VU-Bugzoo

Bug Hunting Games to Educate Gen Z Software Testers

Natalia Silvis-Cividjian

VU VRIJE UNIVERSITEIT AMSTERDAM

TestDag-2025*Utrecht

VU BUG 01000110 ZOO

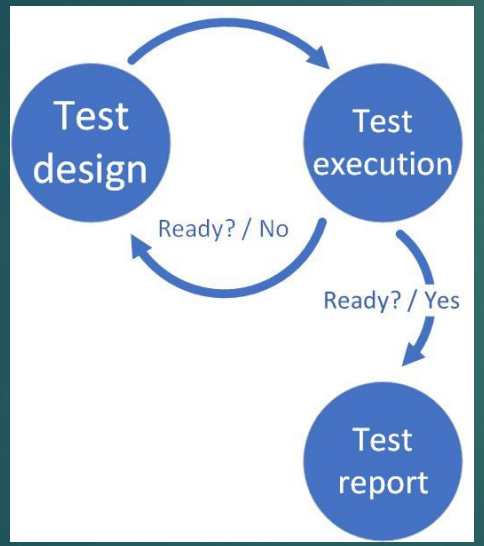
Outline

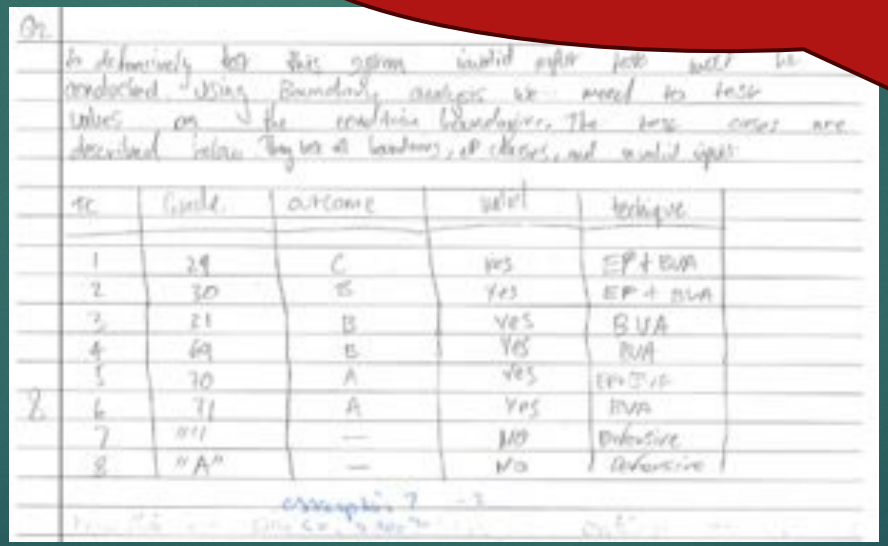
- The problem
- A solution
- A few studies
- Conclusions & future work

- **GenZ** was born between 1997 and 2012 and is considered the first generation to have largely grown up using the internet, modern technology and social media.
- Members of Gen Z are sometimes known as zoomers.

GenZ problems

- Old: Students lack the motivation to learn software testing





Tasks were repetitive and a little boring.

My software is the best, no need to test it

GenZ problems

- New: Generative AI, lack of comprehensive reading

Outline

- The problem
- A solution
- A few studies
- Conclusions & future work

A solution

Expose students to bugs and their effects. HOW?

- Sensitize students by showing them that reckless testing can kill people
- Engage students in bug-hunting games

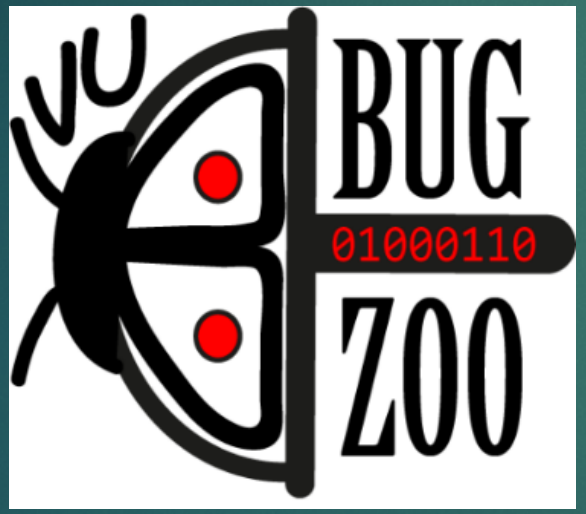
#1. Interviewing a witness of the Therac-25 accidents

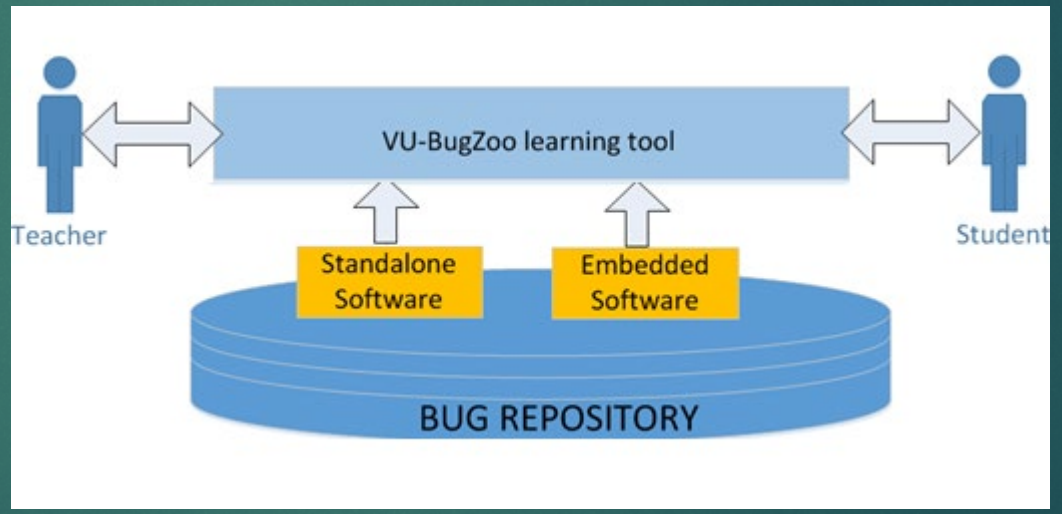
Silvis-Cividjian, Hager. Speak Memory! Analyzing Historical Accidents to Sensitize Software Testing Novices, *ICSE-SEET 2023.*

ComeniusNetwerk
de HogerOnderwijsVernieuwers

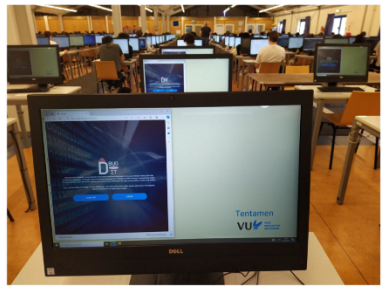
A solution

Engaging students in bug-hunting games for standalone and embedded software





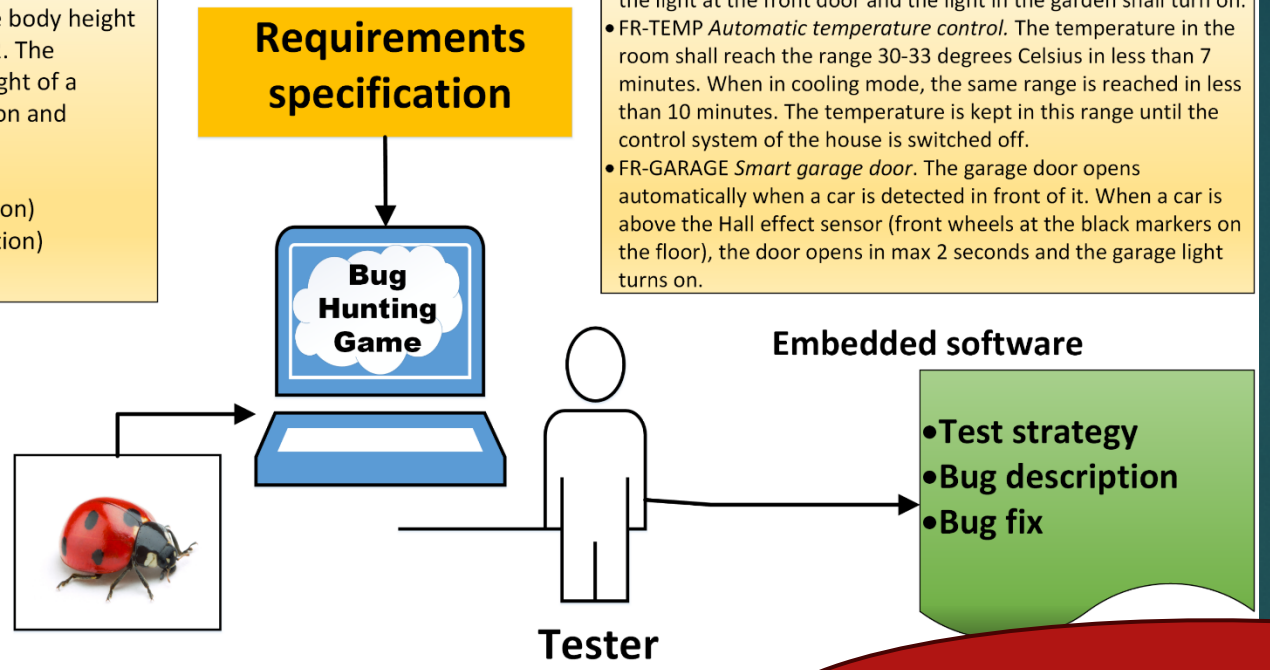
Silvis-Cividjian, N., Limburg, R., Althuisius, N., Apostolov, E., Bonev, V., Jansma, R., Visser, G., & Went, M. (2020). VU-BugZoo: A Persuasive Platform for Teaching Software Testing. In *ITiCSE 2020*



BMI
The body mass index (BMI) is a value derived from the mass (weight) and a height of a person. It is calculated as the body mass (in Kg) divided by the square of the body height (in meters), and is expressed in units of kg/m2. The application reads the weight in kg and the height of a person in meters using a decimal point notation and outputs the calculated BMI.

- Parameter 0: height (m, decimal point notation)
- Parameter 1: weight (kg, decimal point notation)
- Expected output: BMI (kg/m2)

Smart Home
- FR-LIGHT-1 *Smart lighting.* When the light intensity outdoors is lower than 27 lux, the curtains shall close and the lights indoors, the light at the front door and the light in the garden shall turn on.
- FR-TEMP *Automatic temperature control.* The temperature in the room shall reach the range 30-33 degrees Celsius in less than 7 minutes. When in cooling mode, the same range is reached in less than 10 minutes. The temperature is kept in this range until the control system of the house is switched off.
- FR-GARAGE *Smart garage door*. The garage door opens automatically when a car is detected in front of it. When a car is above the Hall effect sensor (front wheels at the black markers on the floor), the door opens in max 2 seconds and the garage light turns on.



DBugIT

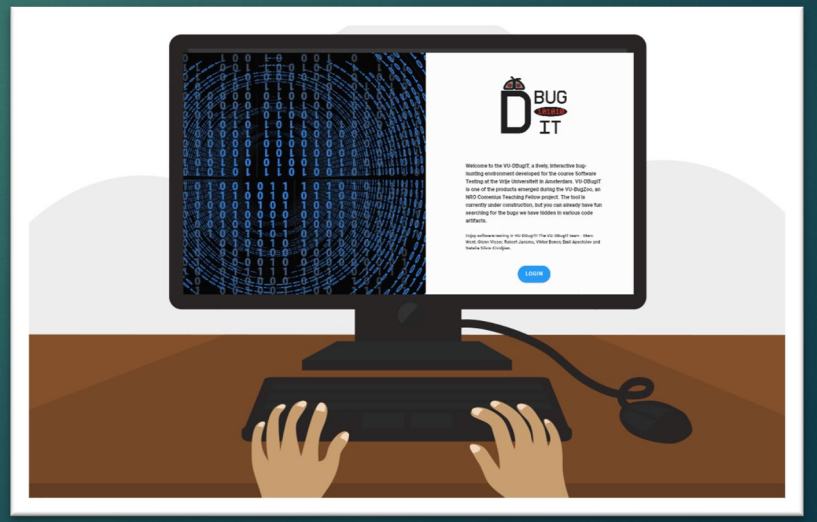
VU-SmartHome

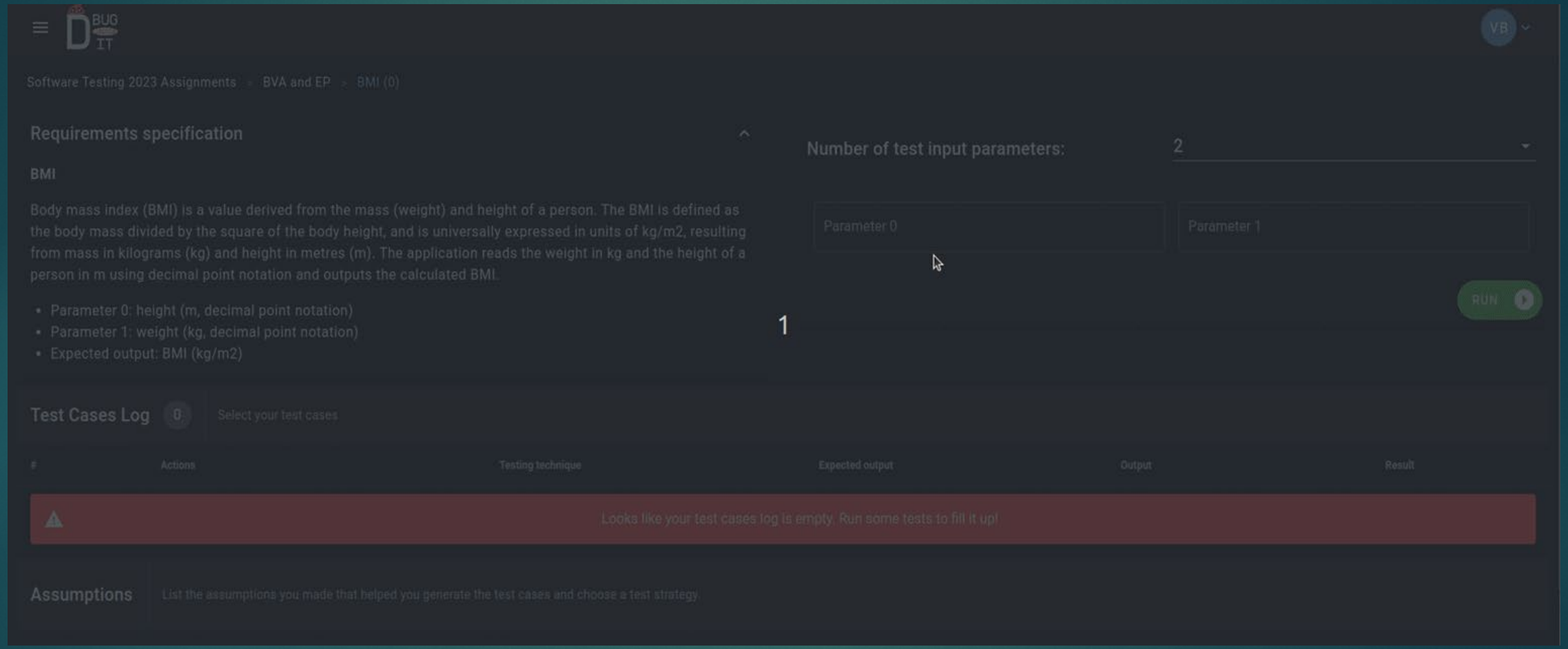
DBugIT

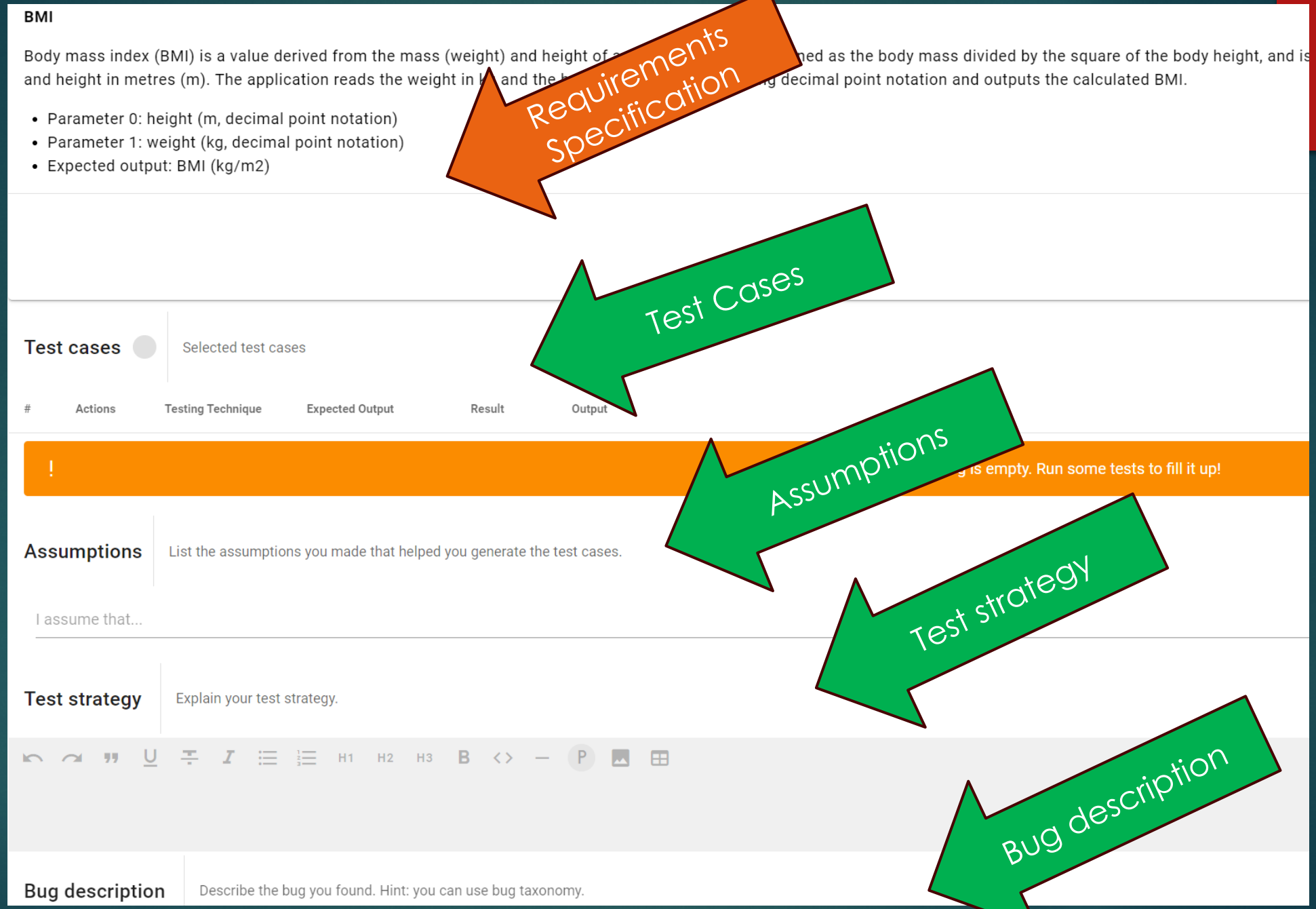
- Tests written on paper
- Hypothetical, correct code
- Pen and paper exams (impossible during COVID-19 lockdown)
- Complex and expensive tools

- Tests run on a computer
- Real, buggy code
- Remote online assessment
- Simple, affordable tool

VS.




BUG IT
VB
Software Testing 2023 Assignments > BVA and EP > BMI (0)
Requirements specification
BMI
Body mass index (BMI) is a value derived from the mass (weight) and height of a person. The BMI is defined as the body mass divided by the square of the body height, and is universally expressed in units of kg/m2, resulting from mass in kilograms (kg) and height in metres (m). The application reads the weight in kg and the height of a person in m using decimal point notation and outputs the calculated BMI.
Parameter 0: height (m, decimal point notation)
Parameter 1: weight (kg, decimal point notation)
Expected output: BMI (kg/m2)
Number of test input parameters:
2
Parameter 0
Parameter 1
RUN
1
Test Cases Log
0
Select your test cases
#
Actions
Testing technique
Expected output
Output
Result
Looks like your test cases log is empty. Run some tests to fill it up!
Assumptions
List the assumptions you made that helped you generate the test cases and choose a test strategy.


BMI
Body mass index (BMI) is a value derived from the mass (weight) and height of
ned as the body mass divided by the square of the body height, and is
and height in metres (m). The application reads the weight in k and the
g decimal point notation and outputs the calculated BMI.
Parameter 0: height (m, decimal point notation)
Parameter 1: weight (kg, decimal point notation)
Expected output: BMI (kg/m2)
Requirements Specification
Test Cases
Test cases
Selected test cases
Actions Testing Technique Expected Output Result Output
!
is empty. Run some tests to fill it up!
Assumptions
Assumptions
List the assumptions you made that helped you generate the test cases.
I assume that...
Test strategy
Test strategy
Explain your test strategy.
H1 H2 H3 B P
Bug description
Bug description
Describe the bug you found. Hint: you can use bug taxonomy.

A bug in standalone software

2024 Assignments > Students > ekl102 > Bug Hunt (Exam) > Element counter (3)

Requirements specification

Element counter

This program counts the number of occurrences of a certain element in an array. \

- Parameter 0: the input array (a string of characters)
- Parameter 1: the element (a single character)
- Expected output: the number of elements in the array that match the element

Bug Description

Does not check the last element of the input string

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>

int main(int argc, char *argv[])
{
    if (argc != 3) {
        printf("Usage:\n\t%s <string> <element>\n", argv[0]);
        return 1;
    }

    char *str = argv[1];
    int str_len = strlen(str);
    char *element = argv[2];
    int element_len = strlen(element);

    if (element_len != 1){
        printf("Invalid element entered");
        return 1;
    }

    int counter = 0;
    for (int i = 0; i < str_len-1; i++) {
        if (str[i] == element[0]) {
            counter++;
        }
    }

    printf("%d\n", counter);
}
```

Test cases

Selected test cases

#	Actions	Testing Technique	Expected Output	Result	Output	Parameter 0	Parameter 1	Par
ef0		EP	Warning me	Pass	Usage: elementCounter/bug3 <string> <element>	*NULL*	*NULL*	
067		EP	Warning me	Pass	Usage: elementCounter/bug3 <string> <element>	abc	*NULL*	
151		EP	1	Pass	1	abc	b	
1f3		EP	Warning me	Pass	Usage: elementCounter/bug3 <string> <element>	abc	b	
356		EP	Warning me	Pass	Invalid element entered	hello	*EMPTY*	
767		EP	Warning me	Pass	Invalid element entered	hello	ll	
f40		EP	0	Pass	0	*EMPTY*	l	
a49		Other	2	Pass	2	hello	l	
d2d		Other	1	Fail	0	hello	o	
80c		Other	1	Pass	1	hello	h	

Assumptions List the assumptions you made that helped you generate the test cases.

Some assumptions: param0 and param1 can besides the alphabet also contain other special characters such as $ and numbers such as 5.
Another assumption is when the user inputs invalid values, an informative warning message is displayed.

Test strategy Explain your test strategy.

H1 H2 H3

1) Code inspection. Check for common mistakes such as using = instead of ==, forgetting to declare variables, etc.

2) Use EP on the number of parameters. Valid class: 2 parameters. Invalid classes: 0 parameters, < 2 parameters and > 2 parameters. This results in 4 test cases.

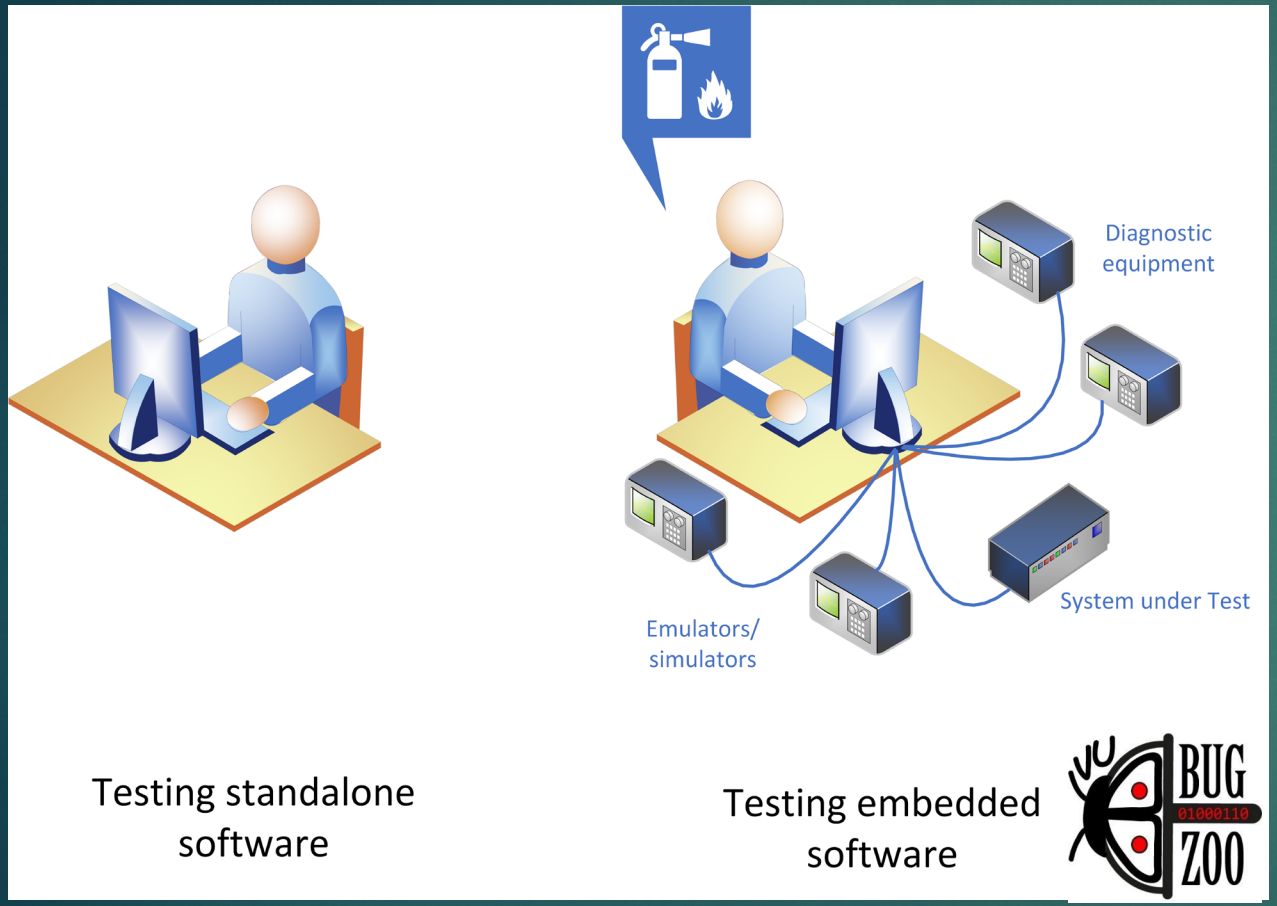
3) Control flow graph testing is used to generate test cases, with 100% decision coverage. See the control flow graph below.
Test cases resulting from the following paths:
Path 1-2: this path is an illegal amount of parameters and already tested using EP.
Path 1-3-4: this path is an illegal length of param1 (the length should be 1). Two test cases are generated for this path, using EP. Where the valid group is length of param1 is (test case #151) , and the invalid group is: length of param1 is 0, length of param1 is > 1. Resulting in 2 new test cases.
Path 1-3-5 and a combination of nodes 6 and 7 followed by node 8: This path has a lot of possible test cases. Therefore we choose a few as representatives:
param0 = hello, param1 = l so the character is in the middle of the word
param0= hello, param1 = o, so the character is the last letter of the word
param0 = hello, param1 = h, so the character is the first letter of the word

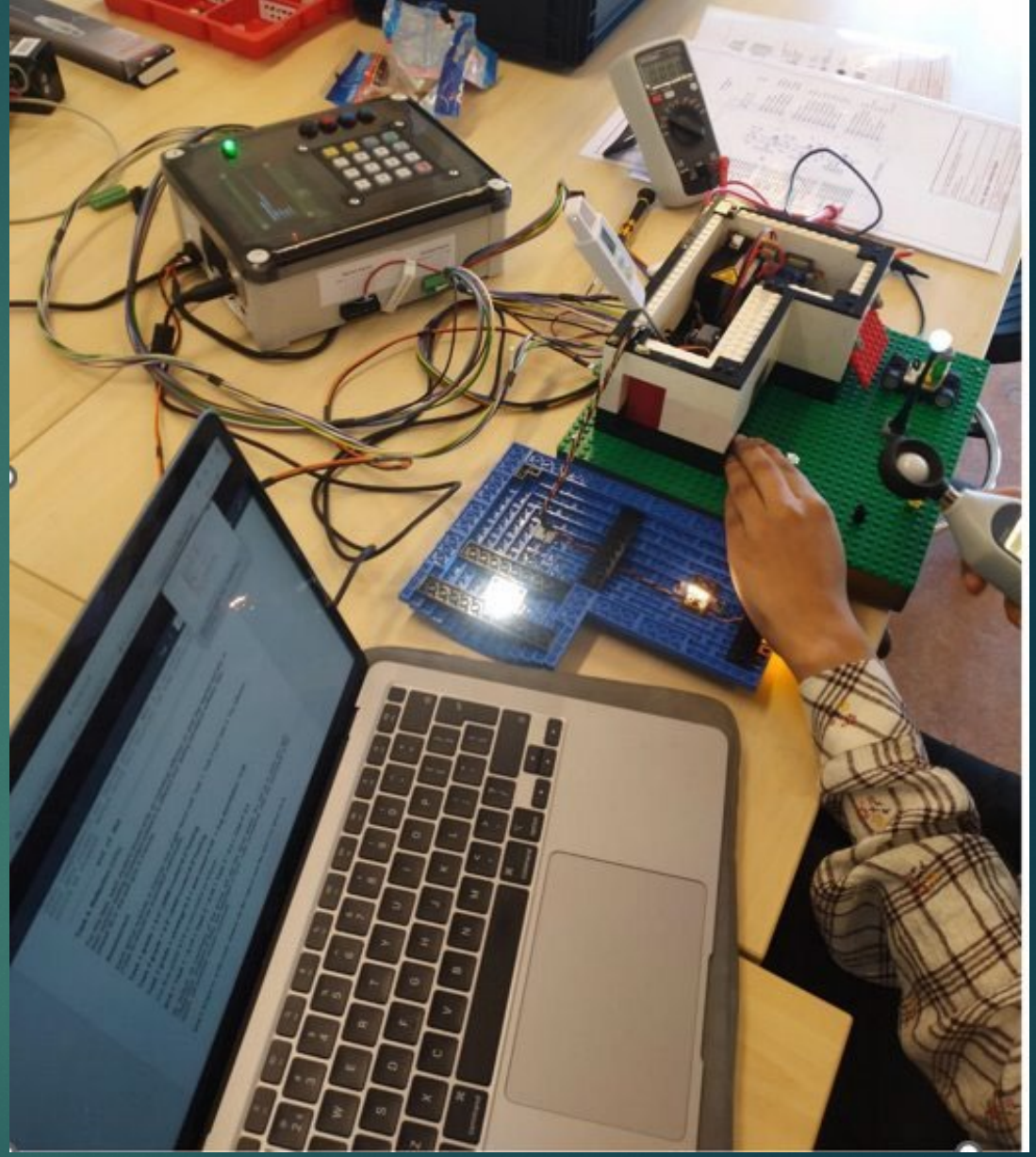
Bug description Describe the bug you found. Hint: you can use bug taxonomy.

I found a bug that when the parameter0 value is the last character of the parameter1 value, the output is 0, where is should not be 0 but at least 1. This bug was found in test case #d2d where param0 was hello and param1 was o, the program gave as output the value 0, where the correct output was the value 1. I can classify this bug as: 314x: loops and interation, because in the for loop (start, stop, increment), the stop statement is not correct because of the -1 that should not be there. This is the cause of this bug, that the for loop stops one character too soon.

DOWNLOAD PDF

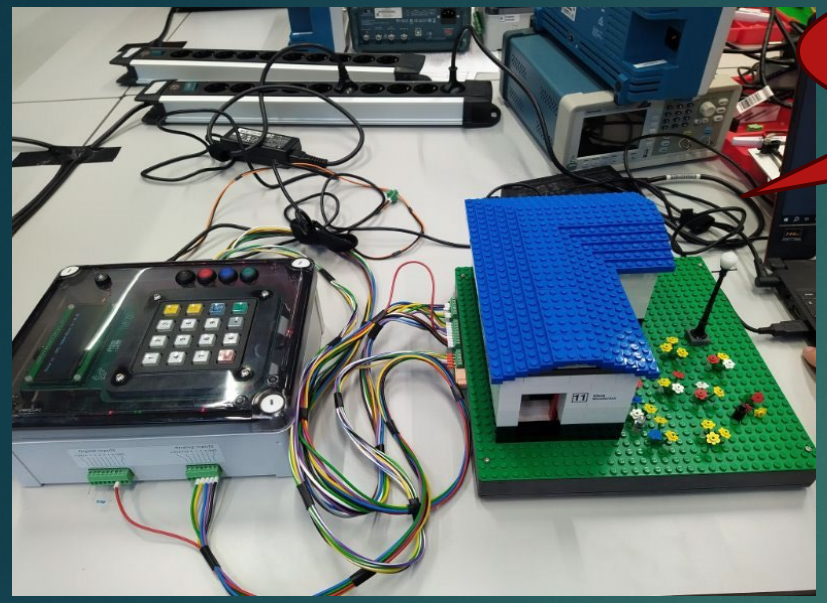
Embedded systems

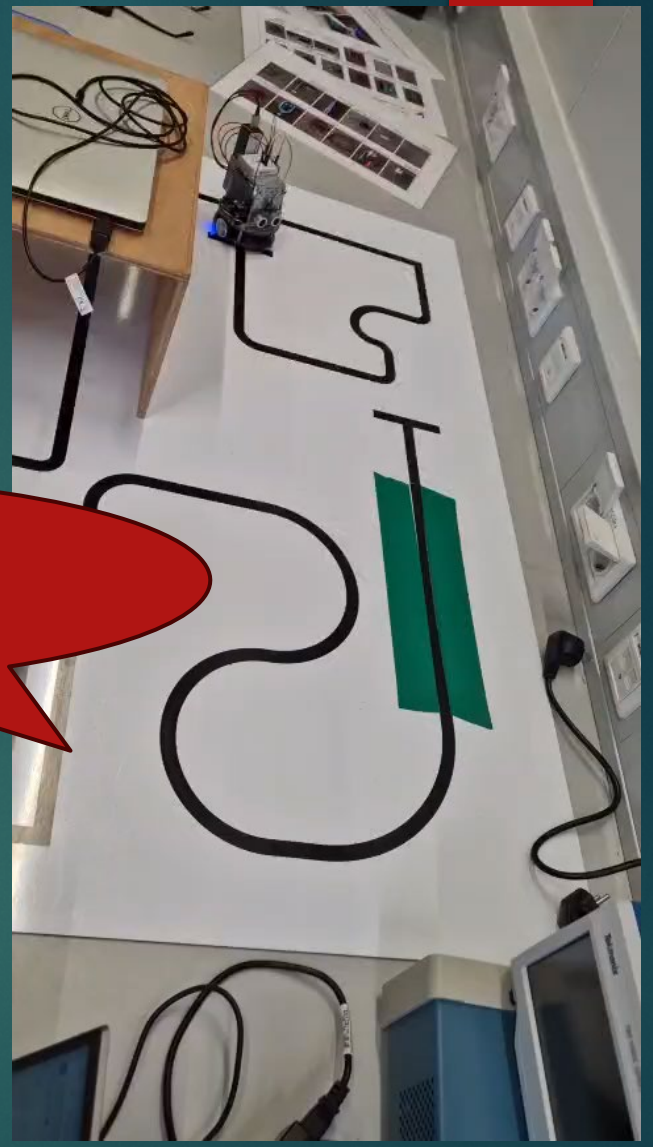




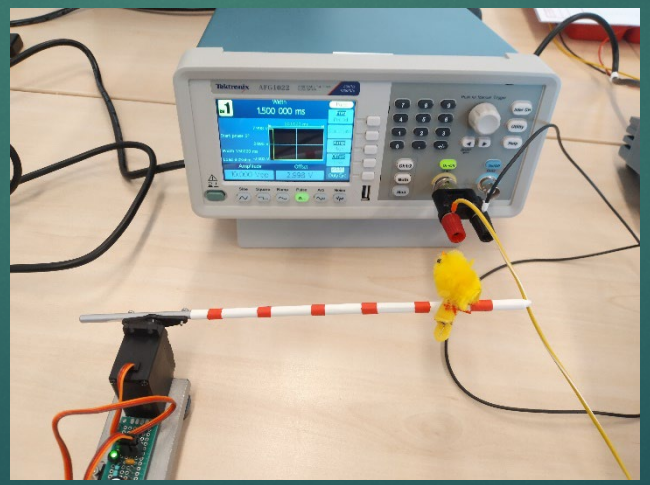
Natalia Silvis-Cividjian, Glenn Visser, Jasper Veltman, Niels Althuisius, Rob Limburg, and Mario Molenaar. House of the Rising Flames: A Hands-on, Bug-centered Tutorial on Embedded Software Testing. ITiCSE '23

VU-SmartHome



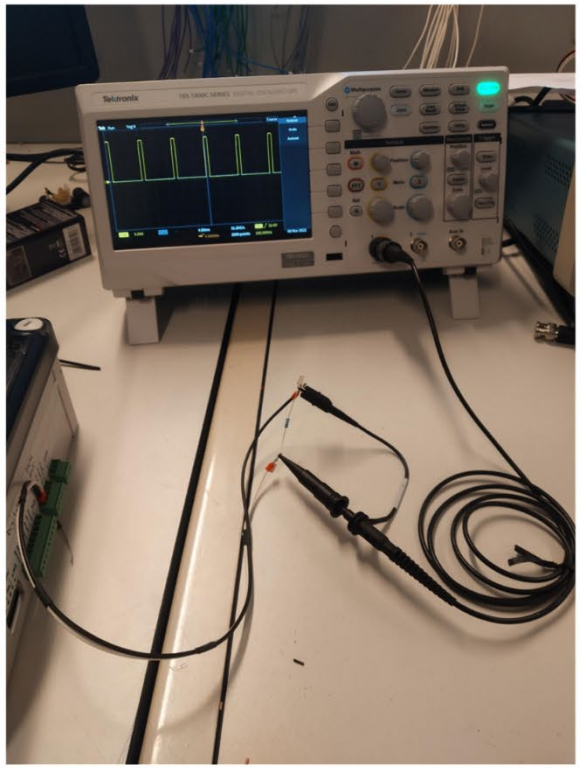


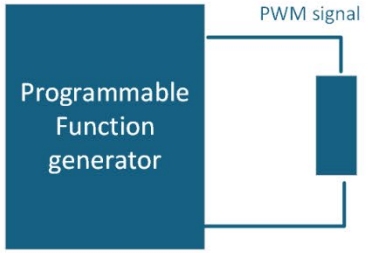
VU-Neville

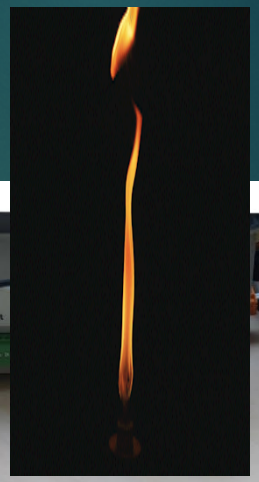


Silvis-Cividjian*TestDag-2025*Utrecht

A bug in embedded software







Correct: A PWM signal with **10%** duty cycle

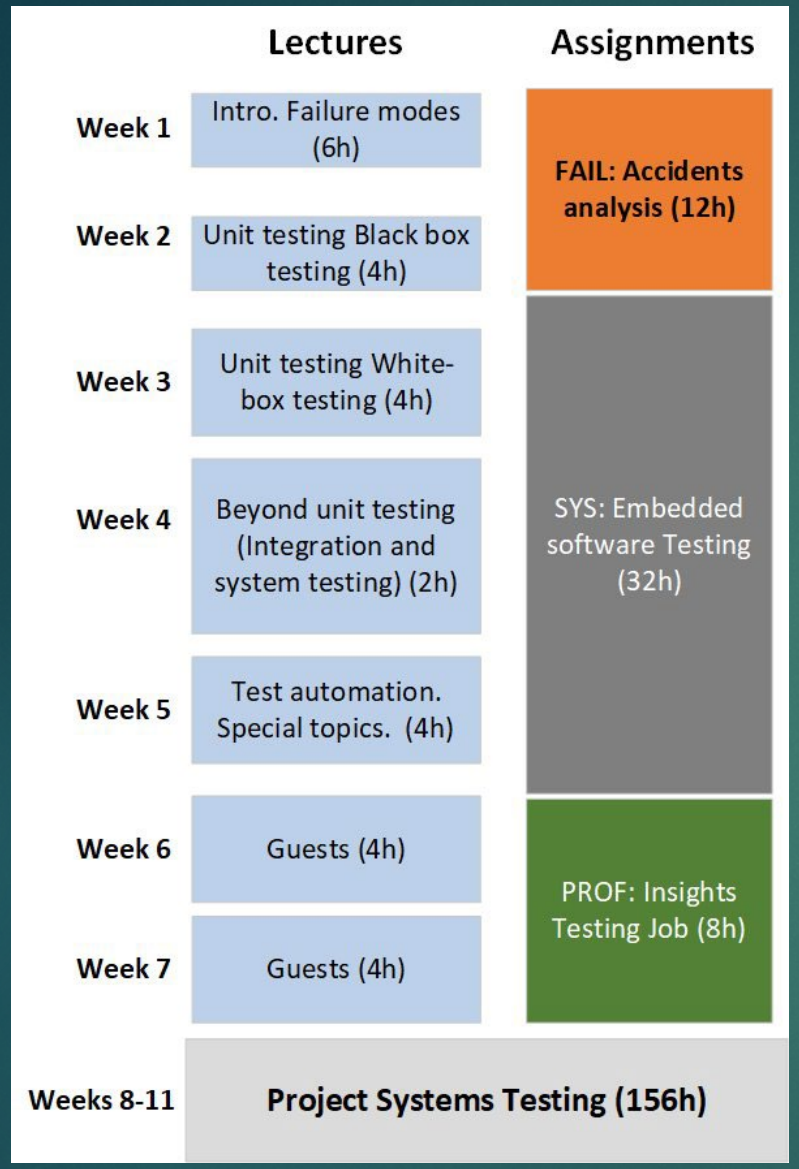
Bug: A PWM signal with **100%** duty cycle

Outline

- The problem
- A solution
- A few studies
- Conclusions & future work

Study#1. A Software Testing course for CS

Implementation



- **BUG-HUNT**: Exam for bug-hunting in **standalone** software
- **SYS**: A hands-on assignment for bug hunting in **embedded** software

Exam BUG-HUNT

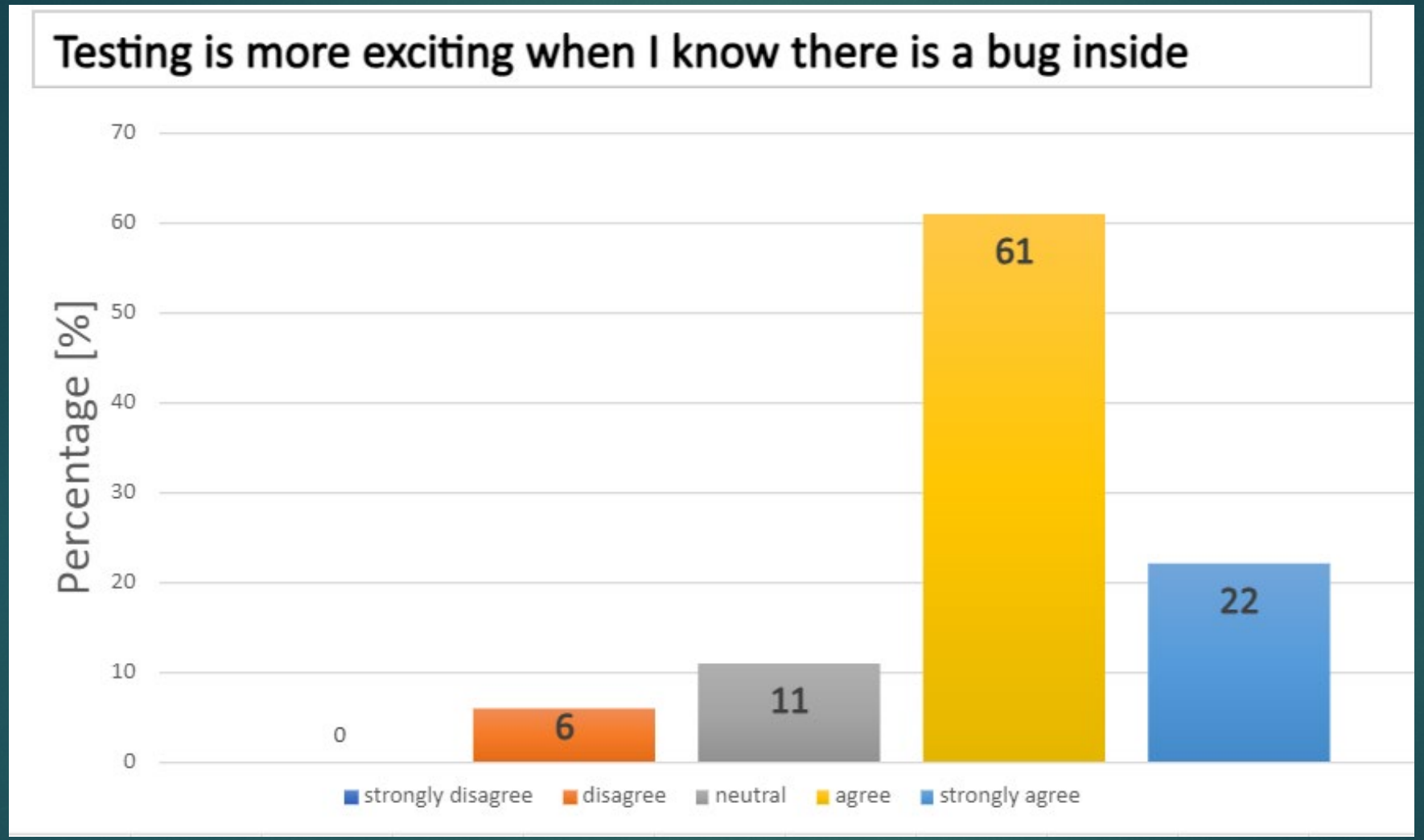
BUG-HUNT exam:

- In DBugIT
- 2020-2024, 100 students per year, online in 5 days, since 2024 in an exam hall for 3 hours, to prevent plagiarism
- Manual off-line grading of the submissions by the teacher

The task:

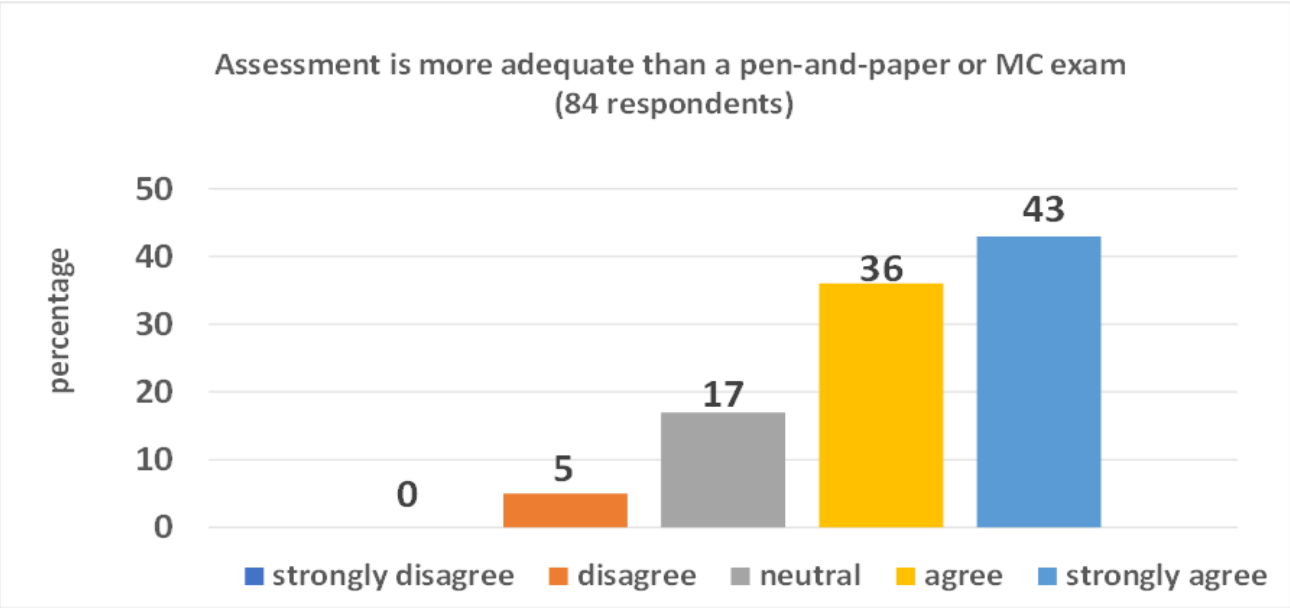
- Given an SRS, write a test strategy for black-box and white –box testing of standalone code.
- Design an optimal test suite
- Detect the bug and describe it using a bug taxonomy

Surveys' results CS students



Silvis-Cividjian, N., Went, M., Jansma, R., Bonev, V., & Apostolov, E. (2021). Good Bug Hunting: Inspiring and Motivating Software Testing Novices. In *ITiCSE 2021: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education. Vol. 1* (Vol. 1, pp. 171-177).

Results
CS students



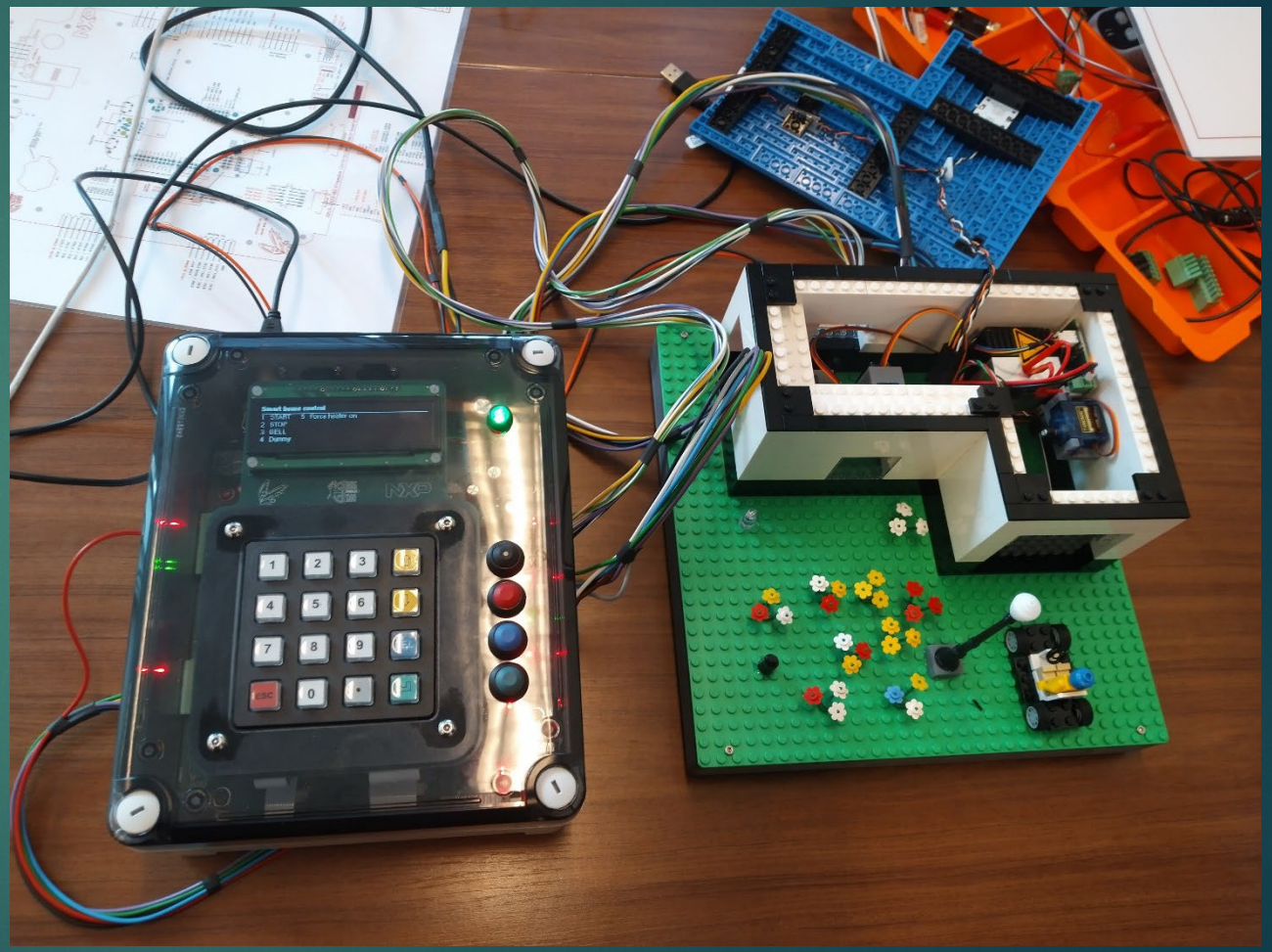
"I **laughed** out loud when I found the bug in black box testing.
I had a very nice assignment for this and I **enjoyed** testing it.
I was **eager** to find the bug(s).
To be honest, it was **stressful** for me because I knew there was a bug but it wasn't obvious.
Excitement when I found the bug.
Slight **stress** with the black box testing that I would not find the bug.
Feeling of **achievement** when found a bug.
Excitement when I figured out how the bug could be resolved.
However, I got a bit **frustrated** with some of the requirements, which were sometimes underspecified. "

SYS assignment

Embedded software

Smart Home

- FR-LIGHT-1 *Smart lighting*. When the light intensity outdoors is lower than 27 lux, the curtains shall close and the lights indoors, the light at the front door and the light in the garden shall turn on.
- FR-TEMP *Automatic temperature control.* The temperature in the room shall reach the range 30-33 degrees Celsius in less than 7 minutes. When in cooling mode, the same range is reached in less than 10 minutes. The temperature is kept in this range until the control system of the house is switched off.
- FR-GARAGE *Smart garage door*. The garage door opens automatically when a car is detected in front of it. When a car is above the Hall effect sensor (front wheels at the black markers on the floor), the door opens in max 2 seconds and the garage light turns on. 2 seconds after the car is gone, the door closes and the garage light turns off.



Tasks

- Given the requirements specification for VU-SmartHome
- Design a system test plan
- Find and describe two bugs hidden in the VU-SmartHome embedded code

Results (1) CS students

I liked the emphasis on bugs and the opportunity to make my own mistakes.

I disagree	2 respondents	3 %
I am neutral	13 respondents	16 %
I agree	64 respondents	81 %

Testing an embedded system when you know there is a bug in it (like the smart home) is more exciting than when you know that the code is bug-free (like the thermostat).

I disagree	5 respondents	6 %
I am neutral	9 respondents	11 %
I agree	65 respondents	82 %

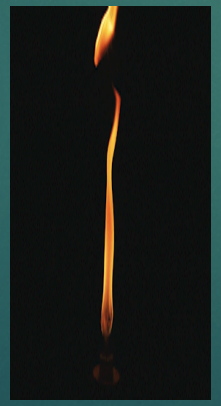
SYS was a necessary component in the course.

I disagree	3 respondents	4 %
I am neutral	14 respondents	18 %
I agree	62 respondents	78 %

With permission

Lessons learned

- Pro's: Games made CS students enthusiastic about testing
- Contra's: High costs and unpredictable damage
- However, the Eureka effect makes it all worth



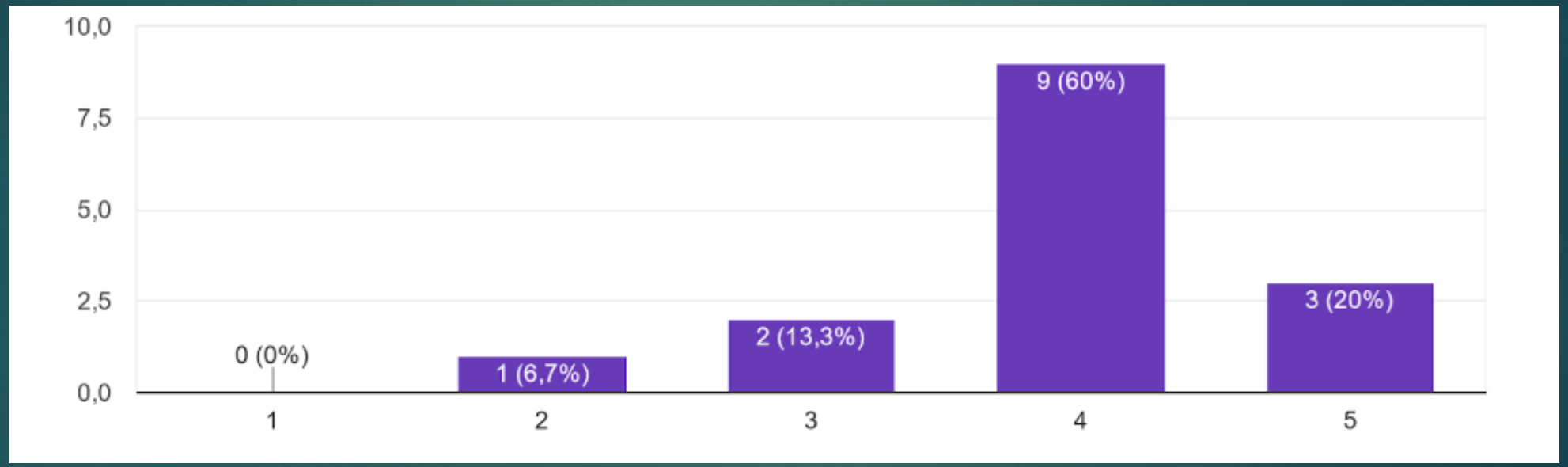
vs.

Study#2. A software testing workshop for non-VU students

- Used DBugIT for MakeITWork, a software engineering reintegration training course at Amsterdam Faculty of Applied Sciences
- 30 SE students with no testing experience

Results (MakeITWork/HvA)

Software testing is more exciting when you know there is a bug in it (15 respondents)



Other comments

Spel element is leuk

Speelse manier van leren. Spel element werkt goed van mij

-de gamification van het testen is vermakelijk

Maakt testen leuker

*Translation of **spelen** in Dutch–English dictionary*

spelen

verb

play [verb] to amuse oneself

play [verb] to take part in (games etc)

Outline

- The problem
- A solution
- A few studies
- Conclusions & future work

Conclusion

- Analyzing accidents increases motivation to test
- Bug-hunting adds excitement to learning.
- A bug-hunting exam is a better assessment instrument compared to a traditional MC exam.
- Bug hunting students reported experiencing emotions like excitement, relief, fun, but also stress and frustration.

Future work

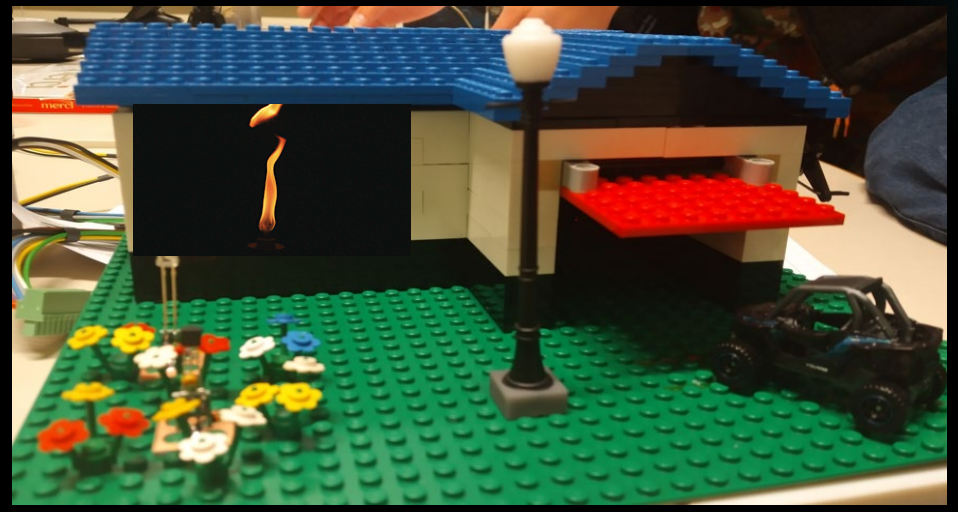
- We search for more pilots! Can you help?
- Use AI for automatic grading and immediate feedback
- Monitor students' emotions while testing
- Embed more gaming elements, score board, competition, rewards, etc

Acknowledgements

- NRO, The Netherlands Initiative for Education Research, for funding (Comenius Teaching Fellow grant and Kennisbenutting Plus grant)
- DBugIT dev team
- VU Mechanical and Electrical Engineering workshops
- Teaching assistants and students

DBugIT demo

https://app.dbugit.io/register/?reflink=c3f90da0-9541-475d-91a8-63b90eb22542



Thank you!