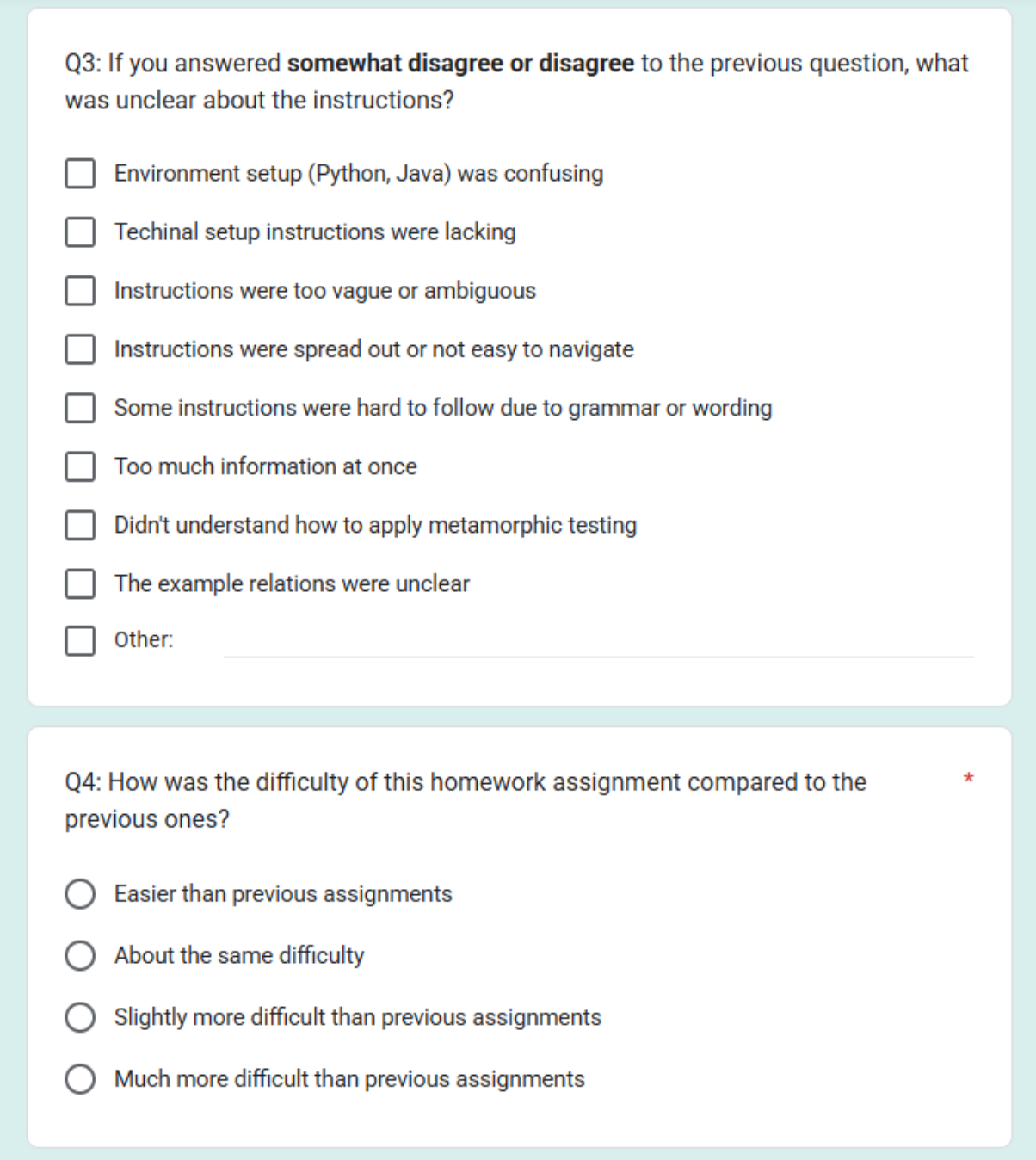


Figure 6. Student feedback questionnaire used after the lab session (page 2 of 4).

Q5: The technical setup process was easy *

- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree

Q6: How long did completing the homework take you? *

- ○ Under 2 hours
- ○ 2-5 hours
- ○ 5-10 hours
- ○ Over 10 hours

Q7: The concept of metamorphic testing was understandable after completing this lab *

- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree

Figure 6. Student feedback questionnaire (page 3 of 4).

Q8: Do you think testing AI/ML systems should be a larger part of the Software Testing course? *

- ◯ Yes
- ◯ No

Q9: Overall, what I learned in the lab is relevant for working in the software industry *

- ◯ Agree
- ◯ Somewhat agree
- ◯ Somewhat disagree
- ◯ Disagree

Q10: Additional feedback: *

Please feel free to share any feedback you have about this lab. You can answer one or more of the following questions, or comment on anything else you found important:

- What was the most difficult part of this lab?

- What would you improve in the lab structure, materials, or tasks?

- Would you recommend keeping this lab in future versions of the course? Why or why not?

Your answer

Figure 6. Student feedback questionnaire (page 4 of 4).

IV Improved Instructions

Table 3. Improvements made to the lab instructions based on student feedback.

Change	Page	Original Text	Updated Text
1	3	"Select JDK-21, however, it should work with any JDK version 19 or newer."	"Select JDK-21, however, it should work with any JDK version 19 or newer. Make sure you don't have an older Java version enabled from previous labs."
2	5	"...you should use Python version 3.11."	"...you must use Python version 3.11 or 3.12. Python 3.13 will not work due to TensorFlow incompatibility."
3	5	"You can check your Python version with 'python –version'."	"You can check your Python version with 'python –version'. If you do not have Python 3.11 installed, download it from the official Python website…"
4	5	"Open the terminal in the IDE and create a virtual environment with the command … and activate it…"	"Recommended but optional: a. Open the terminal in the IDE and create a virtual environment with the command `py -3.11 -m venv env`; b. Activate the virtual environment with `.\env\Scripts\activate`. After activation, you should see (env); c. Make sure you run all files in the terminal that shows (env)."
5	5	"Run the file 'python setup.py'. This ensures that all required libraries are installed…"	Extended with warning: "This file might run for a while; wait until it says 'Setup complete'."

Change	Page	Original Text	Updated Text
6	5	No troubleshooting comment provided.	"If any installation fails, manually install the missing packages using: pip install package-name."
7	5	"The code will display the original and modified images and print out the consistency score table."	"If the program displays the original and modified images ..., then the setup is complete."
8	14	No troubleshooting section.	New section exists in the improved version with detailed solutions for Python setup errors, PATH, permissions, long paths, etc.

Licence

Non-exclusive Licence to Reproduce the Thesis and Make the Thesis Public

I, **Eliisabet Kaasik**,

1. grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu until the expiry of the term of copyright, my thesis

Software Testing Lab Package: Metamorphic Testing
supervised by **Dietmar Pfahl**;

2. grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence **CC BY-NC-ND 4.0**, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright;

3. am aware of the fact that the author retains the rights specified in points 1 and 2;

4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Eliisabet Kaasik
(13/05/2025)