Improving Engagement of Students in Software Engineering

Tanja E. J. Vos
Open Universiteit
Universidad Politecnica de Valencia

https://impress-project.eu/

ERASMUS+

Project 2017-1-NL01-KA203-035259

impress : about the project

An EU-funded project aiming at improving students' engagement in Software Engineering courses through gamification.

Open Universiteit

Universiteit Utrecht

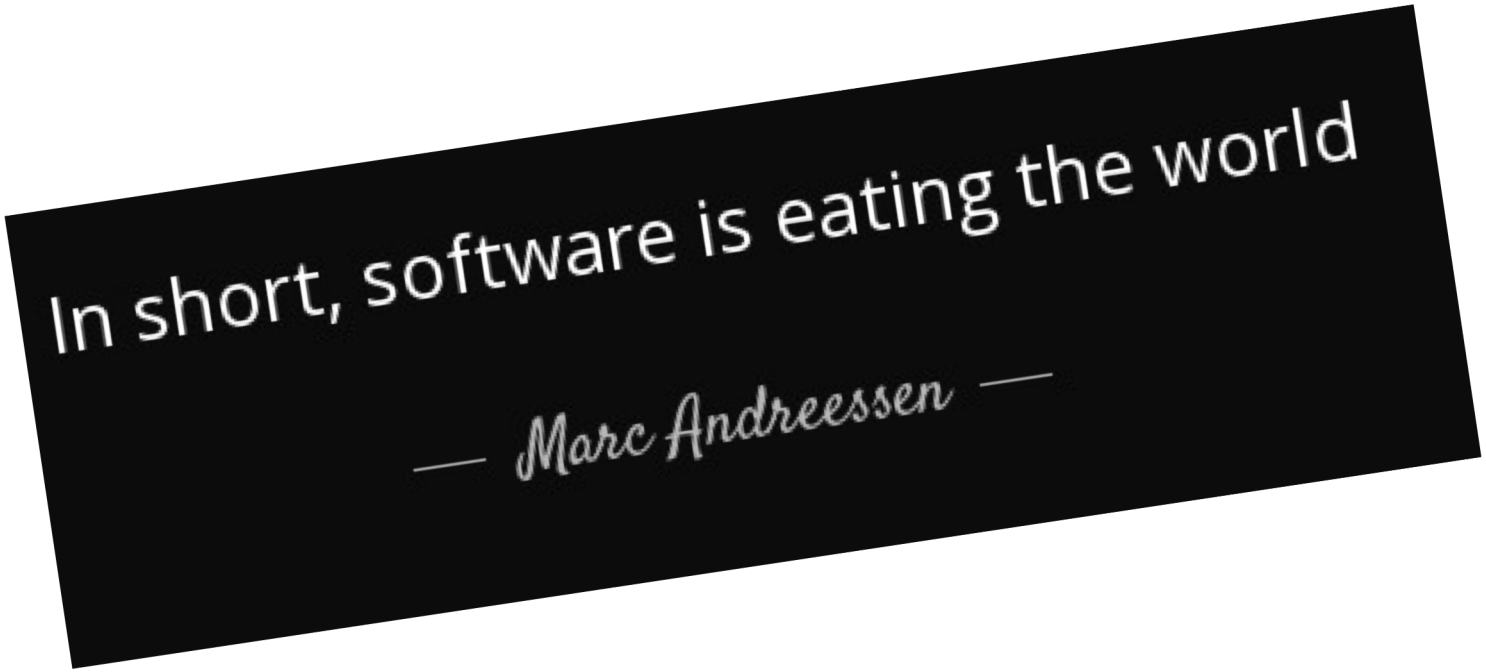
UNIVERSIDAD COMPLUTENSE MADRID

UNIVERSITÄT PASSAU

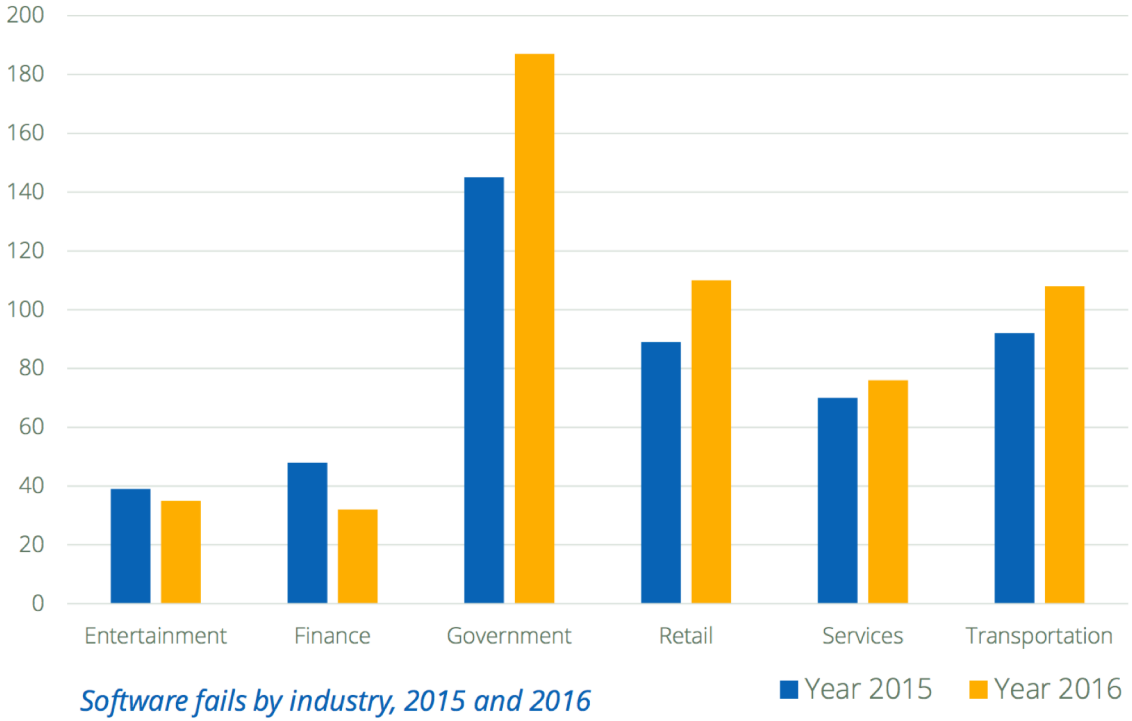
inesc id lisboa

https://impress-project.eu/

Software is everywhere



Failing software is everywhere



Software fails by industry, 2015 and 2016

2017 stats effects:

(3.7 billion people)

$1.7 trillion in assets

TRICENTIS

LOSSES FROM SOFTWARE FAILURES (USD)

1,715,430,778,504

ONETRILLIONSEVENHUNDREDFIFTEENBILLIONFOURHUNDREDTHIRTYMILLIONSEVENHUNDREDSEVENTY-EIGHTTHOUSANDFIVEHUNDREDFOUR

Consequences of failing software get worse


The Coming Software Apocalypse

Software engineering stakeholders

* Customers want to have quality products
* Bosses want to make money
* Engineers want to program wonders

Software engineering stakeholders

* Customers want to have quality products
* Bosses want to make money
* Engineers want to program wonders

What should we teach students?

Teaching/learning programming is fun

* Create something!
* Solve puzzles!
* See it work!
* Different solutions

Lego

Scratch

Pex (Microsoft)

My Duels▼ | Settings▼ | Sign In

Pex Coding Duel for fun

Random Puzzle | Learn | APCS | New | 1,858,343 clicked 'Ask Pex!' | C# | Visual Basic | F#

This puzzle is an interactive Coding Duel. Can you write code that matches a secret implementation? Other people have already won this Duel 10419 times! Help

```
using System;

public class Program {
  // Can you fill the puzzle method to match the secret arithmetic
operation?
  public static int Puzzle(int x, int y) {
    if (x == 0 && y == 0) return 0;
    if (x == 0 && y == 1) return 1;
    if (x == 1 && y == 0) return 1;
    return 0;
  }
}
```

Ask Pex! *Done. 4 interesting inputs found.* How does Pex work? Permalink

Pex found 1 difference between your puzzle method and the secret implementation. Improve your code, so that it matches the other implementation, and 'Ask Pex!' again.
You are not signed in. Sign In to rate duels and track your achievements. Help

	x	y	your result	secret implementation result	Output/Exception	Error Message
✓	0	0	0	0		
✓	0	1	1	1		
✗	0	2	0	2	Mismatch	Your puzzle method produced the wrong result.
✓	1	0	1	1		

Pex and Moles
Like Page 4.8K likes Tweet

Software engineering

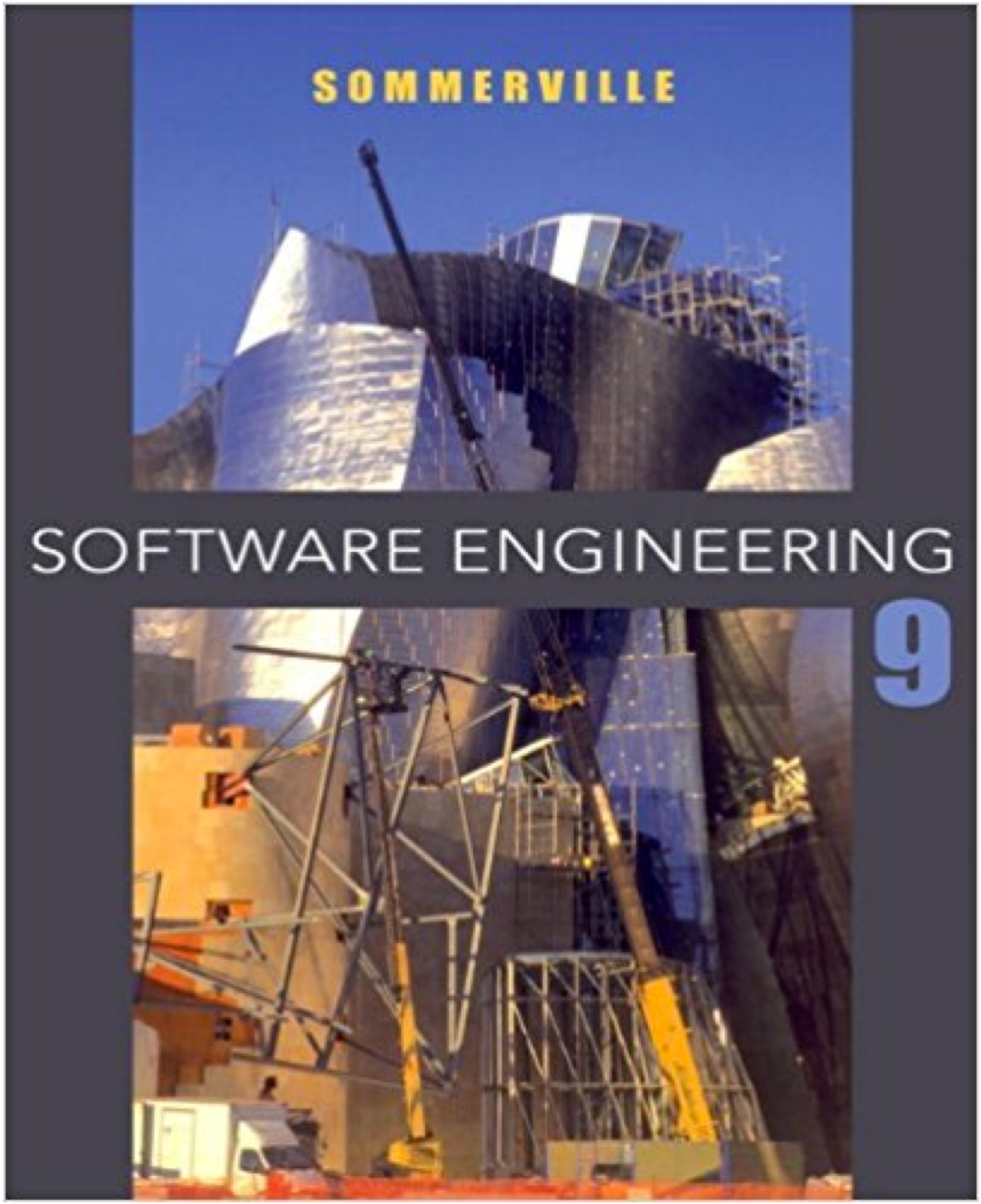
* Customers want to have quality products
* Bosses want to make money
* Engineers want to program wonders

But…… engineers should not only program
They also need to test the modules they build
... and invest in formalizing the modules' specification

Not only programming…
we need to teach software engineering

* waterfall, iterative, agile
* requirements, architecture, ….
* 14 UML diagram types
* 23 design patterns
* over 80 refactorings
* Testing
* Security
* ….
* …
* …

For example



For example


Roger S. Pressman
Sexta edición
INGENIERÍA DEL SOFTWARE
Un enfoque práctico
Mc Graw Hill

IMPRESS

- Can gamification improve the engagement in SE courses?
- Different level of gamification:
  - Gamified class room SE quizzes
  - SE education games
- Two additional aspects: integrated analytics and AI/automation to reduce teachers' effort.

Quizzes

kahoot.it

https://play.kahoot.it/#/?quizId=3a549d3a-c964-47d5-ad6c-c80f01964206

IMPRESS result 1

Quizzes on:

* Testing
* Introduction Software Engineering
* Security
* Formal specifications
* Java programming
* Software architecture

Games to learn testing

code-defenders.org

Mutation Testing

1970's — DeMillo (Georgia Tech), Lipton (Princeton), Sayward (Yale)

```
@Test
void testAbs() {
    int res = abs(42);
    assertEquals(42,
res);
}
```

```
int abs(int x) {
    if (x >= 0)
        return x;
    else
        return -x;
}
```

Mutation Testing

1970's — DeMillo (Georgia Tech), Lipton (Princeton), Sayward (Yale)

```
@Test
void testAbs() {
   int res = abs(42);
   assertEquals(42,
res);
}
```

```
int abs(int x) {
    if (x >= 0)
        return x;
}
```

```
int abs(int x) {
    if (x <= 0)
        return x;
    else
        return -x;
}
```

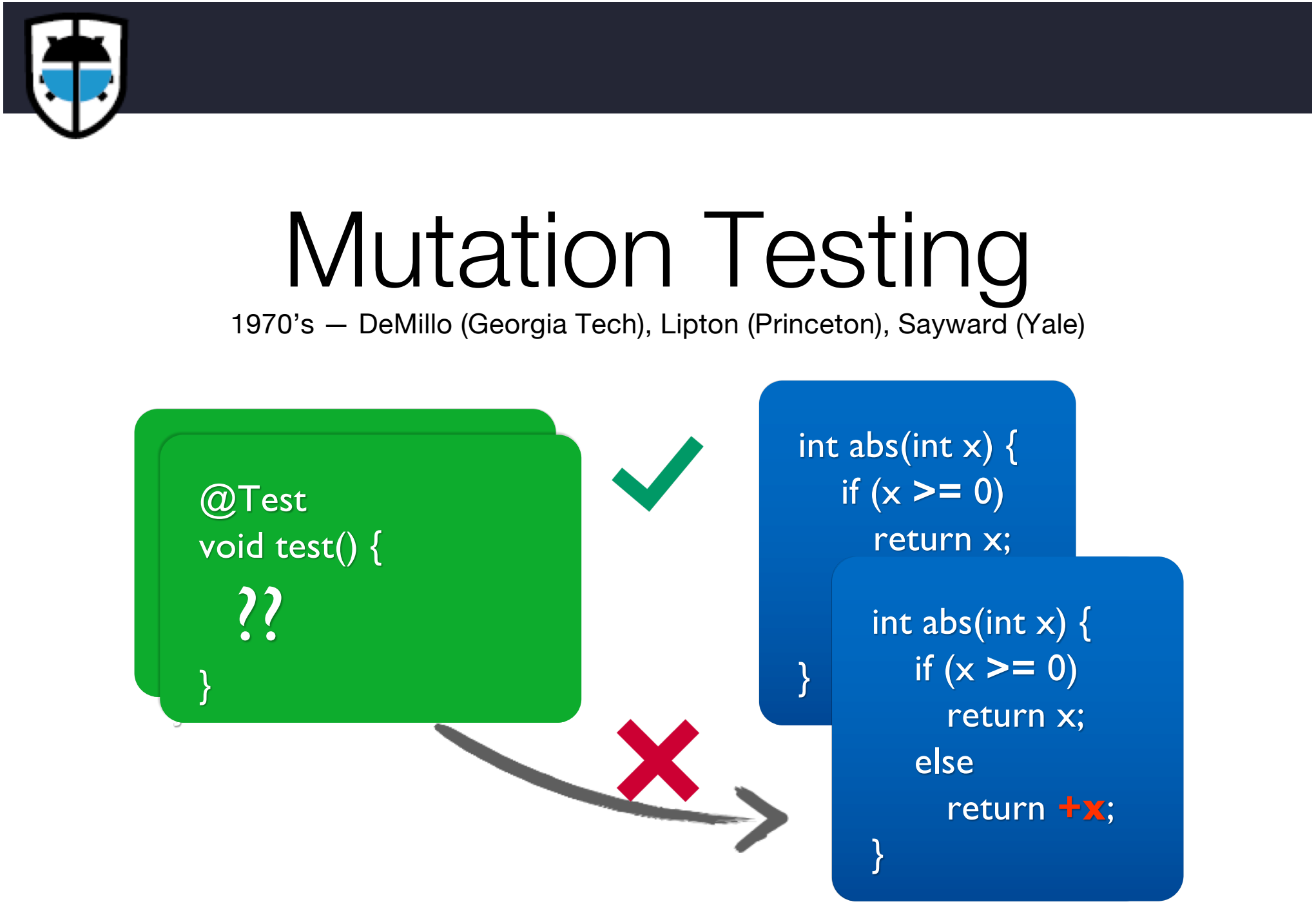
Mutation Testing

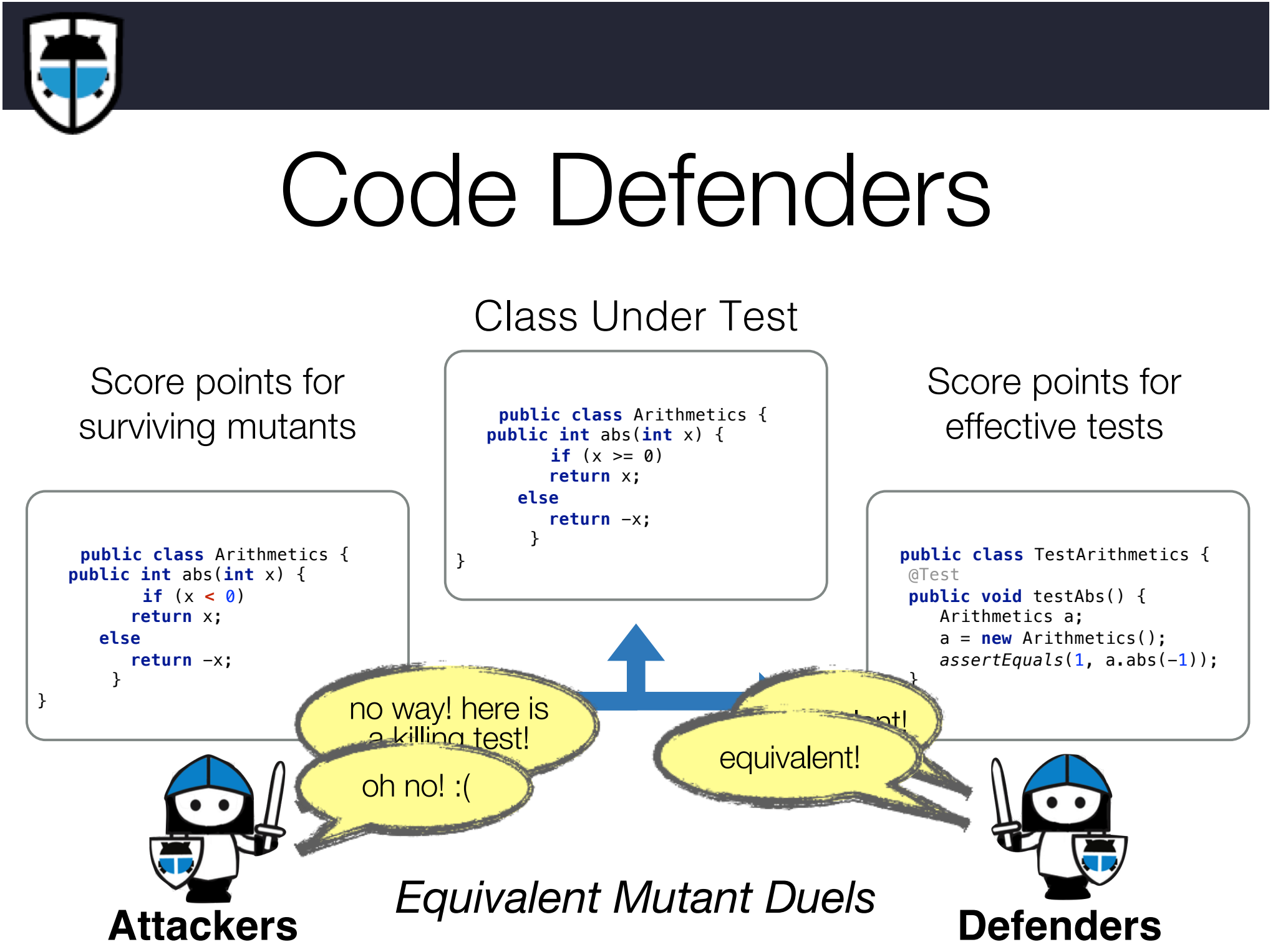
1970's — DeMillo (Georgia Tech), Lipton (Princeton), Sayward (Yale)

```
@Test
void testAbs() {
   int res = abs(42);
   assertEquals(42,
res);
}
```

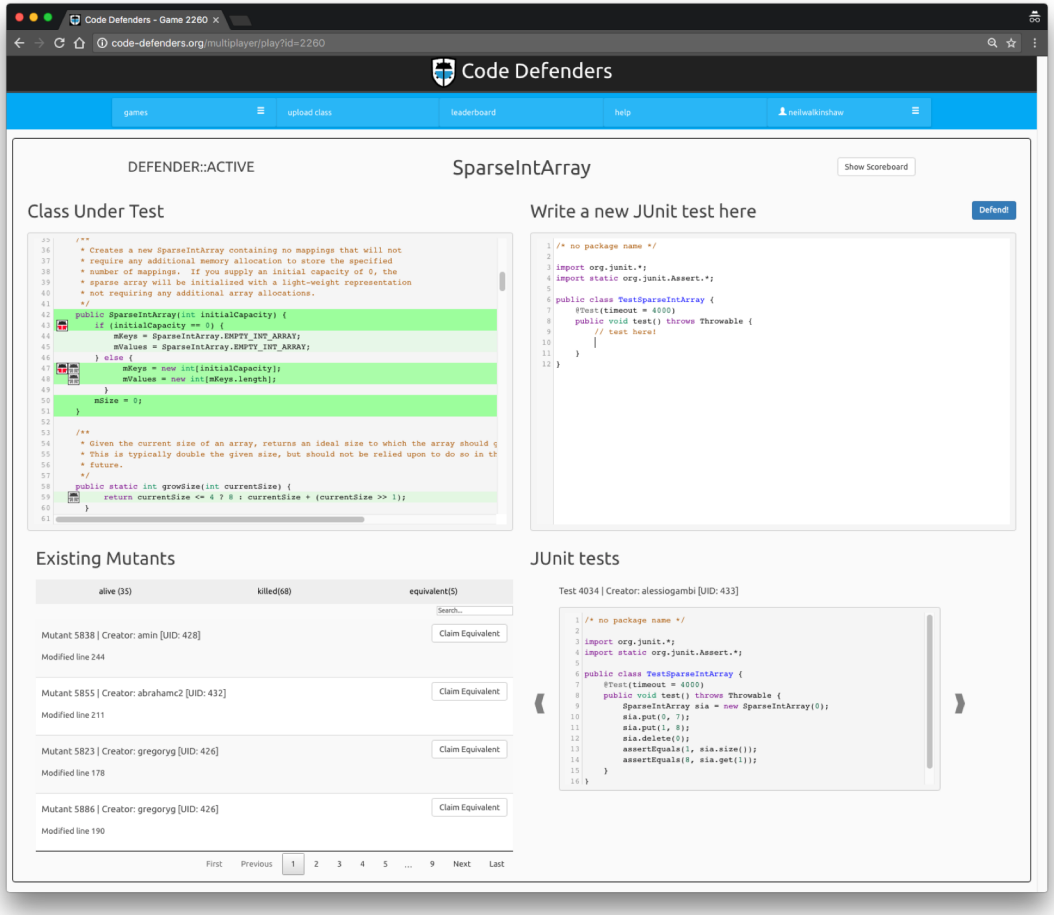
```
int abs(int x) {
    if (x >= 0)
        return x;
}
```

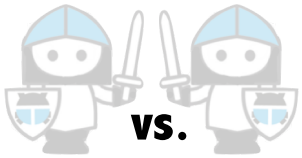
```
int abs(int x) {
    if (x >= 0)
        return x;
    else
        return +x;
}
```


Mutation Testing
1970's — DeMillo (Georgia Tech), Lipton (Princeton), Sayward (Yale)
@Test
void test() {
??
}
int abs(int x) {
if (x >= 0)
return x;
}
int abs(int x) {
if (x >= 0)
return x;
else
return +x;
}


Code Defenders
Class Under Test
Score points for surviving mutants
Score points for effective tests
public class Arithmetics {
public int abs(int x) {
if (x >= 0)
return x;
else
return -x;
}
}
public class Arithmetics {
public int abs(int x) {
if (x < 0)
return x;
else
return -x;
}
}
public class TestArithmetics {
@Test
public void testAbs() {
Arithmetics a;
a = new Arithmetics();
assertEquals(1, a.abs(-1));
}
no way! here is a killing test!
oh no! :(
equivalent!
Attackers
Equivalent Mutant Duels
Defenders

Code Defenders

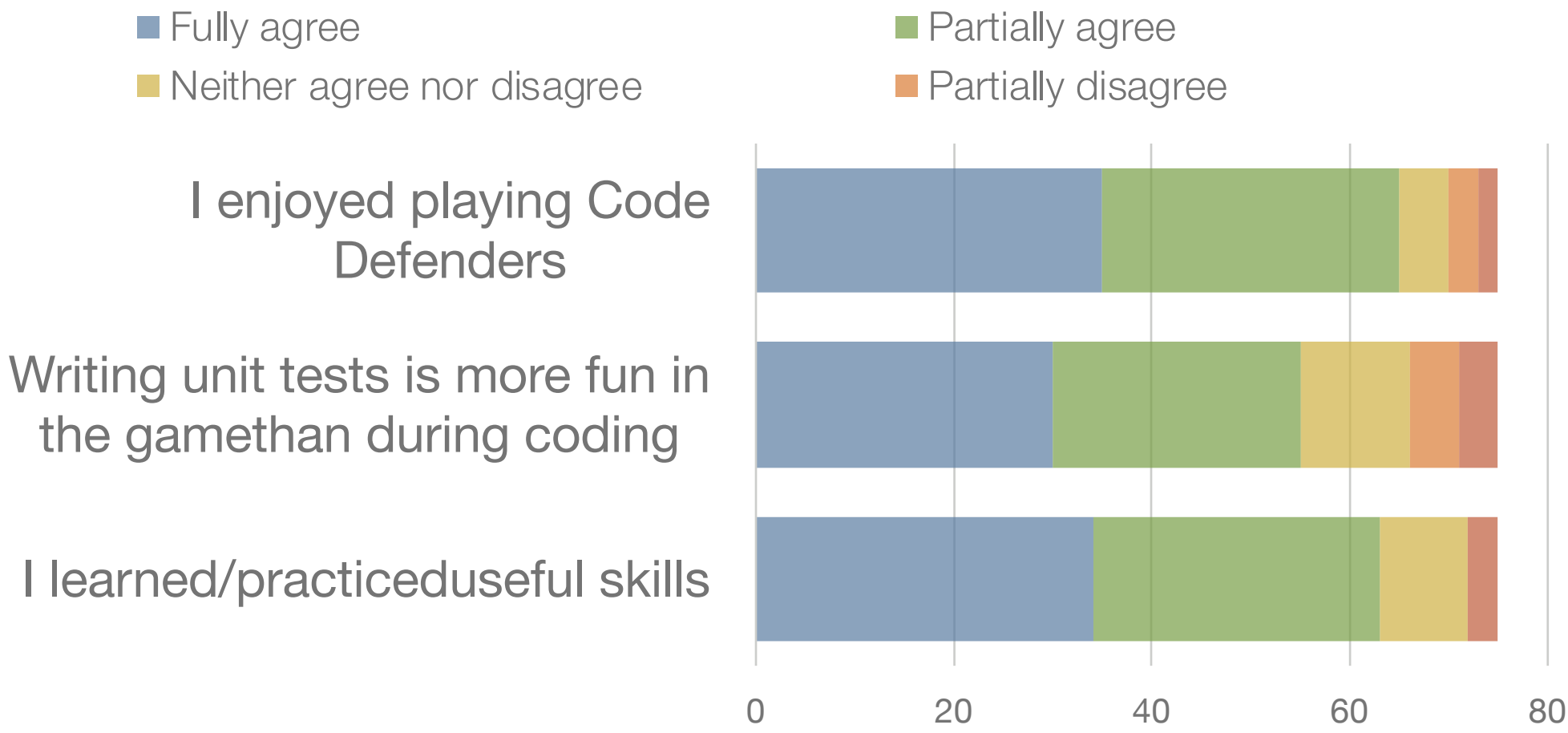




Two-player

Multi-player

Survey


Fully agree
Partially agree
Neither agree nor disagree
Partially disagree
I enjoyed playing Code Defenders
Writing unit tests is more fun in the gamethan during coding
I learned/practiceduseful skills
0
20
40
60
80

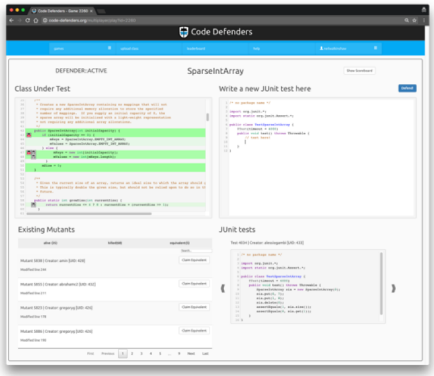
Open Challenges

- Communication mechanism to foster collaboration
- Many opportunities to spice up gameplay
- Integrating more technologies (GUI, concurrency, etc.)
- Analysis for grading and intervention
- …

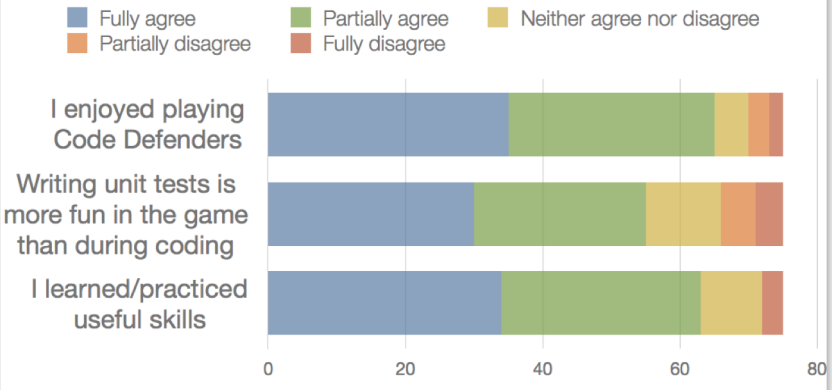
IMPRESS result 2

Teaching Software Testing

Code Defenders



Survey



code-defenders.org

impress

Games to learn to write formal specifications

Informal specifications…. source of bugs..

Article 5.4 – Marks

1. Marks will be assigned on a scale of 1 to 10. The final assessment of a course is satisfactory or unsatisfactory, where a 6 or higher is satisfactory. The examiner determines (final) grades using no more than one decimal. The final assessment is determined according to the method published along with the course and subsequently rounded as follows:

grade equals or larger than	until grade	rounded grade
3,85	4,00	3,9
4,95	5,50	5
5,50	6,05	6

Other grades will be rounded using one decimal: upwards if the second decimal equals 5 or more, and downwards if the second decimal equals 4 or less.

2. Alphanumeric results will be assigned in the following cases:

- a student who has registered for a course but who has not participated in a single test module will be assigned an ND (Niet Deelgenomen [Not Participated]);
- a student who has not participated in all of the mandatory test modules will be assigned a NVD (NietVoldaan [Not Completed]);
- a student who has completed a unit but who has not received a mark for it may be assigned a V (Voldoende [Satisfactory]) as their result;
- if the student has not completed a unit but does not receive a mark for it, the student can be given an ONV (ONVoldoende - Unsatisfactory) as the result;
- instead of an NVD or ONV the student who has performed to the best of their ability during a course may receive the mark AANV [AANVullende toets][extension];
- The AANV may also be granted in case no numerical grade can be determined, but the student is, according to the scoring rules of the course, entitled to an additional or substitute test, or by decision of the board of examiners.

A lesson in writing formal specifications

* We can write simple expressions:
  * constants like 1,2,3
  * identifiers like x,y,Students
  * properties, e.g. x.age, y.goal
  * $e_1 \otimes e_2$ where $\otimes$ is + , - , * , = , > , ≥ , < , ≤ , ∈

* A simple formula is a simple expression of type Boolean

A lesson in writing formal specifications

* A formula is either:
  * a simple formula
  * ∀identifier∈simple-expression● formula
  * ∃identifier∈simple-expression● formula)
* For example:
  * ∀x∈Students● x.age ≥ 16
  * ∃x∈Students● x.age = 16

A lesson in writing formal specifications

Let us

kahoot.it

https://play.kahoot.it/#/k/fef8e9b9-d851-4823-95ab-4cac2ad10b45

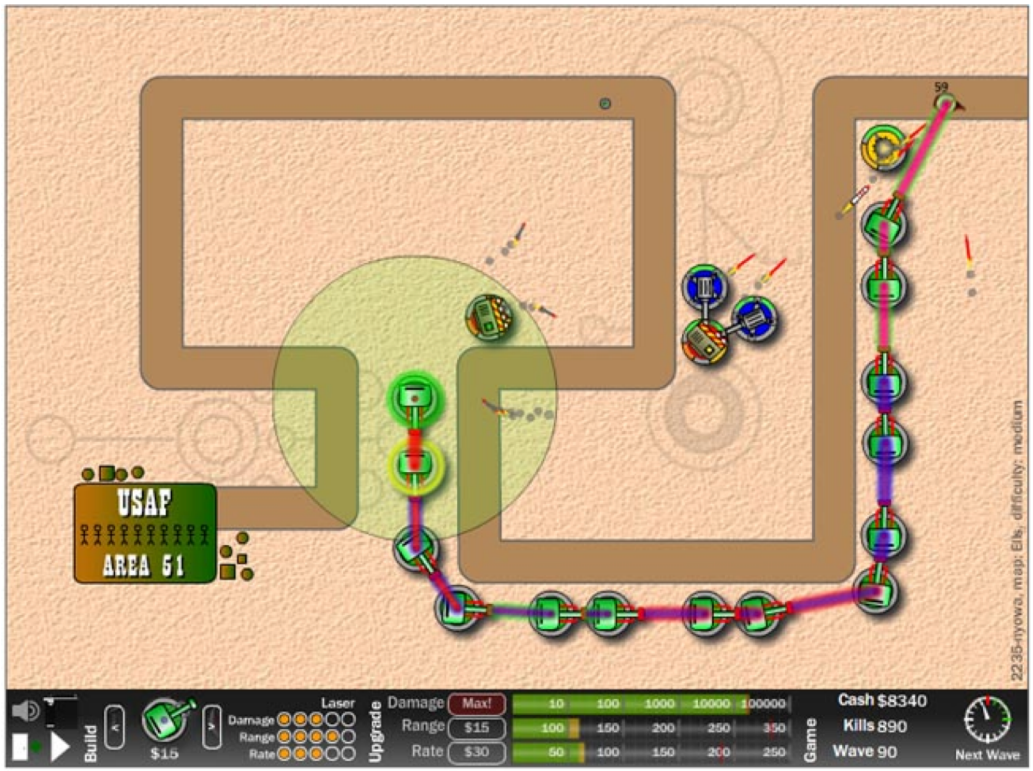
In production: Formal-Z game

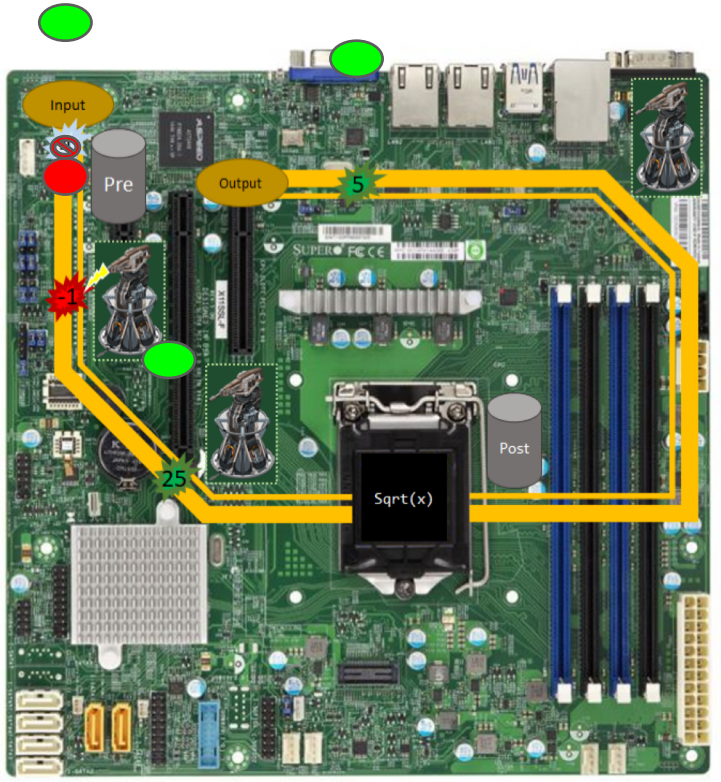
* a game to train student to write formal specifcations interpretable in Java
* will lean more towards the “engagement” aspect
* https://git.science.uu.nl/impresshs/javawlp

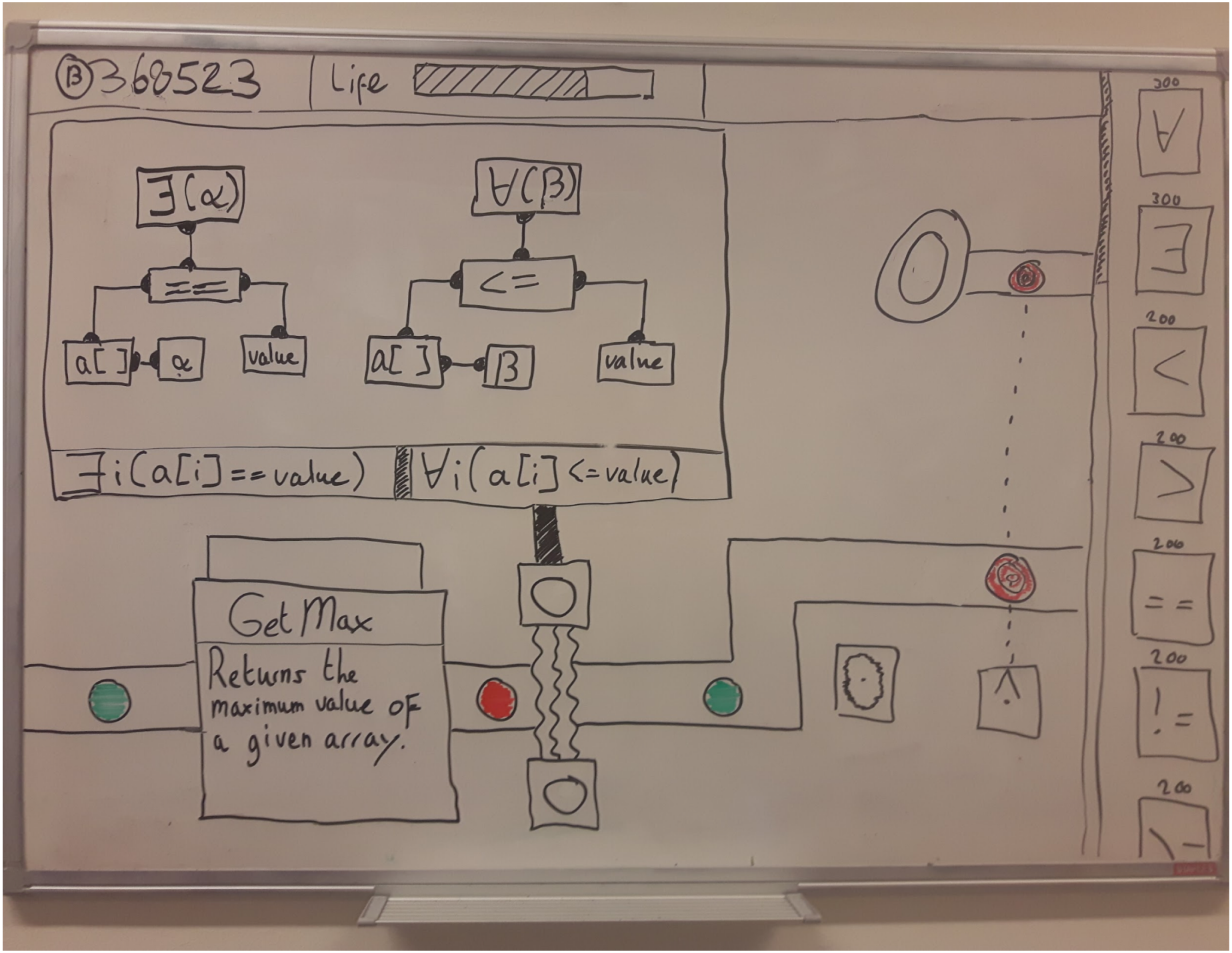
```
public static void getMax_spec1(int[] a) {
    // preconditions
    pre(a != null);
    pre(a.length > 0);

    // call the actual function implementation
    int retval = getMax(a);

    // postconditions
    post(exists(a, i -> a[i] == retval)); // A
    post(forall(a, i -> a[i] <= retval)); // B
}
```

The concept of Fomal-Z

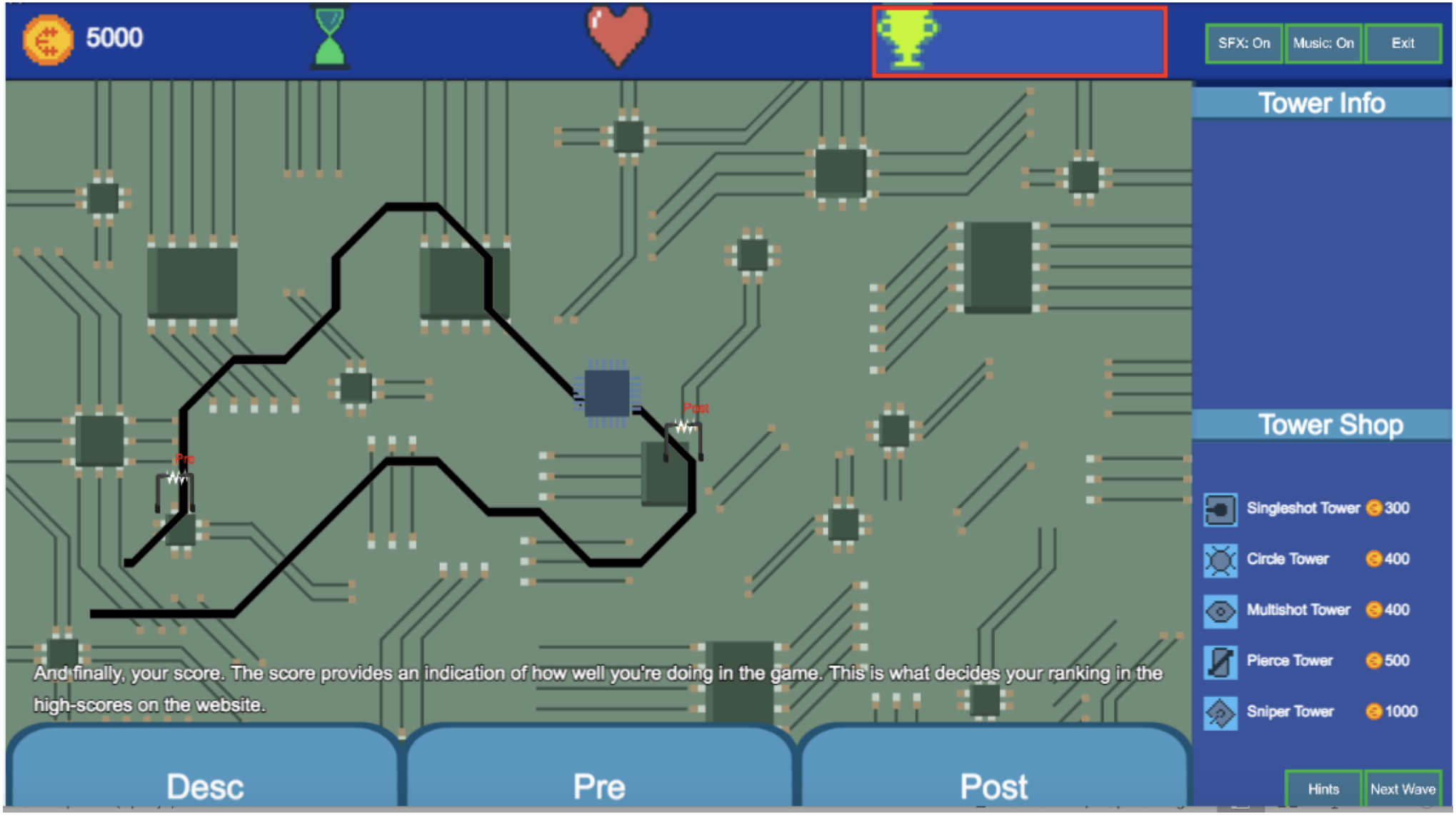
From tower defense to computer defense






B 368523
Life
∃(α)
==
a[ ]
α
value
∀(β)
<=
a[ ]
β
value
∃i(a[i] == value)
∀i(a[i] <= value)
Get Max
Returns the maximum value of a given array.
300
∀
300
∃
200
<
200
>
200
==
200
!=
200

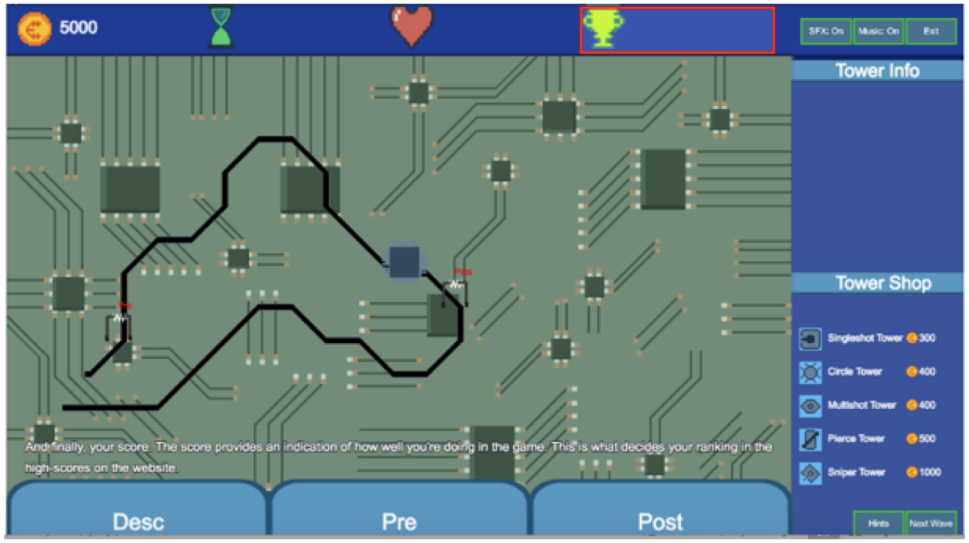
In production: Formal-Z game



IMPRESS result 3

A lesson in writing formal specifications

In production: Formal-Z game



impress

mpress future work

* Education quizzes and games for Software Engineering, experimenting with the balance between “seriousness” and “excitement”.

* Data analytics.

* Studying these innovations in actual class rooms.