Stichting NIOC en de NIOC kennisbank

Stichting NIOC (www.nioc.nl) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website www.nioc.nl ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2025, gehouden op donderdag 27 maart 2025 jl. en georganiseerd door Hogeschool Windesheim). Bij elkaar zo'n 1500 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

Het eerstvolgende NIOC vindt plaats in 2027 en wordt dan georganiseerd door HAN University of Applied Sciences. Zodra daarover meer informatie beschikbaar is, is deze hier te vinden.

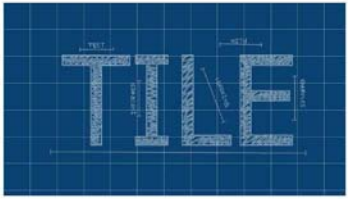
Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

www.nioc.nl/nioc-kennisbank/aanmelden nieuwsbrief

Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.



Test Informed Learning with Examples (TILE)

Set the right example when teaching programming

NIOC 2023: Onstuimige vragen, creatieve oplossingen?

Niels Doorn

Open Universiteit, NHL Stenden

niels.doorn@{ou.nl, nhlstenden.com} research.nielsdoorn.nl 0000-0002-0680-4443

nielsdoorn @niels76@mastodon.online

We would love it if our IT graduates would become so called ‘Test Obsessed’

Current situation

Software testing is an important skill for software engineers

- **Little attention** is given to it
- Integrating software testing in **early** programming courses is **beneficial**
- There are **drawbacks** to integrating testing in programming courses

Question

How can we improve teaching / learning strategies for software testing?

The TILE Approach

Test Informed Learning with Examples (TILE) is a way to integrate testing in education:

- **Early**: from the very first programming exercise on
- **Seamless**: a smooth and continuous way as an inherent part of programming
- **Subtle**: clever and indirect methods to teach testing

Types of TILE

We developed four different types of TILEs:

- Test **run** TILEs
- Test **cases** TILEs
- Test **message** TILEs
- Test **domain** TILEs

Test run TILEs

We can ask the students to **test** the program instead of asking them to **run** the program

Example of a test run TILE

Consider the following program:

```
n = int(input("Enter a number: "))
square = n * n
print("The square is: ", square)
```

Compare the wording of the following two ways:

1. Now let us **run** this program, the user can give input through the keyboard and the results will be shown on the screen
2. Now let us **test** this program by running it and **entering test input data** through the keyboard and **checking the resulting output** on the screen

Test case TILEs

Students often only test **happy path** execution

We can add **add more concrete examples of possible test cases** to create awareness of other useful test cases

Test case TILEs

TILE-ing exercises this way can be done in different ways

For example:

1. Adding **example test executions**
2. Adding **example test cases**
3. Forcing students to think about test cases explaining needed **combinations and boundary values**
4. Point the students to a **parallel oracle**

Example of a test case TILE: adding example runs

Consider this exercise:

Exercise: *Implement a program that reads an integer corresponding to a month of the year and displays the name of the corresponding month. An error message should be displayed if the entered number does not belong to the range [1, 12].*

```
>>> %Run
Enter the number of the month: 5
May
>>> %Run
Enter the number of the month: 1
January
>>> %Run
Enter the number of the month: 13
Error: enter a number between 1 and 12
>>> %Run
Enter the number of the month: 0
Error: enter a number between 1 and 12
>>> %Run
Enter the number of the month: -3
Error: enter a number between 1 and 12
```

Example of a test case TILE: presenting test cases

Exercise: *Implement a program that asks the user for a comparison operator:* `<`, `<=`, `>`, `>=`, `==`, `!=` *and 2 values. Your program has to display on screen the result* (`True` or `False`) *of the given operation applied to the two values.*

test id	test inputs			expected output
	operator	value1	value2	
1	<	12	4	False
2	>	100	40	True
3	==	"Hello!"	40	False
4	!=	100	"Python"	True
5	>=	98.67	0.45	True
6	<=	-100	40	True
7	<	24	"24K"	True
8	>=	"email"	"correo"	True

Example of a test case TILE: forcing students to think about test cases

Exercise: *Write a program that receives as input an amount of euros, and displays as output the minimum breakdown in bills and coins for that amount on the screen. We assume that there are 500, 200, 100, 50, 20, 10 and 5 bills, and 2 and 1 coins.*

Then, students are asked to think about other tests they should run to ensure that the program has the desired behaviour

- Did they run a test with the amount 0 and with a negative amount?
- Did they make sure to choose different amounts in such a way that each bill and coin was returned at least once?
- Did they choose different amounts to test whether the program returns the minimum number of correct combinations of bills and coins?

Example of a test case TILE: forcing students to think about test cases

Exercise: *Write a program that receives as input an amount of euros, and displays as output the minimum breakdown in bills and coins for that amount on the screen. We assume that there are 500, 200, 100, 50, 20, 10 and 5 bills, and 2 and 1 coins.*

For example:

- for 2, one coin of 2 must be returned (and not two of 1)
- for 10, one bill of 10 must be returned (and not two of 5)
- for 6, a bill of 5 and a coin of 1 must be returned (and not for example 3 coins of 2)
- for amount 12, a bill of 10 and a coin of 2 must be returned

Test Message TILEs

TILEs of this type hide **a subliminal message** about the importance of testing.

Example of a test message TILE

Exercise: Write a program that asks the user for something important and returns a billboard ASCII art.

```
>>> %Run
Something important: Testing your code

                \|||||/
                ( O O )
|---------ooO-----(_)--------------|
|                                  |
|  Testing your code is important! |
|                                  |
|------------------------Ooo-------|
                |_||_|
                ||  ||
               ooO  Ooo
```

Example of a test message TILE

MadLibs is a word game where a player asks others for a list of words to substitute for blank spaces in a story, before reading the story aloud

MadLibs can be turned in all sorts of message TILEs, e.g.:

Exercise: *Consider the following little MadLibs.*

Whether you ..... *(verb)* computer programs to solve ..... *(plural noun)*

or just for ..... *(noun)*. It is very important that you ALWAYS TEST

your code for ..... *(adejctive)* bugs.

Example of a test message TILE

MadLibs can be turned in all sorts of message TILEs, e.g.:

Exercise: *Consider the following little MadLibs.*

We need to ask the following words:

- verb, for example: `write`
- plural noun, for example: `problems`
- noun, for example: `fun`
- adjective, for example: `nasty`

and use these words to fill in the placeholders in the figure. Try any combination of words, and when your program returns the result, read it out loud

Domain TILEs

Exercise: Create a program that guides the user through the process of figuring out the type of fruit on hand. Use the following decision tree to build your program.

Domain TILEs

Exercise: Create a program that guides the user through the process of figuring out the type of fruit on hand. Use the following decision tree to build your program.

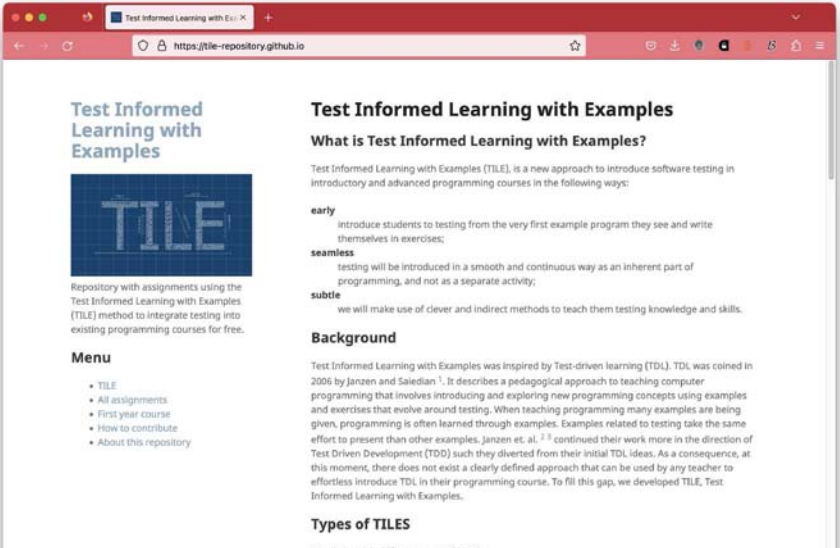
```
>>> %Run
  Color (green/yellow/red): green
  Size (big/medium/small): big
  watermelon

>>> %Run
  Color (green/yellow/red): yellow
  Shape (round/thin): round
  Size (big/small): big
  grapefruit
```

Example of a domain TILE

Exercise: Imagine Ana wrote a program that sorts a list of numbers. Evidently, this needs to be tested. Create a program that guides her through the process of deciding whether she has tested sufficiently as shown below. Try to extend the program with possible cases that test other important things (at least two are useful to add).

```
>>> %Run
  Hello! I'm gonna help you improve your
  sorting program!

  Did you check a basic case like:
  [3, 1, 8] is sorted into [1, 3, 8]? (y/n) y

  Excellent!

  Did you check what happens when the list
  is empty? (y/n) y

  Nice.

  Did you check what happens for a list with a
  single element, like [3]? (y/n) y

  Well done!

  Did you verify it also works with negative
  numbers, like [4, -8, 10]? (y/n) n

  You'd better try that right now!

  That was my last question!
  You took care of 75% of the cases. Well done!
```

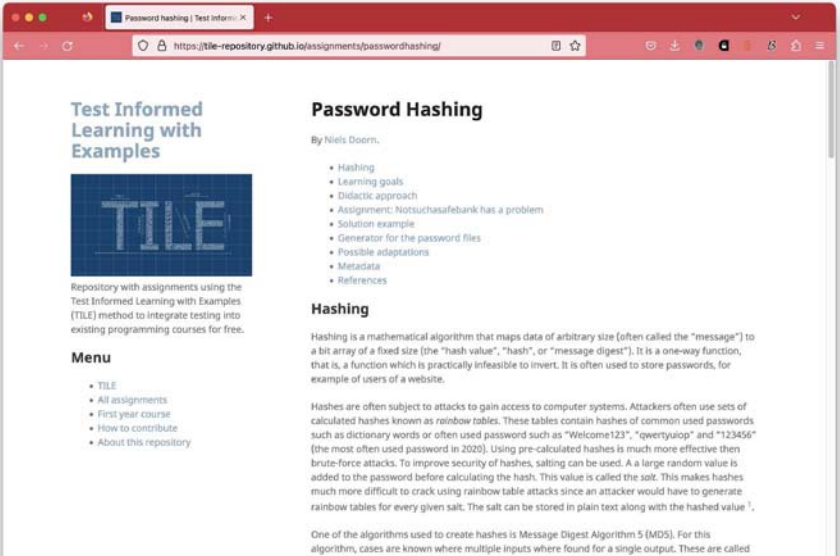
Repository with TILEd exercises



Repository with TILEd exercises

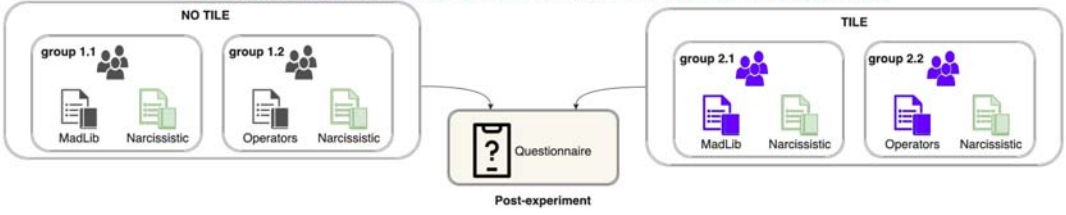
- We created an open GitHub repository
- Over 100 exercises (all types of TILEs)
- Both and exercises (but easy to transform)
- Each exercise has meta data to make it easier to fit them in existing courses
- We encourage lecturers to contribute!

Repository with TILEd exercises



Initial exploratory study

How does TILE contribute to the test-awareness of students?



Initial exploratory study

We measured:

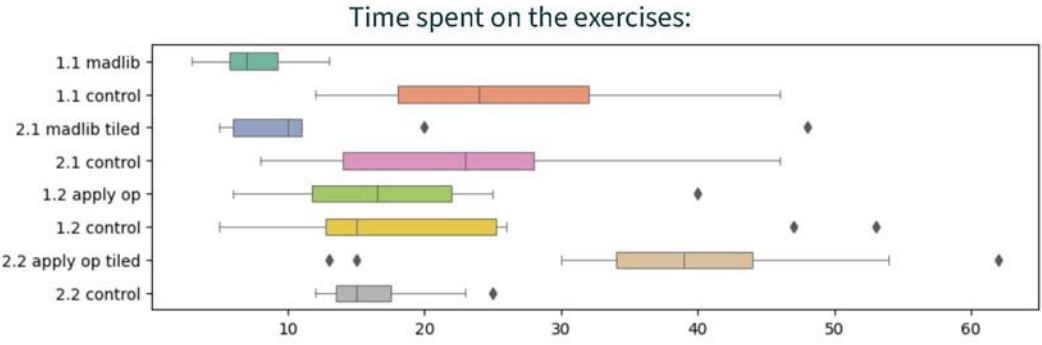
- the testing awareness of students, which is measured in number of testcases produced by students
- the time used to solve the exercises, measured in minutes
- the perception of the students about the confidence of their solution, measured in a 5-point Likert scale

Initial exploratory study

Number of Testcases and Confidence of subjects ($n = 50$):

Groups	First exerc. #testcases	Control exerc. #testcases	Average Confidence	Std.dev.
non-TILEd	5	18	4,06	1,19
TILEd	33	23	4,12	1,03

Initial exploratory experiment



Next steps in TILE

Our next steps in TILE will be:

- Linking the Computer Science Educational Research theories to TILE
- Improve the repository by making the meta-data more accessible
- Adding exercises to the repository

Next steps in Software Testing Education research

Our next steps in Software Testing Education research will be:

- Getting a better understanding of the sensemaking of students and expert practitioners while designing test cases
- Develop instructional designs to improve software testing in education, including
- with the use of serious games, and study the effects in different educational contexts

Summary

Test Informed Learning with Examples:

- TILE as a new concept for test-aware introductory programming courses
- It allows **early**, **seamless** integration in a **subtle** way
- There are test **run**, test **case**, test **message** and test **domain** tiles
- Our initial study shows an increase of test cases created by students

Thank you!

Read more about this research

edu.nl/utgcw

The repository can be found here

edu.nl/f9ptp

https://research.nielsdoorn.nl

Come join Martijn and me...

...at the Romanian Testing Conference