Co-funded by the
Erasmus+ Programme
of the European Union

Department of
Computers and Informatics
FEEI TU of Košice

# Software Evolution — From green and greener to greenest application

Csaba Szabó

Technical university of Košice, Slovakia

# Green software, green IT

- Goals:
  - Save energy by more efficient hardware
  - Save energy by optimised/custom software
  - Save energy by location of hardware
- To make it really green:
  - Develop new working hardware
  - Develop energy efficient working software
  - Teach users to save energy when using the software
  - Make sure the used energy is also green

# Measuring energy consumption

Incl. improvements

- System level
- Application level
- Component level
- Code level

- Process level

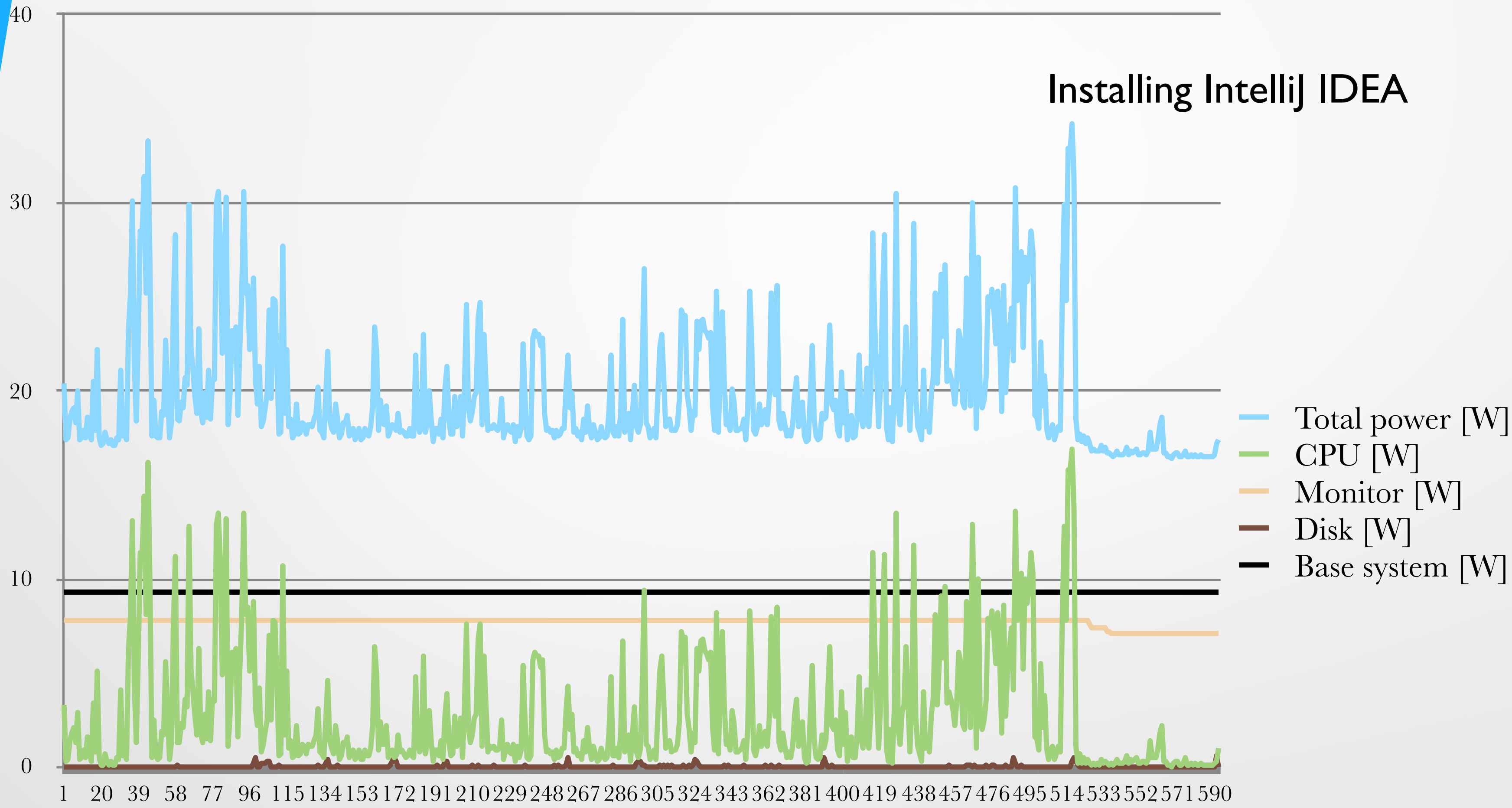
More details can be found in Intellectual output 1 and 4 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

# System level measurement

- SW-to-SW/HW solutions (servers, IoT)
- Uptime/availability prediction
- Providing a different evaluation perspective

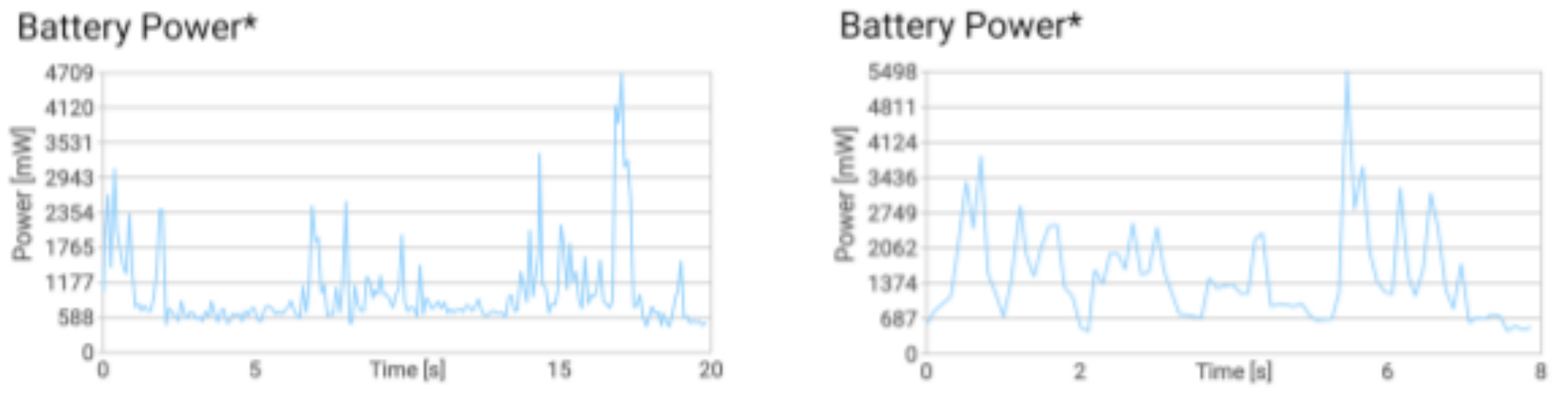
More details can be found in Intellectual output 1 and 4 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

# Application level measurement



More details can be found in Intellectual output 1 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

# Component level measurement



(a)

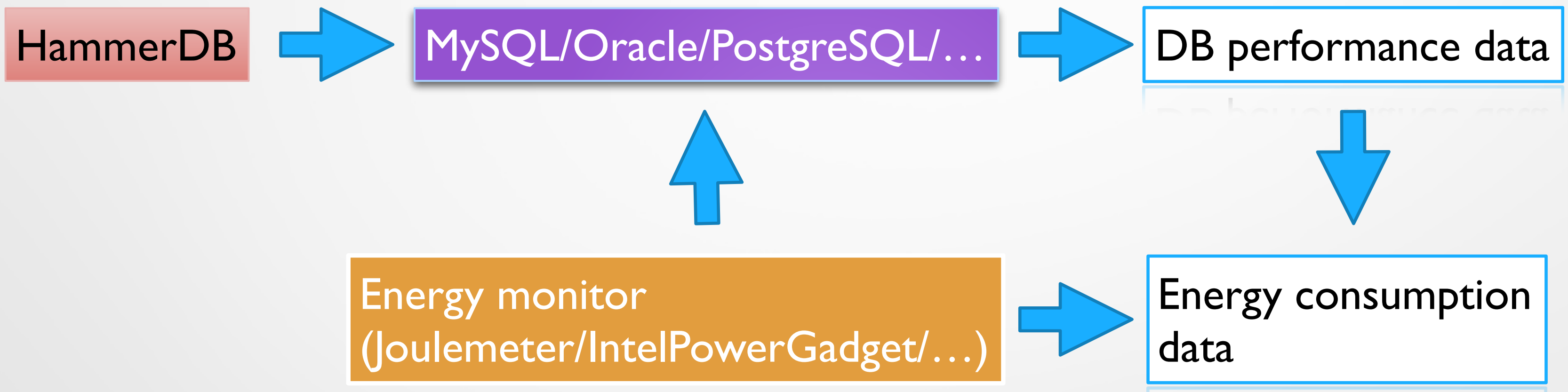
(b)

Fig. 5. Comparing (a) no image caching image caching vs (b) image caching concerning energy consumption

- Test oracles
- Comparing different versions
- The driver of energy (r)evolution

More details can be found in Intellectual output 2 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

# Measuring the DB



HammerDB

MySQL/Oracle/PostgreSQL/…

DB performance data

Energy monitor (Joulemeter/IntelPowerGadget/…)

Energy consumption data

More details can be found in Intellectual output 4 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

# sync_binlog - energy optimisation example

| sync_binlog | CPU energy (W) | HammerDB (trans/min) | trans/min/W |
|---|---|---|---|
| 0 | 8,03 | 5039 | 627,66 |
| 1 | 8,03 | 2877 | 358,46 |
| 5 | 7,71 | 4057 | 526,12 |
| 10 | 7,73 | 4511 | 583,48 |

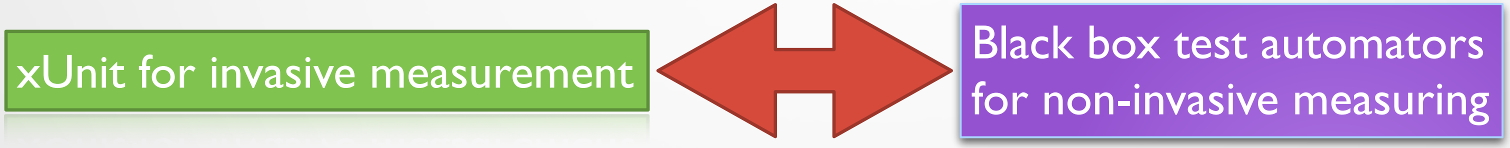
More details can be found in Intellectual output 4 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

# Code level measurement

- Which version of an algorithm is consuming less energy?
- Is it more efficient to store objects in an array than in a list?
- How significantly does the length of execution impact on the energy consumption?

More details can be found in Intellectual output 4 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

# Applying testing frameworks



xUnit for invasive measurement

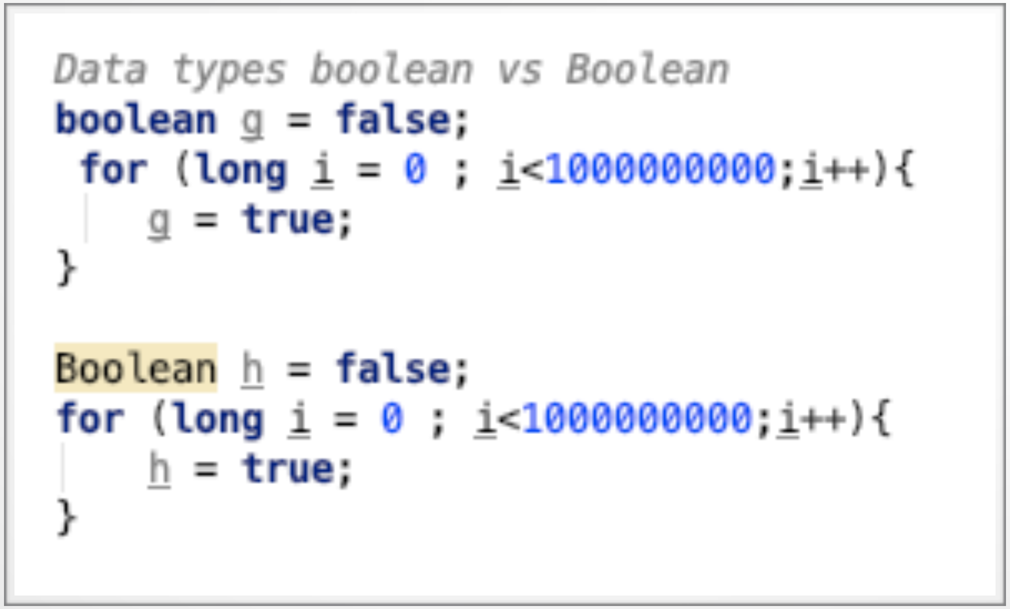
Black box test automators for non-invasive measuring

More details can be found in Intellectual output 1,2 and 4 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

# Boolean vs. boolean (billion times)

| Type | AVG exec (s) | AVG CPU NRG (W) | AVG RAM NRG (W) | AVG HDD NRG (W) | Test count |
|---|---|---|---|---|---|
| **boolean** | 0,49900 | 6,31915 | 0,00087 | n/a | 10000 |
| **Boolean** | 0,49879 | 6,26819 | 0,00071 | n/a | 10000 |

More details can be found in Intellectual output 4 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software



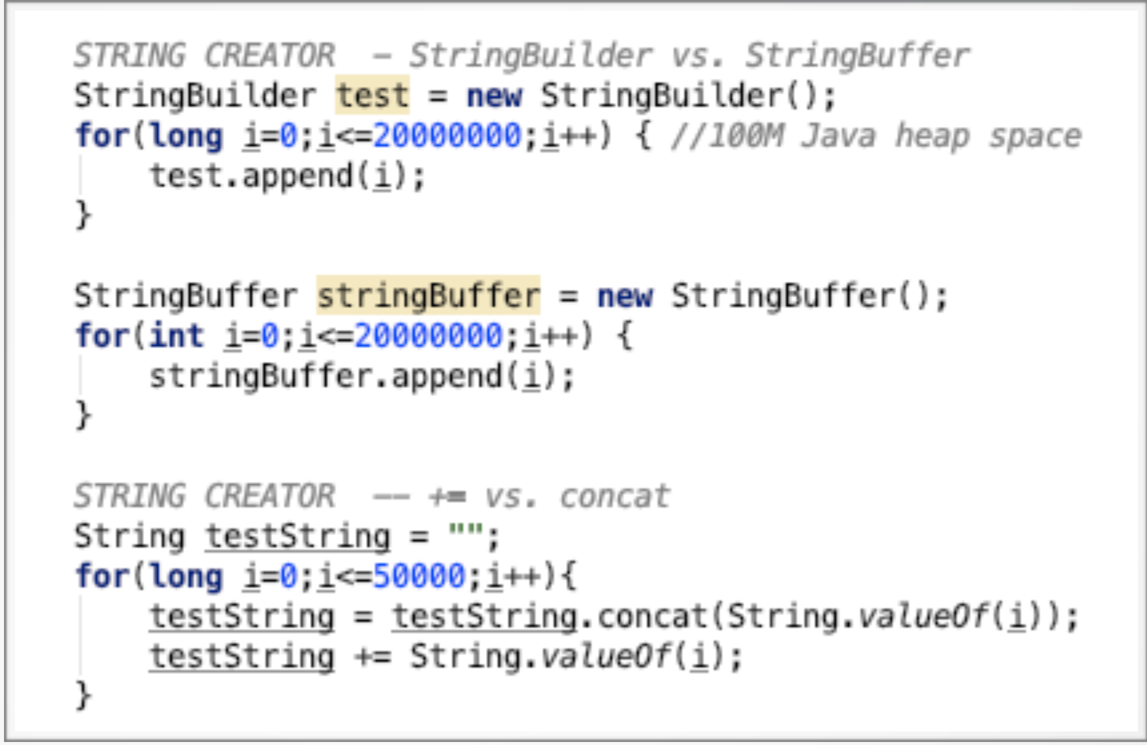
```
Data types boolean vs Boolean
boolean g = false;
 for (long i = 0 ; i<1000000000;i++){
     g = true;
}

Boolean h = false;
for (long i = 0 ; i<1000000000;i++){
     h = true;
}
```

# Double vs. double (billion times)

| Type | AVG exec (s) | AVG CPU NRG (W) | AVG RAM NRG (W) | AVG HDD NRG (W) | Test count |
|---|---|---|---|---|---|
| **double** | 1,01489 | 6,18481 | 0,00096 | n/a | 9643 |
| **Double** | 5,66532 | 7,41853 | 0,01001 | n/a | 9294 |

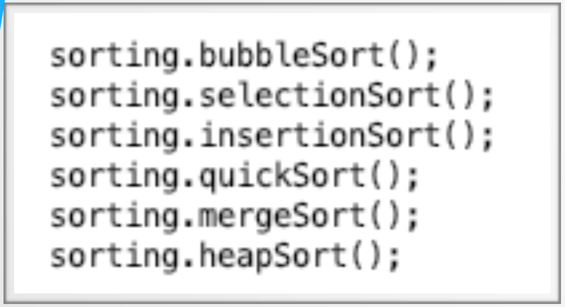
More details can be found in Intellectual output 4 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software



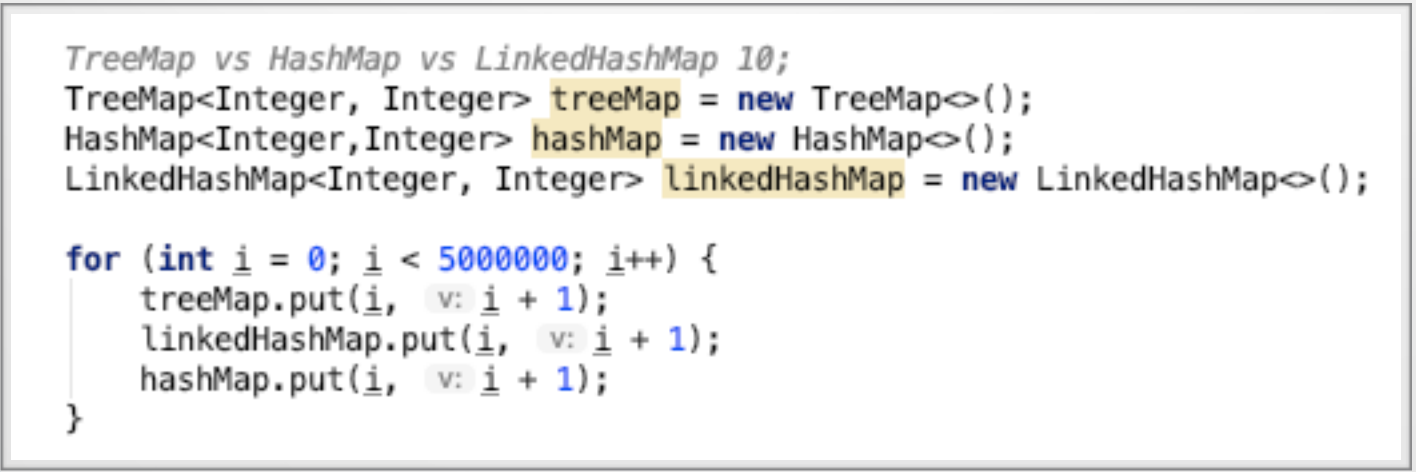
```
STRING CREATOR  – StringBuilder vs. StringBuffer
StringBuilder test = new StringBuilder();
for(long i=0;i<=20000000;i++) { //100M Java heap space
    test.append(i);
}

StringBuffer stringBuffer = new StringBuffer();
for(int i=0;i<=20000000;i++) {
    stringBuffer.append(i);
}

STRING CREATOR  -- += vs. concat
String testString = "";
for(long i=0;i<=50000;i++){
    testString = testString.concat(String.valueOf(i));
    testString += String.valueOf(i);
}
```



```
sorting.bubbleSort();
sorting.selectionSort();
sorting.insertionSort();
sorting.quickSort();
sorting.mergeSort();
sorting.heapSort();
```



```
TreeMap vs HashMap vs LinkedHashMap 10;
TreeMap<Integer, Integer> treeMap = new TreeMap<>();
HashMap<Integer,Integer> hashMap = new HashMap<>();
LinkedHashMap<Integer, Integer> linkedHashMap = new LinkedHashMap<>();

for (int i = 0; i < 5000000; i++) {
    treeMap.put(i,  v: i + 1);
    linkedHashMap.put(i,  v: i + 1);
    hashMap.put(i,  v: i + 1);
}
```

# Hashing algorithms — Which one?

| Type | AVG exec (s) | AVG CPU NRG (W) | AVG RAM NRG (W) | AVG HDD NRG (W) | Test count |
|---|---|---|---|---|---|
| **MD5** | 0,70915 | 6,08208 | 0,02763 | n/a | 9997 |
| **SHA-1** | 0,92689 | 5,40548 | 0,03961 | 0,00046 | 9964 |
| **SHA-256** | 1,44644 | 5,46093 | 0,06166 | 0,44541 | 9997 |
| **SHA-384** | 1,01102 | 5,80849 | 0,04340 | 1,54517 | 9998 |
| **SHA-512** | 1,01143 | 5,67845 | 0,04380 | 0,00054 | 9996 |

# Scaling up

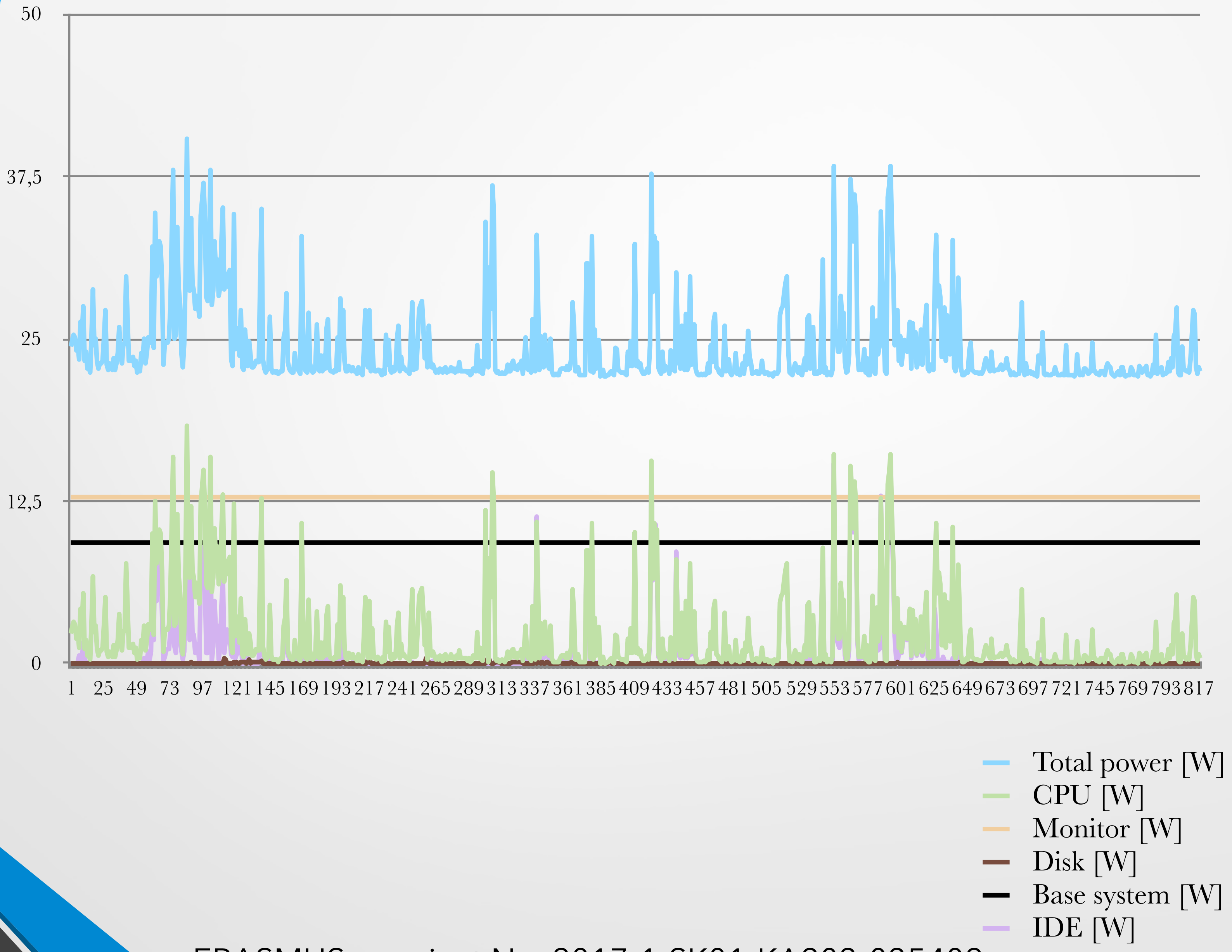
- Usual energy efficiency measurement focuses on software or hardware products.
- But, in our case we will measure the development (host) system's energy efficiency using a black-box testing method.
- We start the measurement before starting the browser and the IDE and we will stop measuring after closing all used tools.

More details can be found in Intellectual output 2 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

During development it is normal to compile and run an application many times and use other design and testing tools as well, which will have an effect on energy consumption. The goal of our measurements is to point out this energy.

# Process level measurement

More details can be found in Intellectual output 2 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software


50
37,5
25
12,5
0
1
25
49
73
97
121
145
169
193
217
241
265
289
313
337
361
385
409
433
457
481
505
529
553
577
601
625
649
673
697
721
745
769
793
817
Total power [W]
CPU [W]
Monitor [W]
Disk [W]
Base system [W]
IDE [W]

# The energy-measured development game

1. Setup the environment
2. Start the energy monitor
3. Develop (think, code, test, fix) for 15 minutes
4. Have a 5 minutes break (stop energy usage monitoring, set up the next one, get a coffee)
5. Finish (for this time) if there is no further idea
6. Repeat (jump to label 2)
7. Analyse collected data (energy efficiency of your development process) inside the team

More details can be found in Intellectual output 1 and 2 of the project 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software

kpi

TECHNICAL UNIVERSITY OF KOŠICE
Faculty of Electrical Engineering
and Informatics

Liked it? ☺

Csaba.Szabo@tuke.sk

# Disclaimer

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

This presentation contains parts of Intellectual output 1, 2 and 4 of the ERASMUS+ project No. 2017-1-SK01-KA203-035402: Focusing Education on Composability, Comprehensibility and Correctness of Working Software as reference to older Intellectual outputs of the specific project.

Co-funded by the
Erasmus+ Programme
of the European Union