Monitoring Energy Across Hardware, Software, and Humans

Prof. Adel Noureddine

University Paris Nanterre
Sorbonne University, CNRS, LIP6

DevOpsSustain workshop, @FSE 2026
Montreal, Canada – 05 July 2026

noureddine.org

Monitoring Energy Across Hardware, Software, and Humans

Prof. Adel Noureddine

University Paris Nanterre
Sorbonne University, CNRS, LIP6

DevOpsSustain workshop, @FSE 2026
Montreal, Canada – 05 July 2026

noureddine.org

Everywhere
Anywhere
for Everyone

Université Paris Nanterre · LIP6 · Sorbonne Université · cnrs


Université de Lille
Inria
THE UNIVERSITY of EDINBURGH
UEL University of East London
catamania
DIGITAL process for HUMAN progress
IVERSITÉ DE PAU ET DES PAYS DE L'ADOUR
Université Paris Nanterre

March 2014 **PhD** in Computer Science

Université de Lille

Inria

THE UNIVERSITY of EDINBURGH

UEL University of East London

catamania DIGITAL process for HUMAN progress

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

Université Paris Nanterre

Energy Consumption of Software Systems

(from hardware components to software source code)

Main results of PhD thesis

Main results of PhD thesis

1. Energy models for CPU, disk, network & memory

Main results of PhD thesis

1. Energy models for CPU, disk, network & memory

2. Estimate software processes' energy

Main results of PhD thesis

1. Energy models for CPU, disk, network & memory
2. Estimate software processes' energy
3. Approach to estimate energy in source code

Research landscape in 2010

Research landscape in 2010

Software energy wasn’t being taken seriously

Research landscape in 2010

Software energy wasn’t being taken seriously

Very few people worked on green software

Research landscape in 2010

Software energy wasn’t being taken seriously

Very few people worked on green software

We didn’t know what impacts software energy

Université de Lille

Inria

THE UNIVERSITY of EDINBURGH

UEL University of East London

catamania
DIGITAL process for HUMAN progress

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

Université Paris Nanterre

March 2014 **PhD** in Computer Science

Université de Lille

Inria

THE UNIVERSITY of EDINBURGH

UEL University of East London

catamania DIGITAL process for HUMAN progress

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

Université Paris Nanterre

Energy Consumption of Software Systems

(from hardware components to software source code)

March 2014 **PhD** in Computer Science

2014-2015 Postdoctoral Researcher

Université de Lille

Inria

THE UNIVERSITY of EDINBURGH

UEL University of East London

catamania DIGITAL process for HUMAN progress

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

Université Paris Nanterre

Energy Consumption of Software Systems
(from hardware components to software source code)

Features & Metrics Influencing Software Energy
(static and dynamic metrics & design patterns)

March 2014 **PhD** in Computer Science

2014-2015 Postdoctoral Researcher

2015-2016 Postdoctoral Researcher

Université de Lille

Inria

THE UNIVERSITY of EDINBURGH

UEL University of East London

catamania DIGITAL process for HUMAN progress

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

Université Paris Nanterre

Energy Consumption of Software Systems
(from hardware components to software source code)

Features & Metrics Influencing Software Energy
(static and dynamic metrics & design patterns)

Energy Efficiency in Data Centers
(procurement, recommendations tool)

March 2014 **PhD** in Computer Science

Université de Lille

Inria

2014-2015 Postdoctoral Researcher

THE UNIVERSITY of EDINBURGH

2015-2016 Postdoctoral Researcher

UEL University of East London

2017-2018 R&D Engineer Project Manager

catamania DIGITAL process for HUMAN progress

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

Université Paris Nanterre

Energy Consumption of Software Systems
(from hardware components to software source code)

Features & Metrics Influencing Software Energy
(static and dynamic metrics & design patterns)

Energy Efficiency in Data Centers
(procurement, recommendations tool)

Product Manager @R&D Innova

March 2014 **PhD** in Computer Science

Université de Lille

Inria

2014-2015 Postdoctoral Researcher

THE UNIVERSITY of EDINBURGH

2015-2016 Postdoctoral Researcher

UEL University of East London

2017-2018 R&D Engineer Project Manager

catamania DIGITAL process for HUMAN progress

2018-2025 Associate Professor

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

Since 2025 Full Professor

Université Paris Nanterre

Energy Consumption of Software Systems
(from hardware components to software source code)

Features & Metrics Influencing Software Energy
(static and dynamic metrics & design patterns)

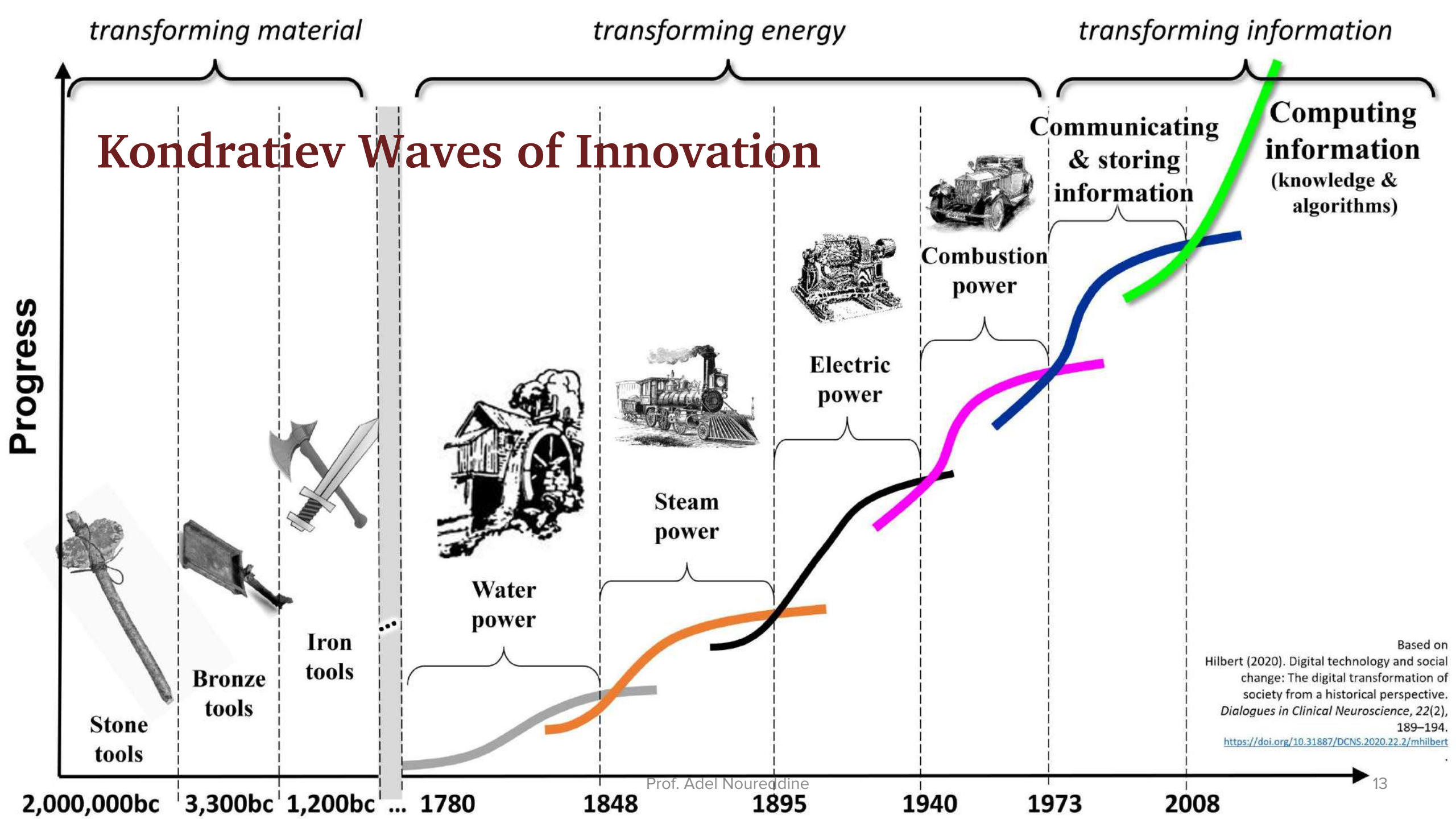
Energy Efficiency in Data Centers
(procurement, recommendations tool)

Product Manager @R&D Innova

★ **Green IT & software engineering**
★ **Behavioral computer science**
★ **Empirical software engineering**
★ **Autonomic computing**

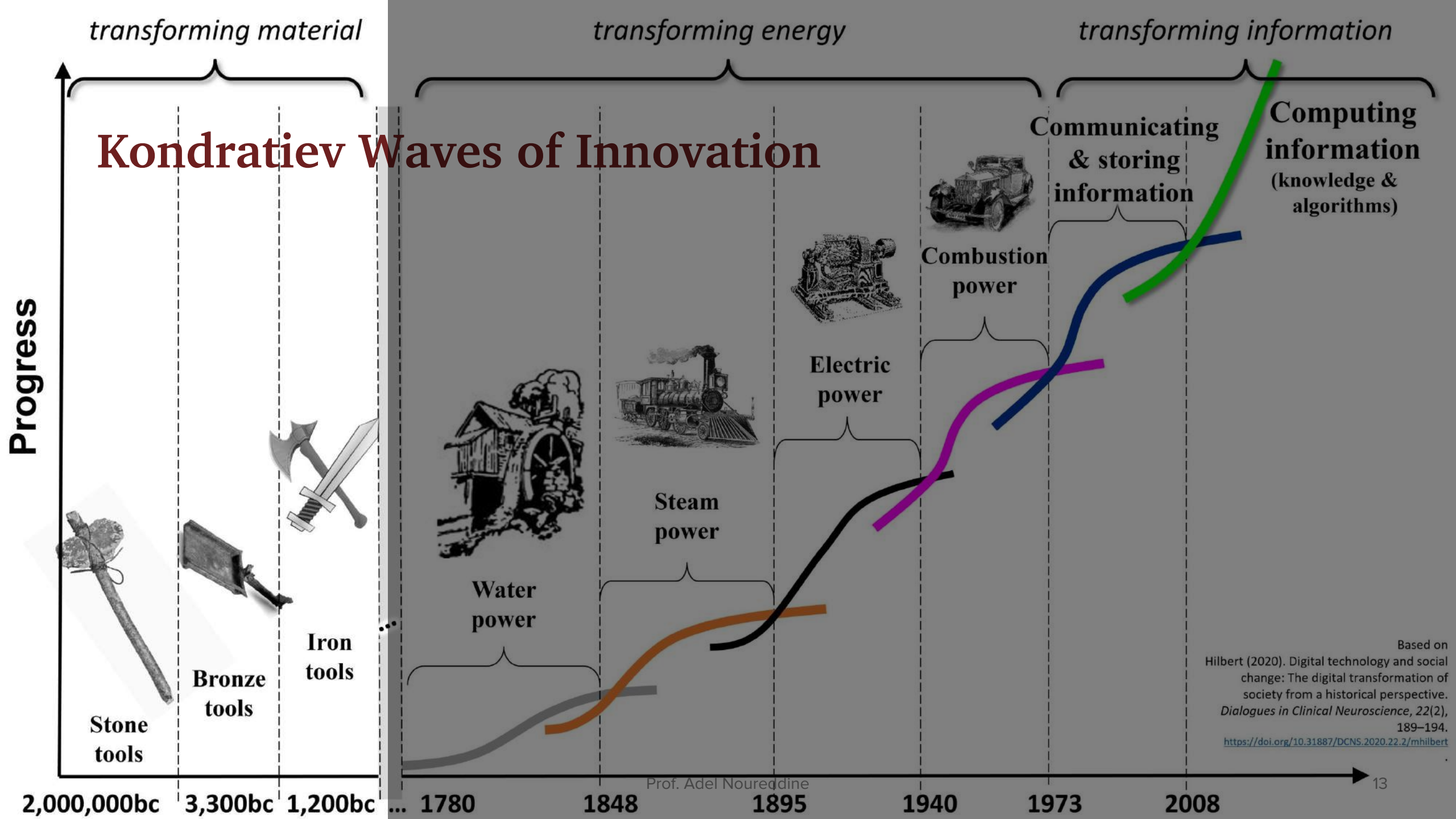
The “No Exit Dilemma”

Kondratiev Waves of Innovation



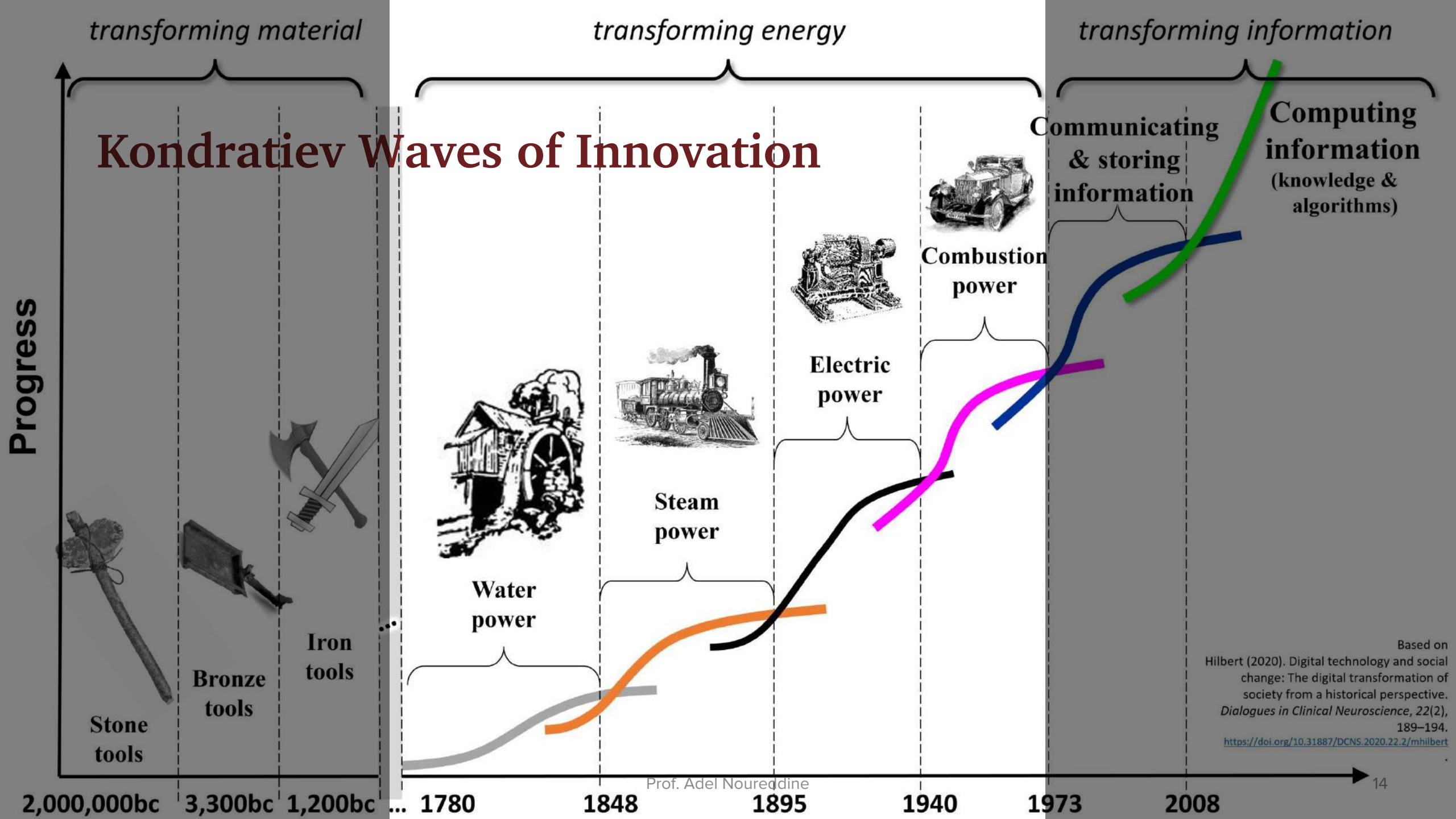
Based on Hilbert (2020). Digital technology and social change: The digital transformation of society from a historical perspective. *Dialogues in Clinical Neuroscience*, *22*(2), 189–194. https://doi.org/10.31887/DCNS.2020.22.2/mhilbert

Kondratiev Waves of Innovation



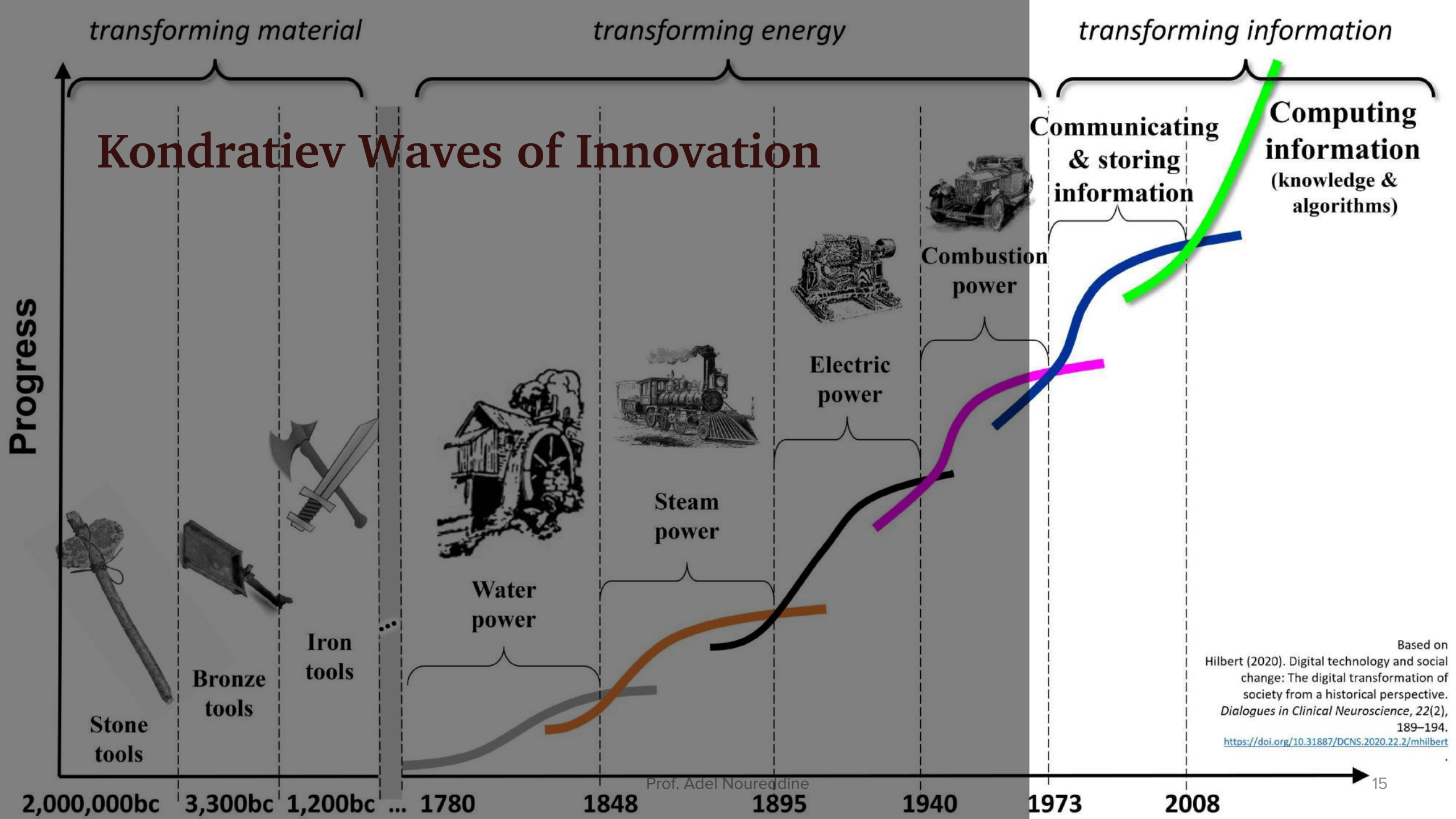
Based on Hilbert (2020). Digital technology and social change: The digital transformation of society from a historical perspective. *Dialogues in Clinical Neuroscience*, *22*(2), 189–194. https://doi.org/10.31887/DCNS.2020.22.2/mhilbert

Kondratiev Waves of Innovation



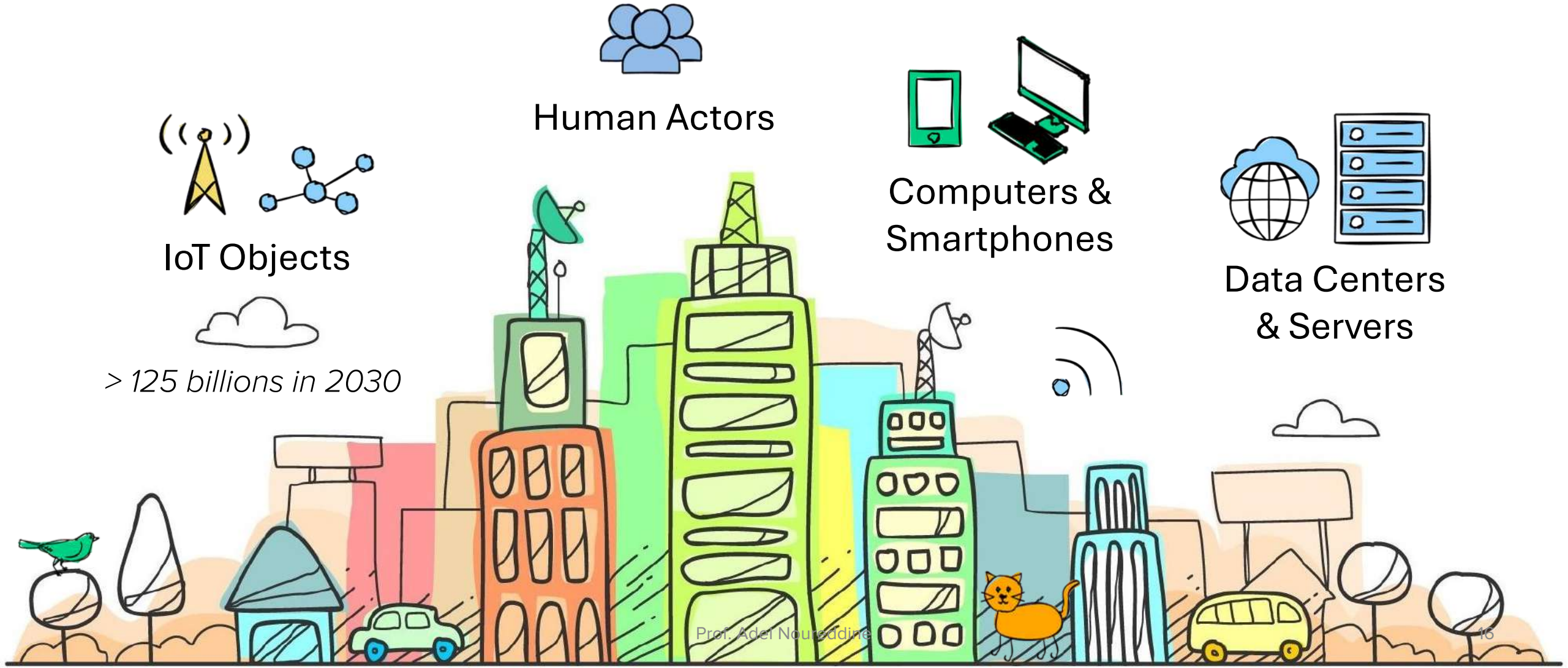
Based on Hilbert (2020). Digital technology and social change: The digital transformation of society from a historical perspective. *Dialogues in Clinical Neuroscience*, *22*(2), 189–194. https://doi.org/10.31887/DCNS.2020.22.2/mhilbert .

Kondratiev Waves of Innovation

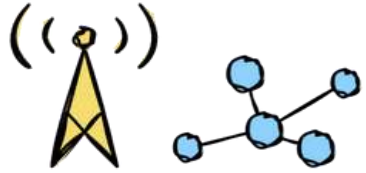


Based on Hilbert (2020). Digital technology and social change: The digital transformation of society from a historical perspective. *Dialogues in Clinical Neuroscience*, *22*(2), 189–194. https://doi.org/10.31887/DCNS.2020.22.2/mhilbert .

Software Ecosystems

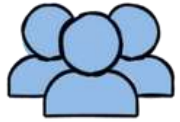


Software Ecosystems

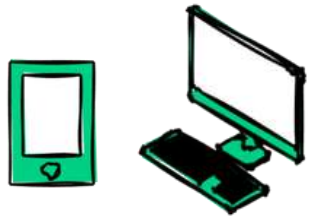


IoT Objects

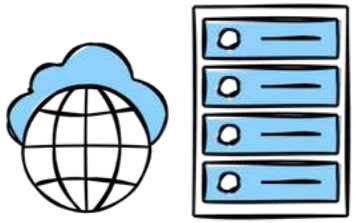
> 125 billions in 2030



Human Actors



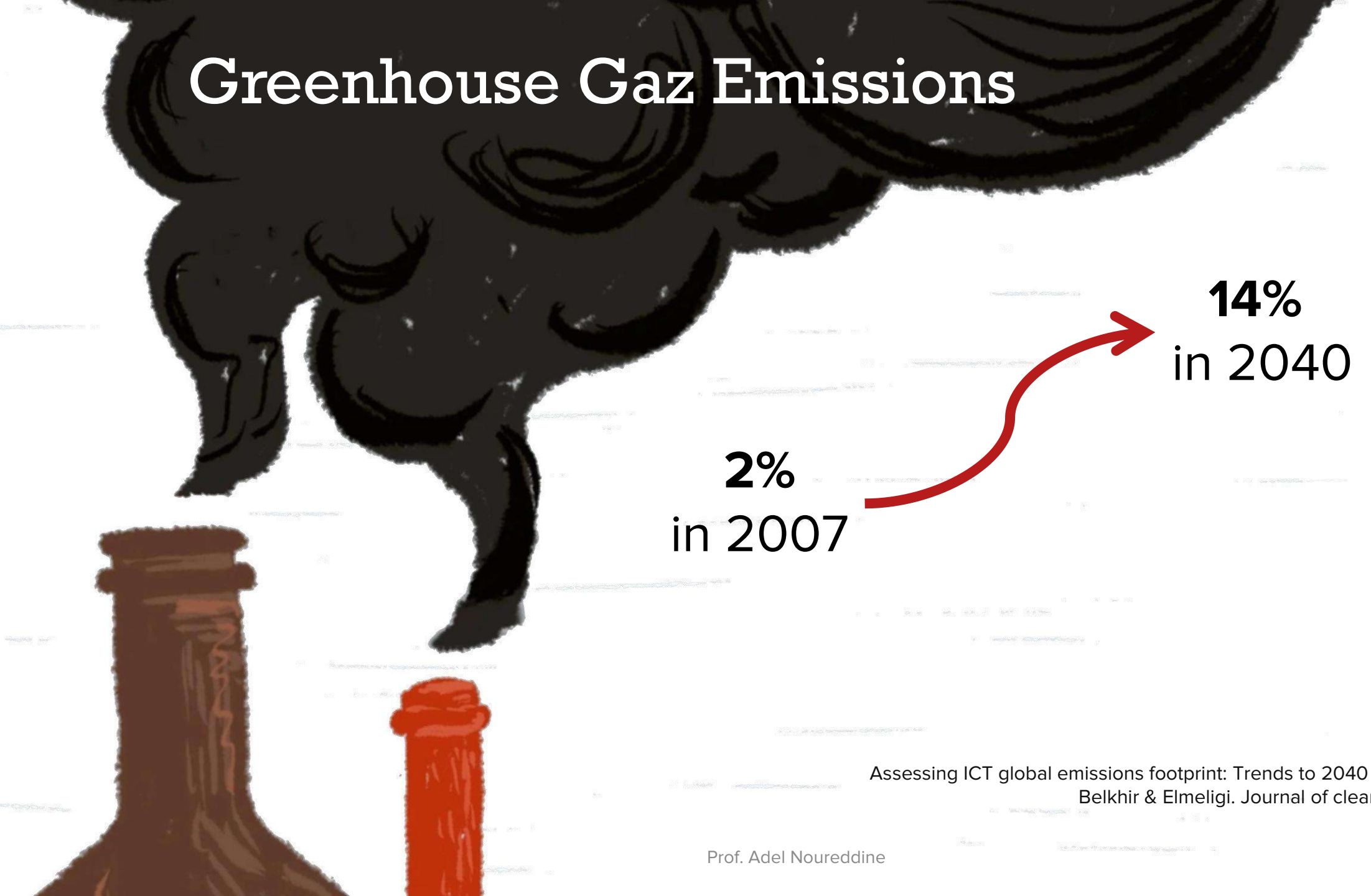
Computers & Smartphones



Data Centers & Servers

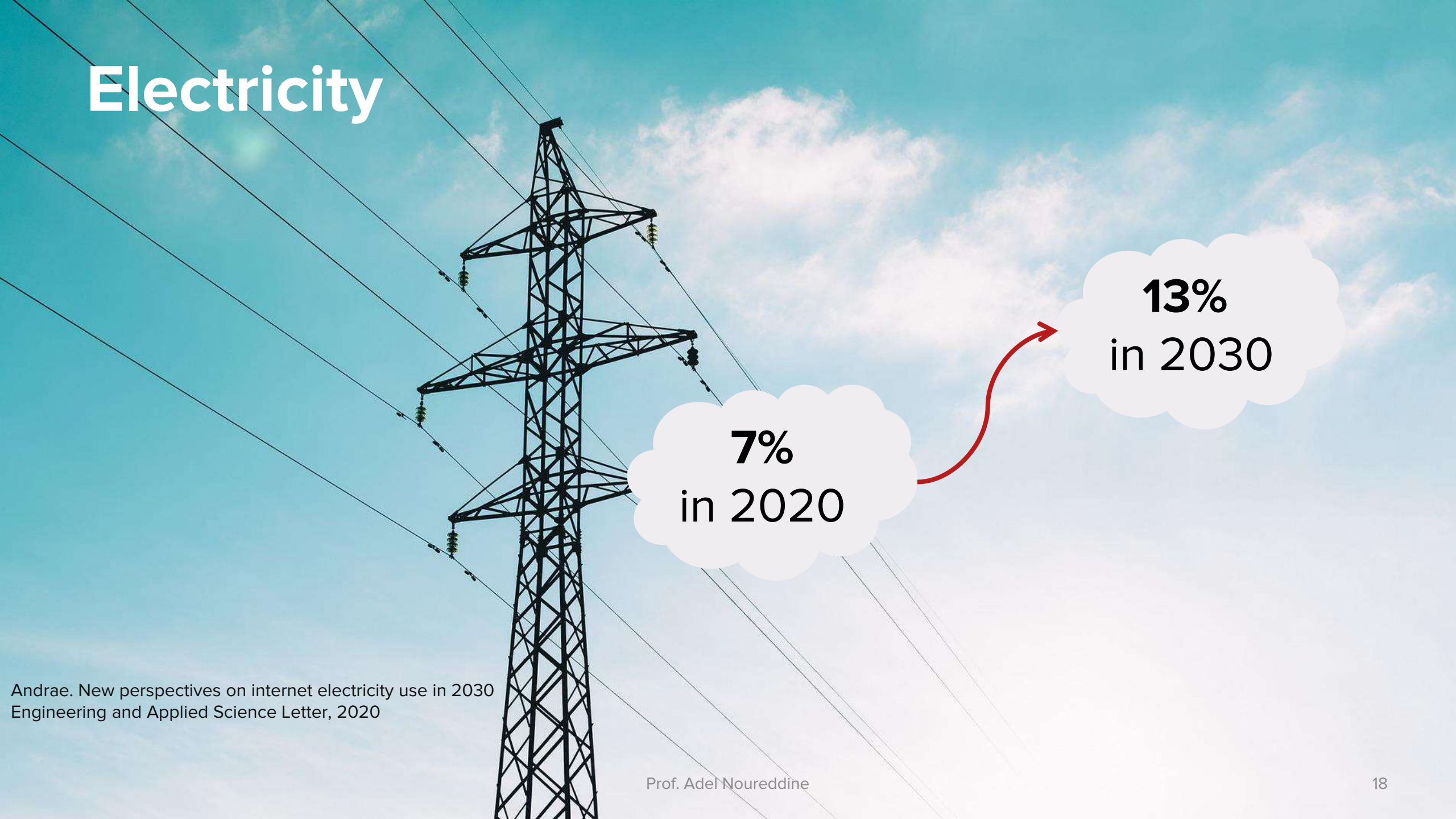
Our lives depend on software, our industry depends on software, our whole society depends on software (Software Heritage Foundation)

Greenhouse Gaz Emissions



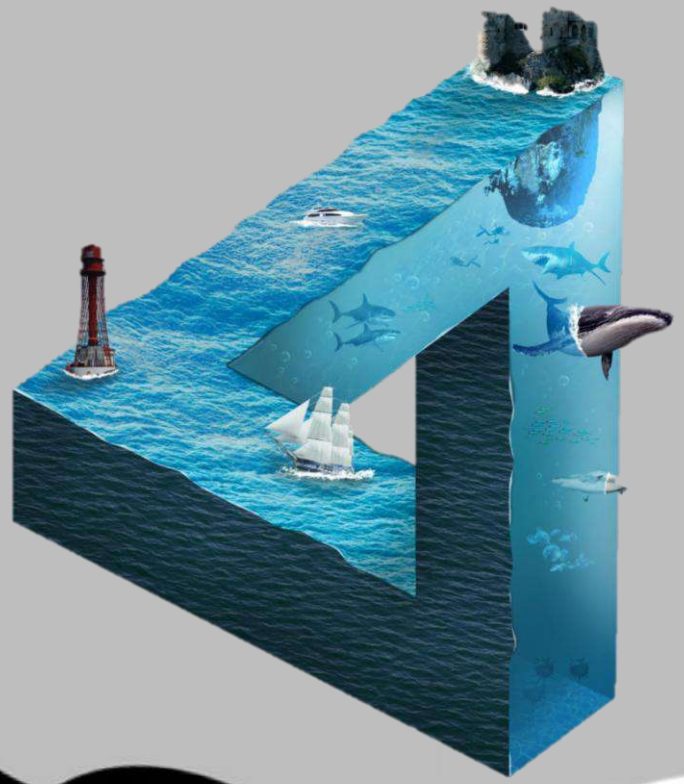
Assessing ICT global emissions footprint: Trends to 2040 & recommendations
Belkhir & Elmeligi. Journal of cleaner production, 2018

Electricity



Andrae. New perspectives on internet electricity use in 2030
Engineering and Applied Science Letter, 2020

We depend on ICT & software ecosystems



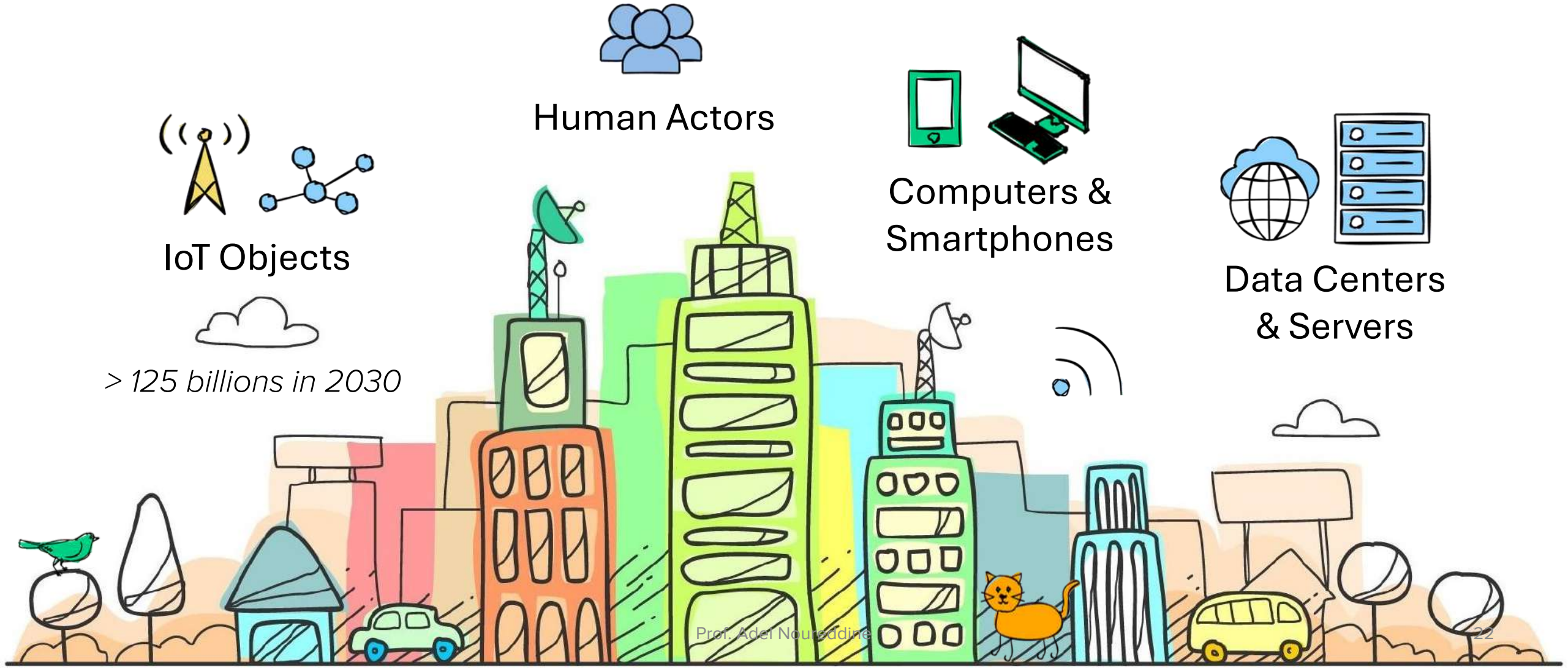
Rising energy, GHGE & environmental impacts

Optimizing ICT will only

“buy us time”

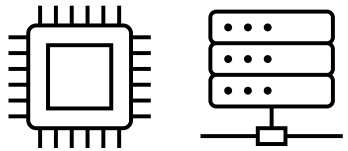
The Challenges of Heterogeneity

Software Ecosystems

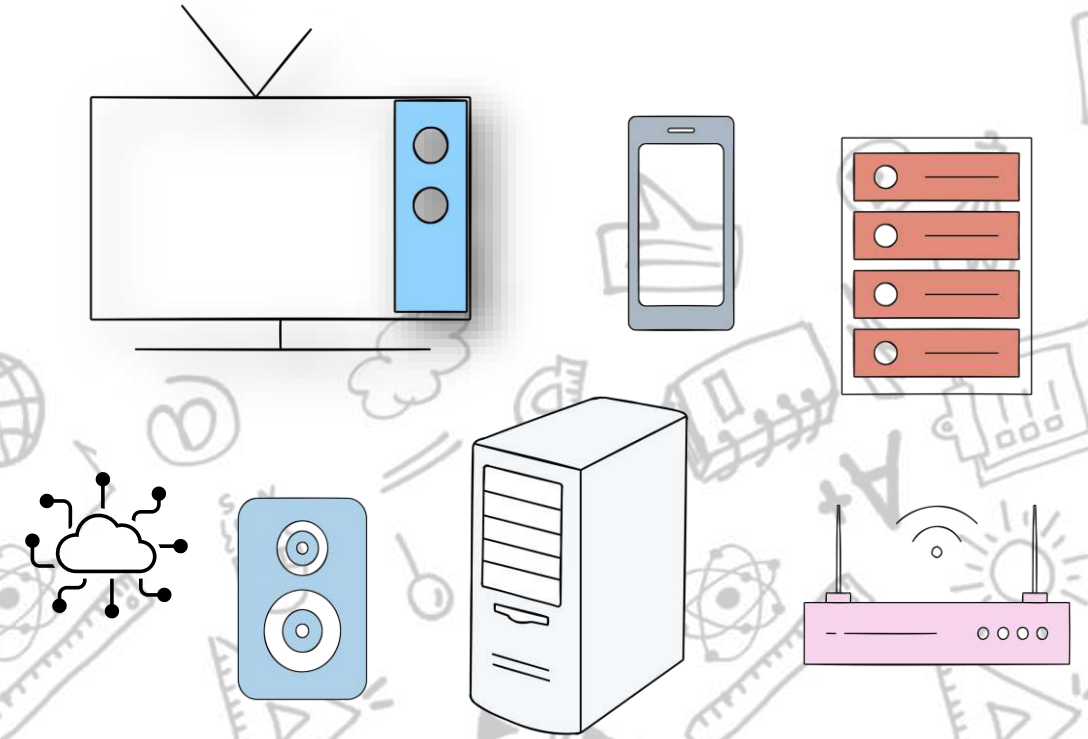


Heterogeneity in Hardware

IoT to data centers, cyber-physical systems

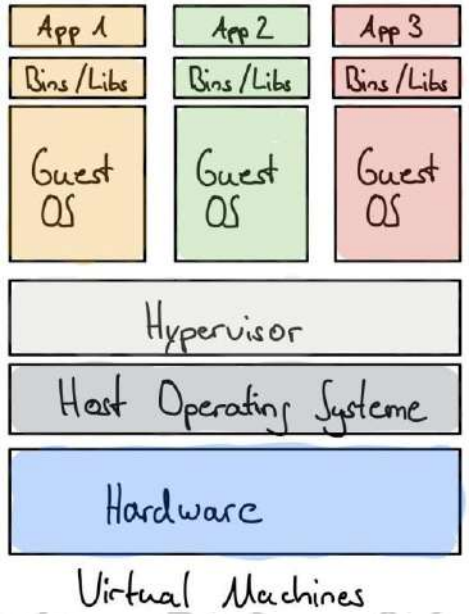


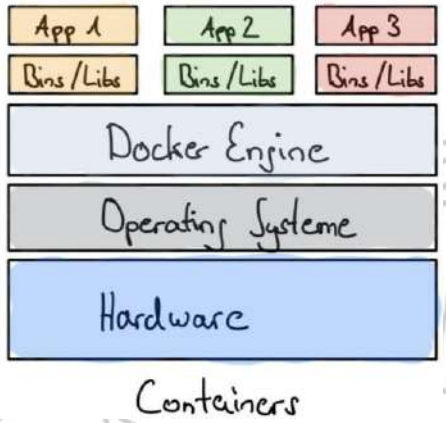
Debian CPU architectures		
	Alpha	mipsel
	Arm	mips64el
	Armel	PowerPC
	armhf	PowerSPE
	arm64	PPC64
	hppa	ppc64el
	i386	riscv64
	amd64	s390
	ia64	s390x
	m68k	SH4
	mips	sparc64



Heterogeneity in Software

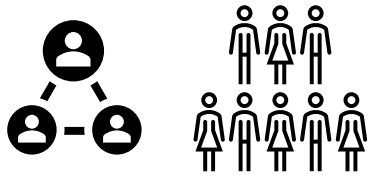
OS, VM, software versions, updates, configurations





Optimization	Included in level O1	O2	O3
-defer-pop	●	●	●
-thread-jumps	●	●	●
-branch-probabilities	●	●	●
-cprops-registers	●	●	●
-guess-branch-probability	●	●	●
-omit-frame-pointer	●	●	●
-align-loops	○	●	●
-align-jumps	○	●	●
-align-labels	○	●	●
-align-functions	○	●	●
-optimize-sibling-calls	○	●	●
-cse-follow-jumps	○	●	●
-cse-skip-blocks	○	●	●
-gcse	○	●	●
-expensive-optimizations	○	●	●
-strength-reduce	○	●	●

Heterogeneity in Humans



- ★ Project managers
- ★ Partners, sponsors
- ★ Developers, architects
- ★ Designers, UI/UX
- ★ Testers, analysts, Q&A
- ★ End users

US Adults' Social Platform Use, by Demographic Group

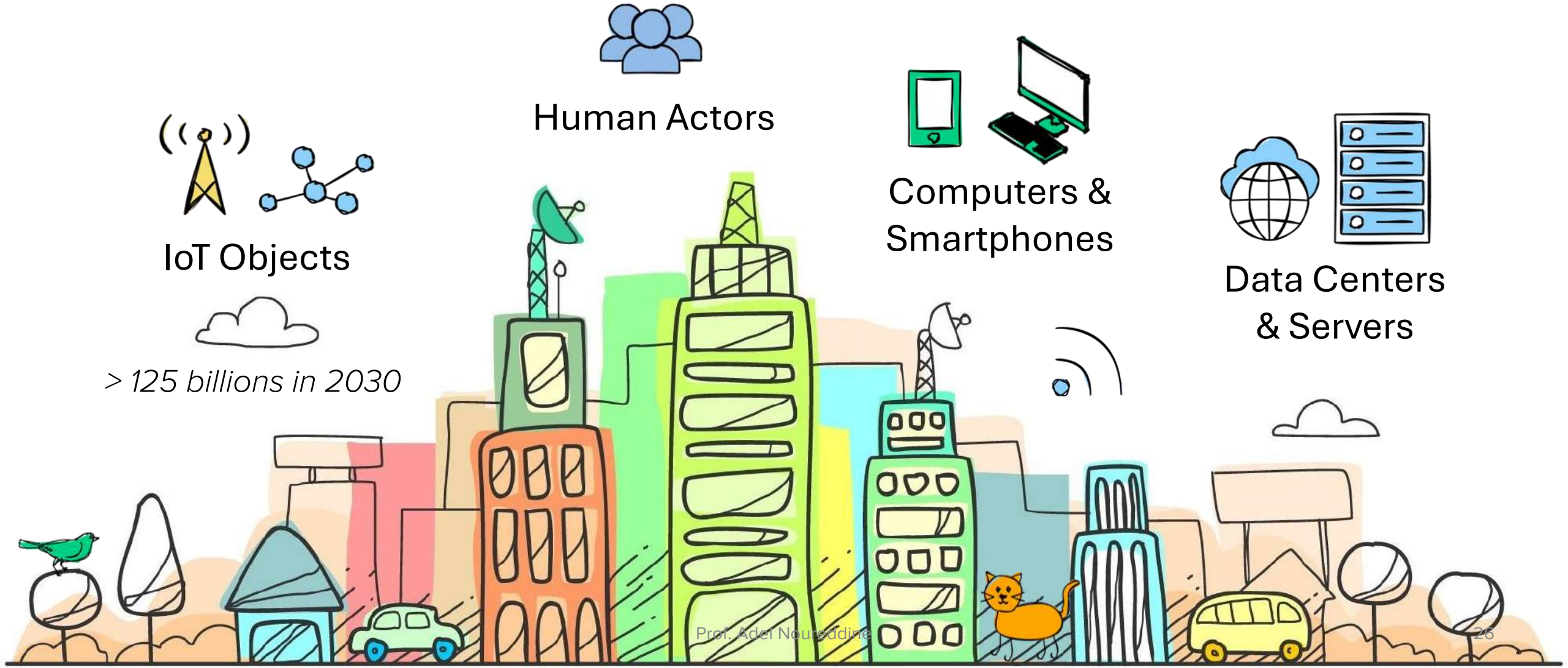
marketing charts

% of US adults who use:	YouTube	Facebook	Instagram	Pinterest	LinkedIn	Snapchat	Twitter	WhatsApp	Reddit
Total	73%	69%	37%	28%	27%	24%	22%	20%	11%
Men	78%	63%	31%	15%	29%	24%	24%	21%	15%
Women	68%	75%	43%	42%	24%	24%	21%	19%	8%
Non-Hispanic White	71%	70%	33%	33%	28%	22%	21%	13%	12%
Non-Hispanic Black	77%	70%	40%	27%	24%	28%	24%	24%	4%
Hispanic	78%	69%	51%	22%	16%	29%	25%	42%	14%
Ages 18-24	90%	76%	75%	38%	17%	73%	44%	20%	21%
Ages 25-29	93%	84%	57%	28%	44%	47%	31%	28%	23%
Ages 30-49	87%	79%	47%	35%	37%	25%	26%	31%	14%
Ages 50-64	70%	68%	23%	27%	24%	9%	17%	16%	6%
Ages 65+	38%	46%	8%	15%	11%	3%	7%	3%	1%
HHI: <$30k	68%	69%	35%	18%	10%	27%	20%	19%	9%
HHI: $30-75k	75%	72%	39%	27%	26%	26%	20%	16%	10%
HHI: $75k+	83%	74%	42%	41%	49%	22%	31%	25%	15%
High school or less	64%	61%	33%	19%	9%	22%	13%	18%	6%
Some college	79%	75%	37%	32%	26%	29%	24%	14%	14%
College+	80%	74%	43%	38%	51%	20%	32%	28%	15%
Urban	77%	73%	46%	30%	33%	29%	26%	24%	11%
Suburban	74%	69%	35%	30%	30%	20%	22%	19%	13%
Rural	64%	66%	21%	26%	10%	20%	13%	10%	8%

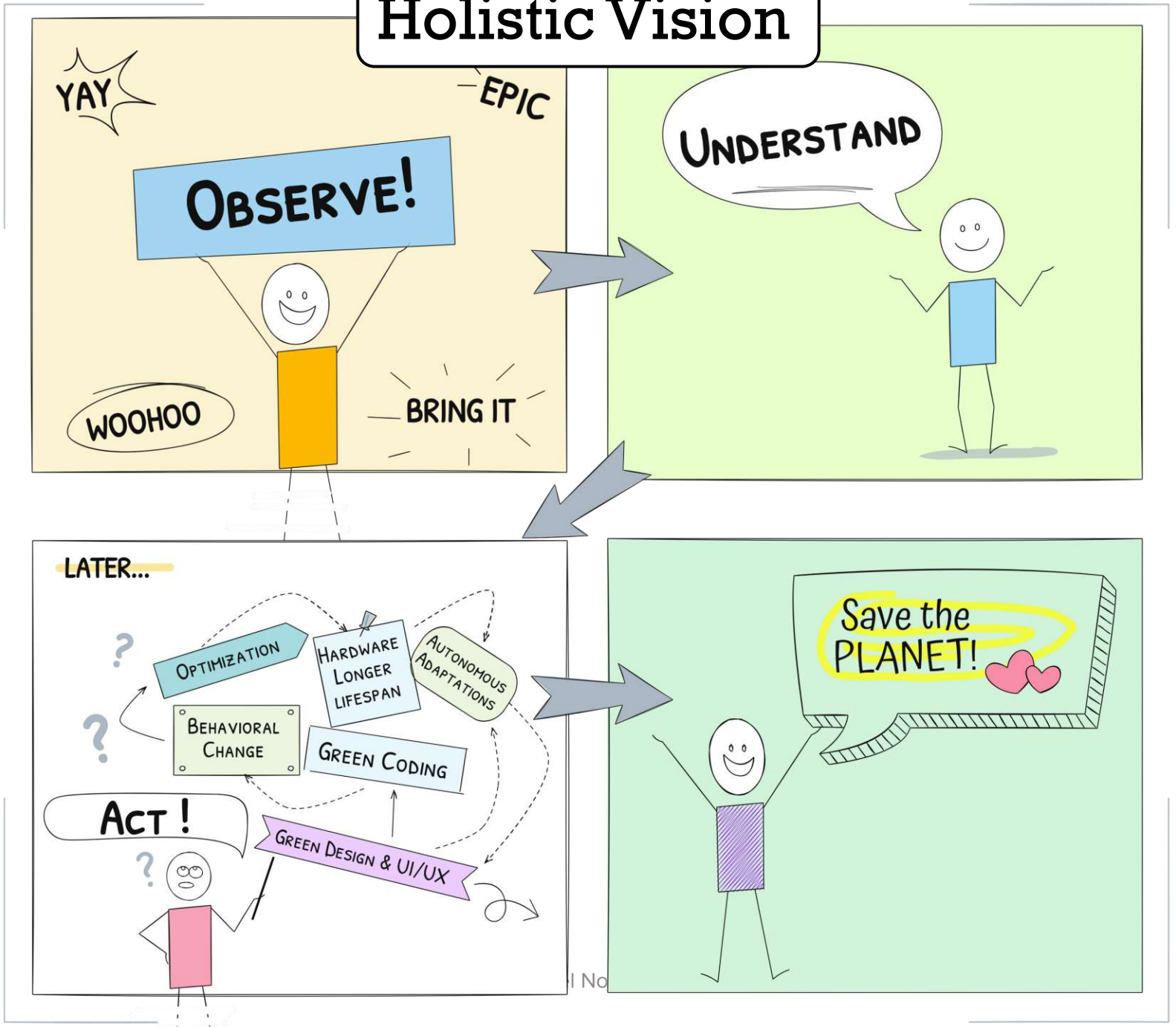
Published by MarketingCharts.com in April 2019 | Data Source: Pew Research Center

Based on telephone surveys conducted 1/8-2/7/19 among a national sample of 1,502 adults (18+)

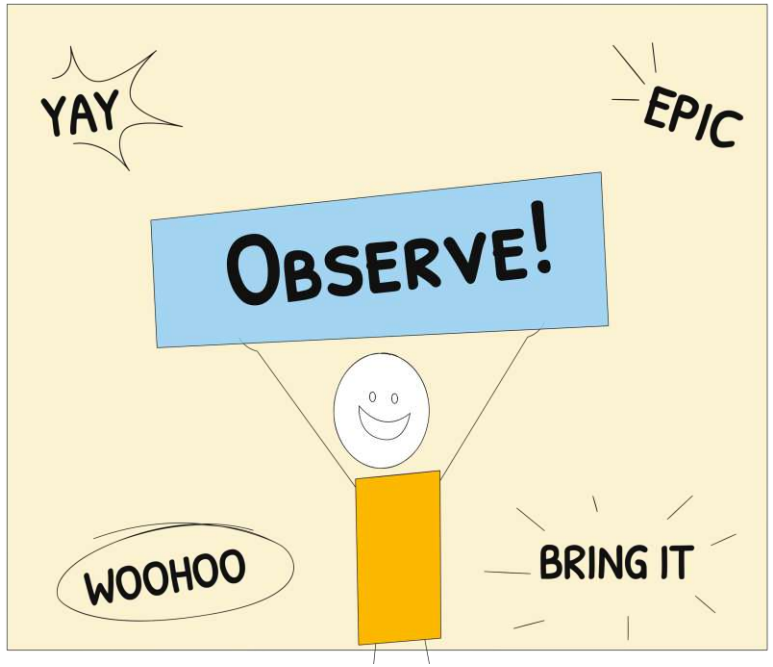
Software Ecosystems



Holistic Vision

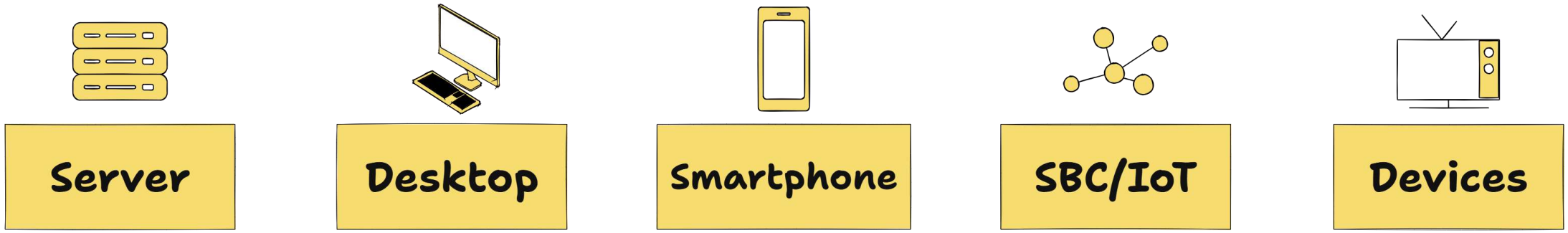


Observe the energy consumption and the environmental impacts of software ecosystems

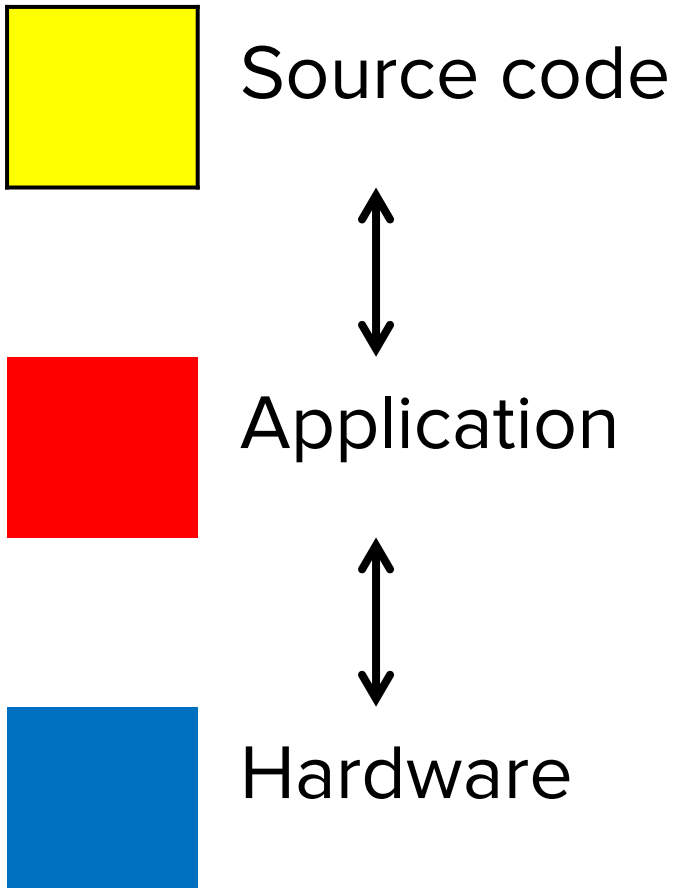


Everywhere
Anywhere
for Everyone

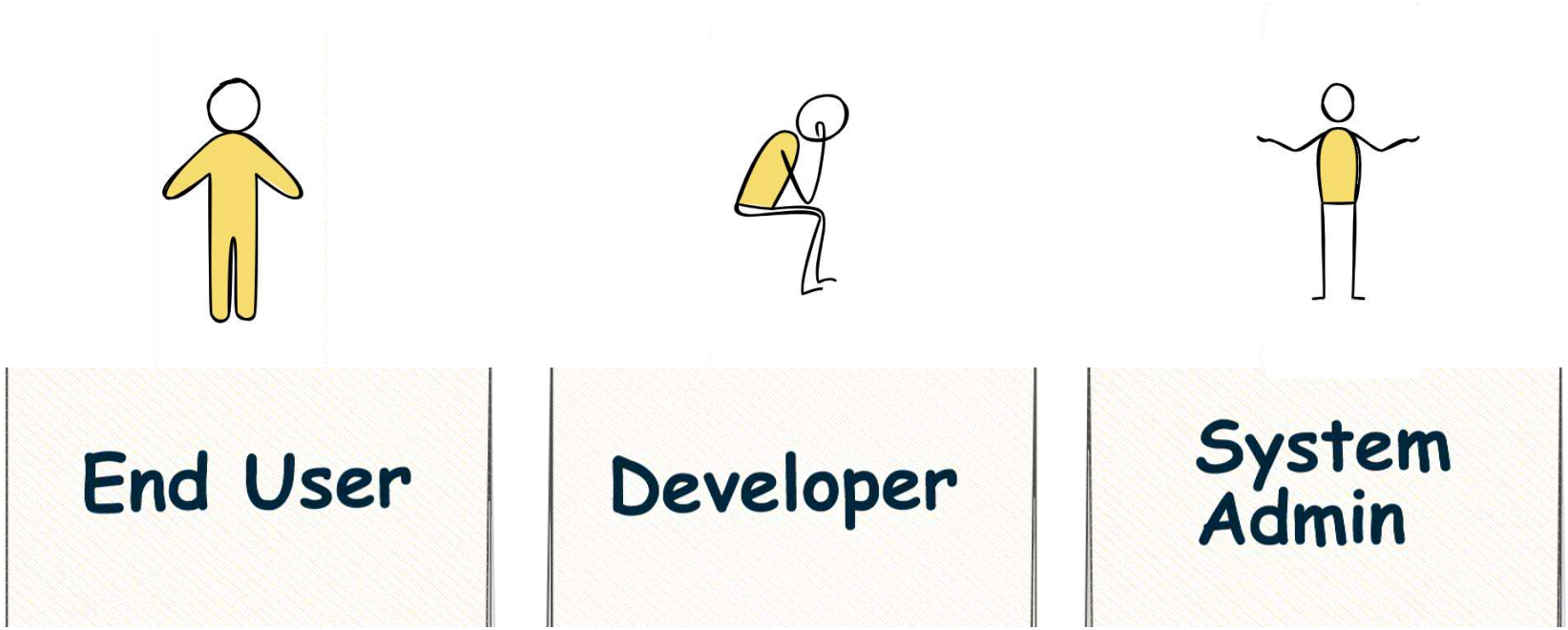
Everywhere

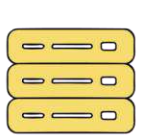


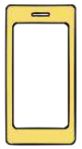
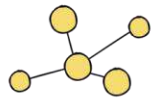
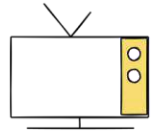
Anywhere



for **Everyone**





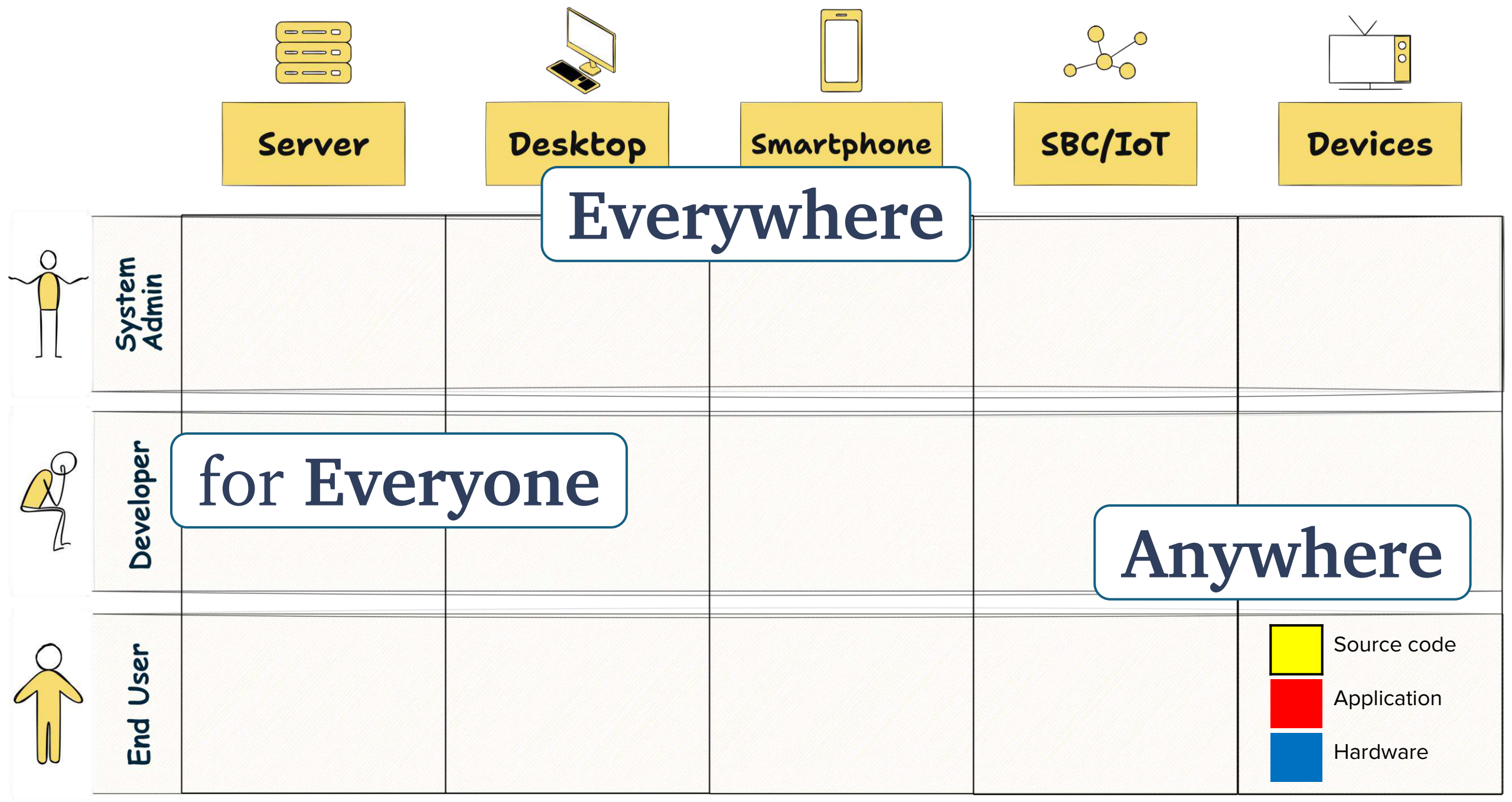




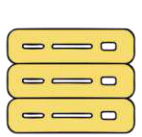
	Server	Desktop	Smartphone	SBC/IoT	Devices
System Admin					
Developer					
End User					

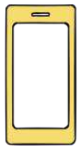
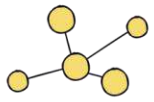
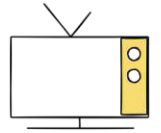
Source code

Application

Hardware


Server
Desktop
Smartphone
SBC/IoT
Devices
Everywhere
System Admin
Developer
for Everyone
Anywhere
End User
Source code
Application
Hardware



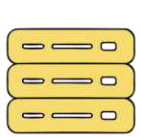




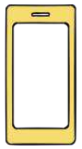
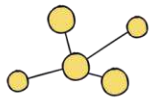
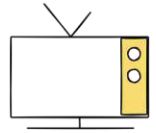
	Server	Desktop	Smartphone	SBC/IoT	Devices
System Admin					
Developer					
End User					

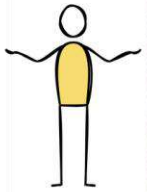
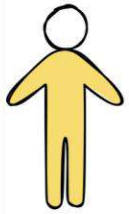
Source code

Application

Hardware



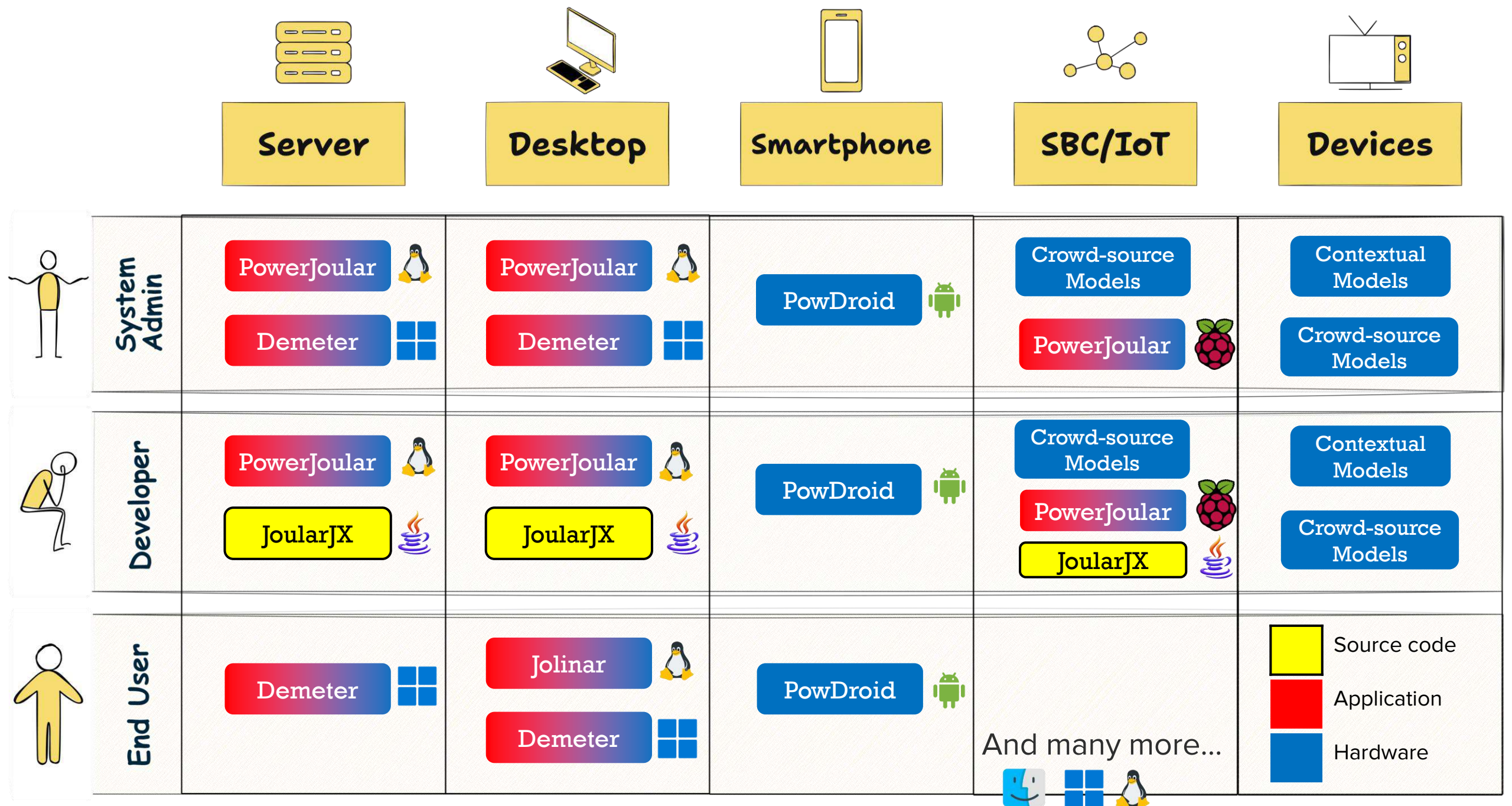
	Server	Desktop	Smartphone	SBC/IoT	Devices
 System Admin	PowerJoular, Demeter	PowerJoular, Demeter	PowDroid	Crowd-source Models, PowerJoular	Contextual Models, Crowd-source Models
 Developer	PowerJoular, JoularJX	PowerJoular, JoularJX	PowDroid	Crowd-source Models, PowerJoular, JoularJX	Contextual Models, Crowd-source Models
 End User	Demeter	Jolinar, Demeter	PowDroid	And many more...	Source code, Application, Hardware

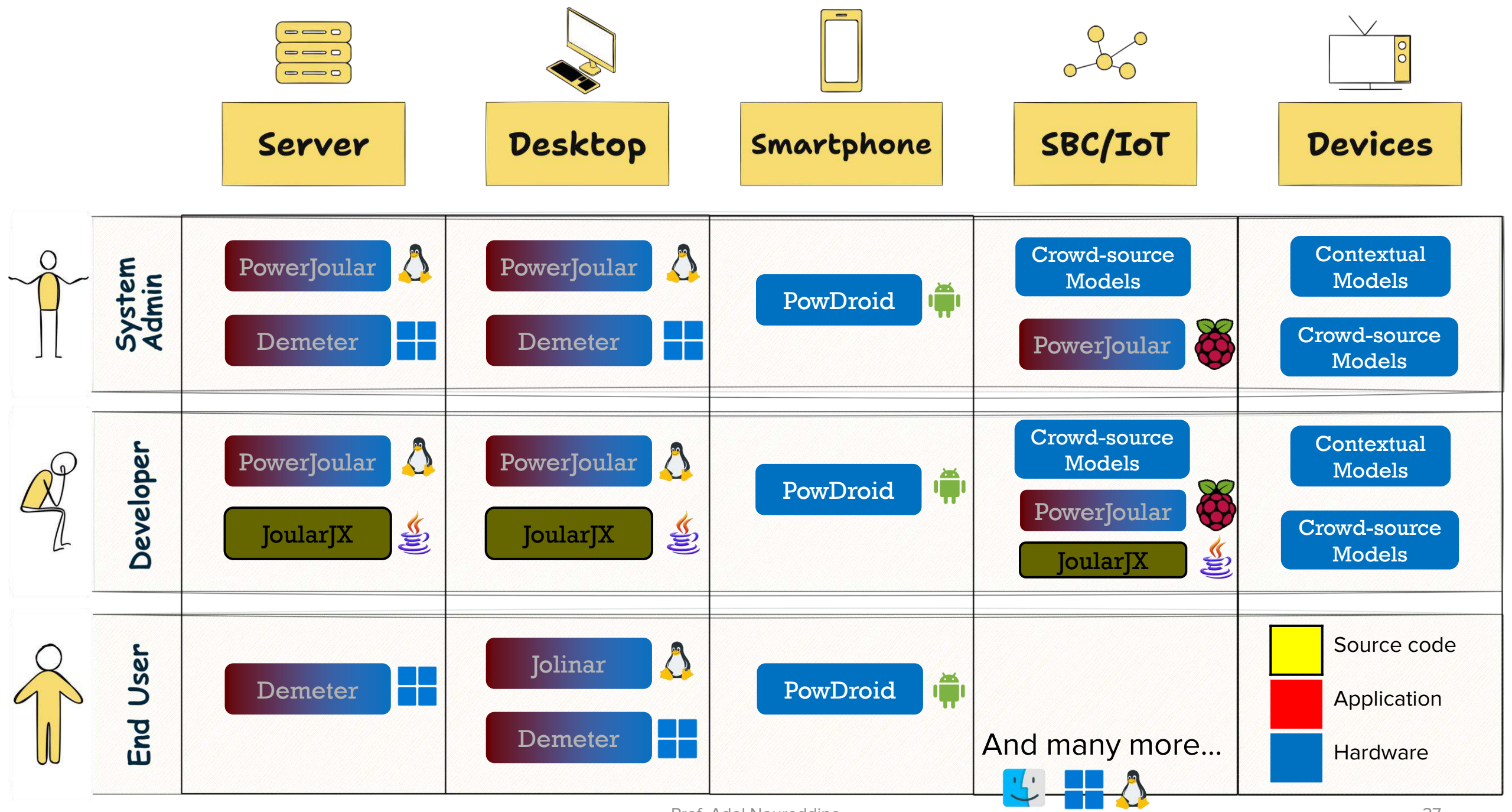
Our landscape, 2025

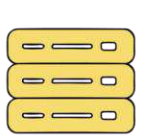
	Server	Desktop	Smartphone	SBC/IoT	Devices
System Admin	PowerJoular, Demeter	PowerJoular, Demeter	PowDroid	Crowd-source Models, PowerJoular	Contextual Models, Crowd-source Models
Developer	PowerJoular, JoularJX	JoularJX	PowDroid	Crowd-source Models, PowerJoular, JoularJX	Contextual Models, Crowd-source Models
End User	Demeter	Jolinar, Demeter	PowDroid	And many more...	

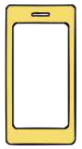
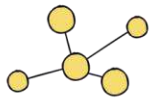
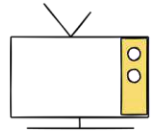
Source code
Application
Hardware

New Software

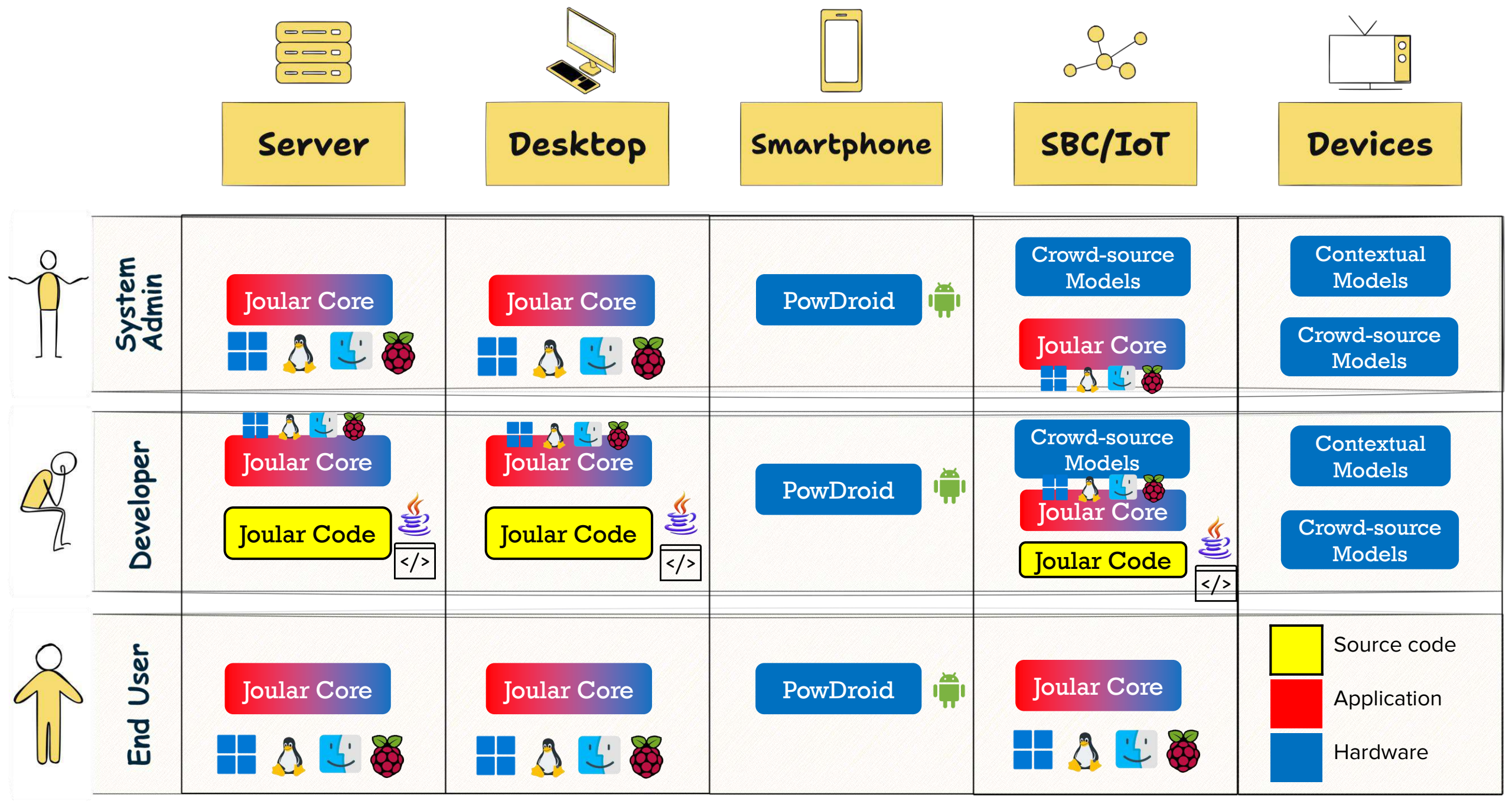

Server
Desktop
Smartphone
SBC/IoT
Devices
System Admin
PowerJoular
Demeter
PowerJoular
Demeter
PowDroid
Crowd-source Models
PowerJoular
Contextual Models
Crowd-source Models
Developer
PowerJoular
JoularJX
PowerJoular
JoularJX
PowDroid
Crowd-source Models
PowerJoular
JoularJX
Contextual Models
Crowd-source Models
End User
Demeter
Jolinar
Demeter
PowDroid
And many more...
Source code
Application
Hardware


Server
Desktop
Smartphone
SBC/IoT
Devices
System Admin
PowerJoular
Demeter
PowerJoular
Demeter
PowDroid
Crowd-source Models
PowerJoular
Contextual Models
Crowd-source Models
Developer
PowerJoular
JoularJX
PowerJoular
JoularJX
PowDroid
Crowd-source Models
PowerJoular
JoularJX
Contextual Models
Crowd-source Models
End User
Demeter
Jolinar
Demeter
PowDroid
And many more...
Source code
Application
Hardware



	Server	Desktop	Smartphone	SBC/IoT	Devices
System Admin			PowDroid	Crowd-source Models	Contextual Models Crowd-source Models
Developer			PowDroid	Crowd-source Models	Contextual Models Crowd-source Models
End User			PowDroid		Source code Application Hardware


Server
Desktop
Smartphone
SBC/IoT
Devices
System Admin
Developer
End User
Joular Core
PowDroid
Crowd-source Models
Contextual Models
Joular Code
Source code
Application
Hardware

Joular Project

Joular Project

- Advance knowledge in software energy efficiency throughout the life cycle of software & across a variety of software systems and devices
- Build solid and safe software to monitor and estimate the energy & power of hardware components & software code
- Understand the factors impacting energy consumption of software and devices
- Since 2020 *(heritage dating back to 2010)*

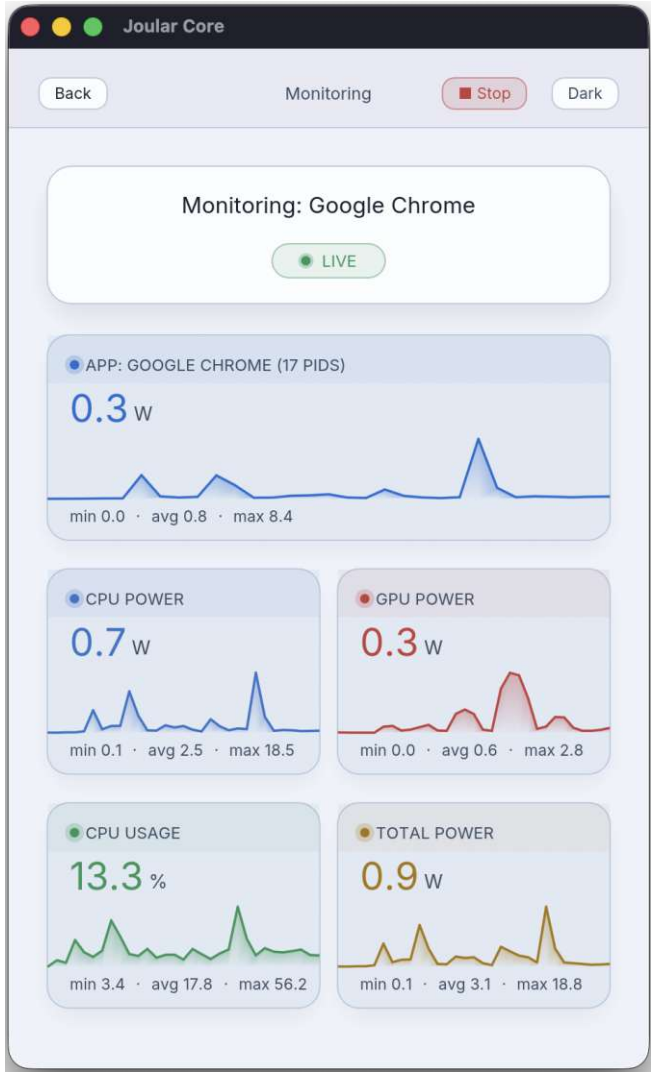
Joular Project: select software

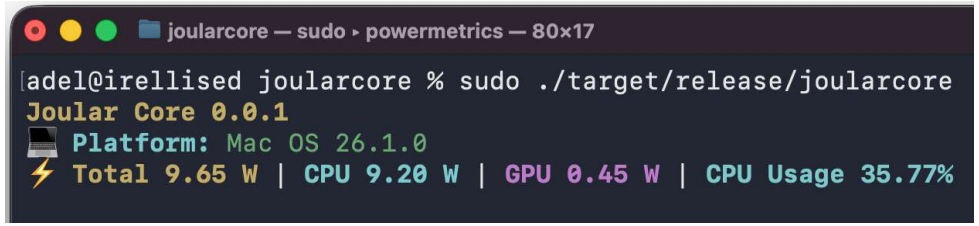
- **Joular Core**: a platform to measure power and energy across all systems and devices
- **Joular Code – Java**: a lightweight and efficient Java agent for monitoring the energy consumption of methods and execution branches at the source code level
- **PowerJoular**: monitor power consumption of hardware and software
- **JoularJX**: a Java-based agent for power monitoring at the source code level
- **CPU Power Benchmark:** an automated benchmark to accurately generate a power model for single-board computers
- **Power Models Database**: a database containing power models for various computing components, and hardware devices

Joular Core

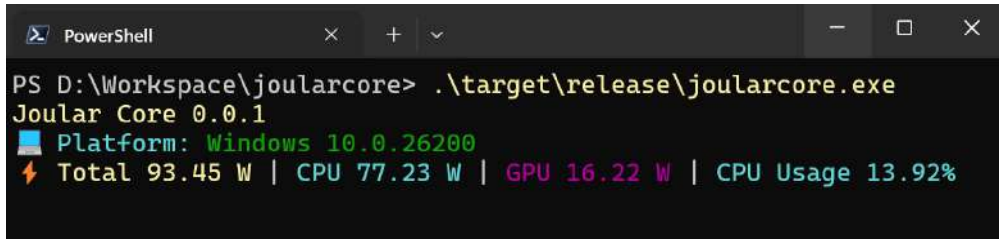
- Joular Core is a platform to measure power and energy across all systems, OSes and devices → **Everywhere**
- Universal energy measurement platform (*everywhere*)
- Works on: Linux, Windows, MacOS, Raspberry Pi, Virtual Machines
- Supported Architectures: x86_64 (amd64), x86/i686, aarch64, arm, armv7, GPUs (Nvidia, Apple, AMD)
- Monitors hardware, processes, & multi-process applications
- Low overhead, Rust-based, CLI, & GUI
- Open source: github.com/joular/joularcore
- noureddine.org/research/joular/joularcore

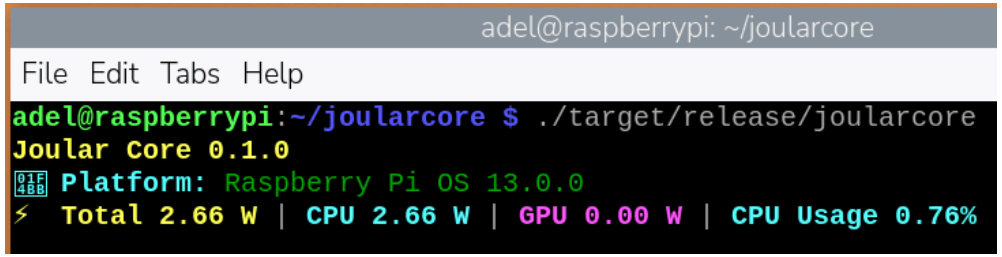
JoularCore


Joular Core
Back
Monitoring
Stop
Dark
Monitoring: Google Chrome
LIVE
APP: GOOGLE CHROME (17 PIDS)
0.3 W
min 0.0 · avg 0.8 · max 8.4
CPU POWER
0.7 W
min 0.1 · avg 2.5 · max 18.5
GPU POWER
0.3 W
min 0.0 · avg 0.6 · max 2.8
CPU USAGE
13.3 %
min 3.4 · avg 17.8 · max 56.2
TOTAL POWER
0.9 W
min 0.1 · avg 3.1 · max 18.8

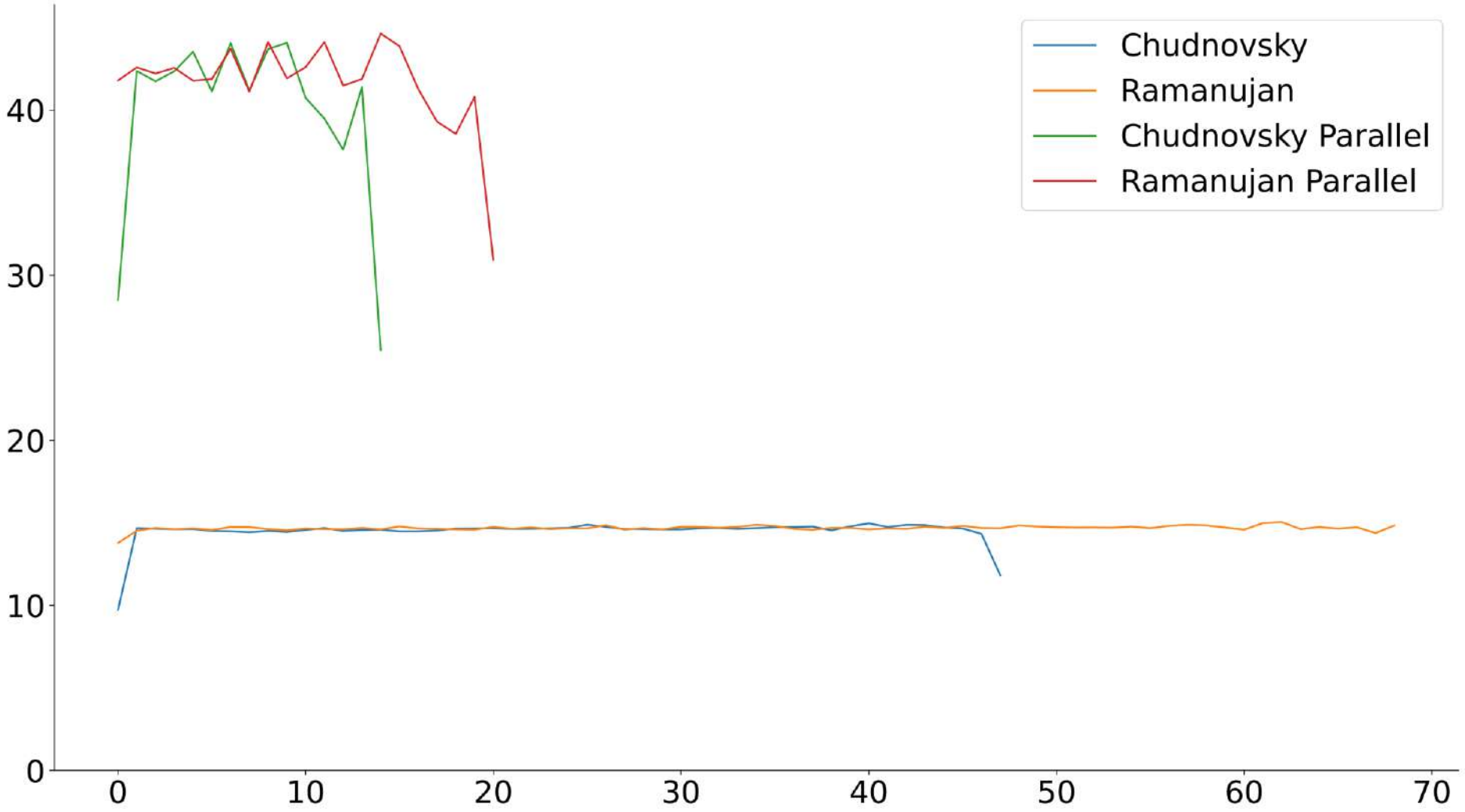

joularcore — sudo ▸ powermetrics — 80×17
adel@irellised joularcore % sudo ./target/release/joularcore
Joular Core 0.0.1
Platform: Mac OS 26.1.0
Total 9.65 W | CPU 9.20 W | GPU 0.45 W | CPU Usage 35.77%


adel@linux: ~ — sudo ./joularcore
adel@linux:~$ sudo ./joularcore
Joular Core 0.0.1
Platform: Ubuntu 25.10.0
Total 9.75 W | CPU 9.75 W | GPU 0.00 W | CPU Usage 1.72%

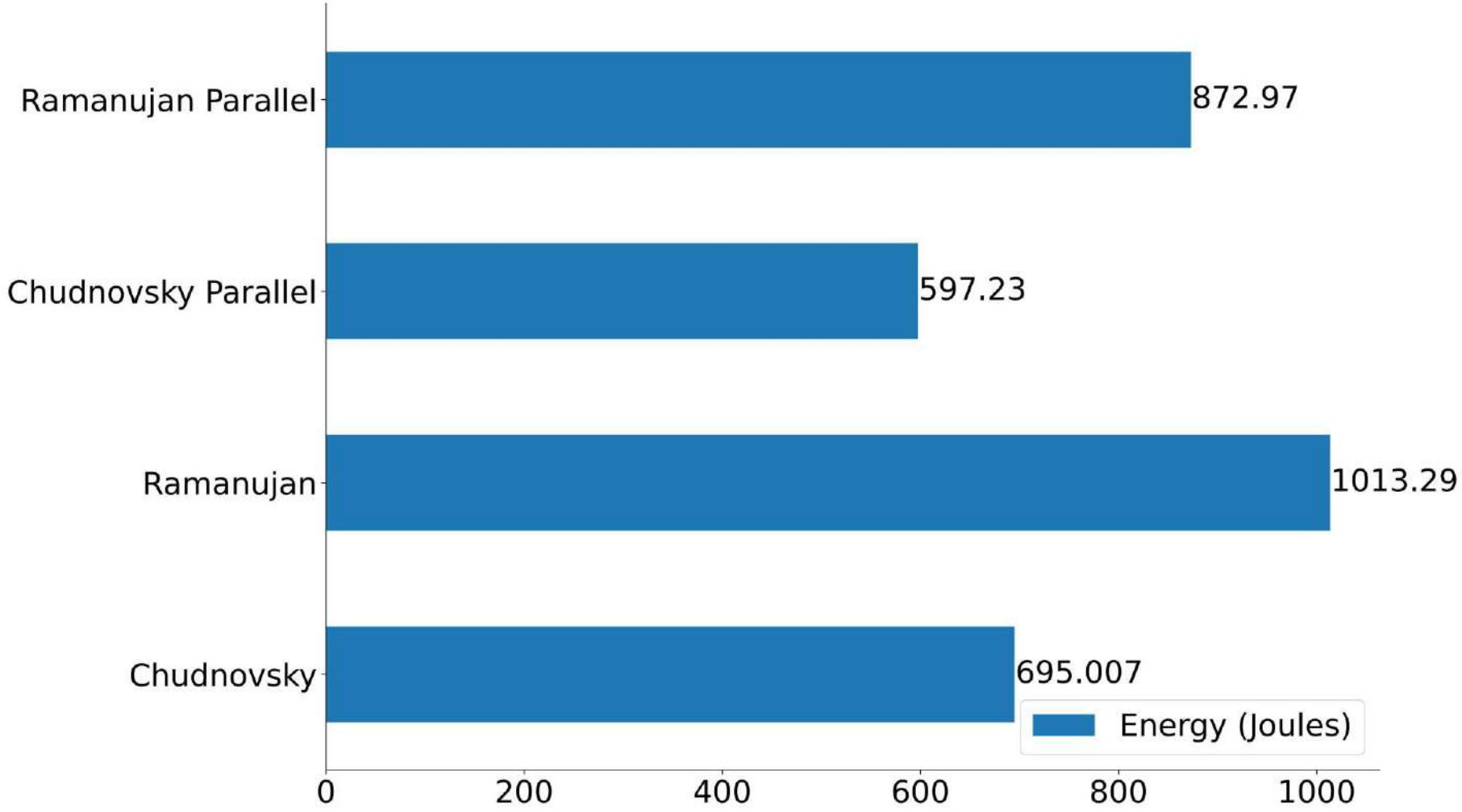

PowerShell
PS D:\Workspace\joularcore> .\target\release\joularcore.exe
Joular Core 0.0.1
Platform: Windows 10.0.26200
Total 93.45 W | CPU 77.23 W | GPU 16.22 W | CPU Usage 13.92%


adel@raspberrypi: ~/joularcore
File Edit Tabs Help
adel@raspberrypi:~/joularcore $ ./target/release/joularcore
Joular Core 0.1.0
Platform: Raspberry Pi OS 13.0.0
Total 2.66 W | CPU 2.66 W | GPU 0.00 W | CPU Usage 0.76%

150 millions digits of Pi



150 millions digits of Pi



Joular Code

Source Code Energy Monitoring

- Monitors energy of source code: methods, functions, classes, packages… → **Anywhere**
- 2 main approaches in existing literature: code instrumentation & power sampling
- Our approach: power sampling
  - Avoid burden on software developers (source code modification)
  - Avoid impact on software performance (binary/bytecode instrumentation)

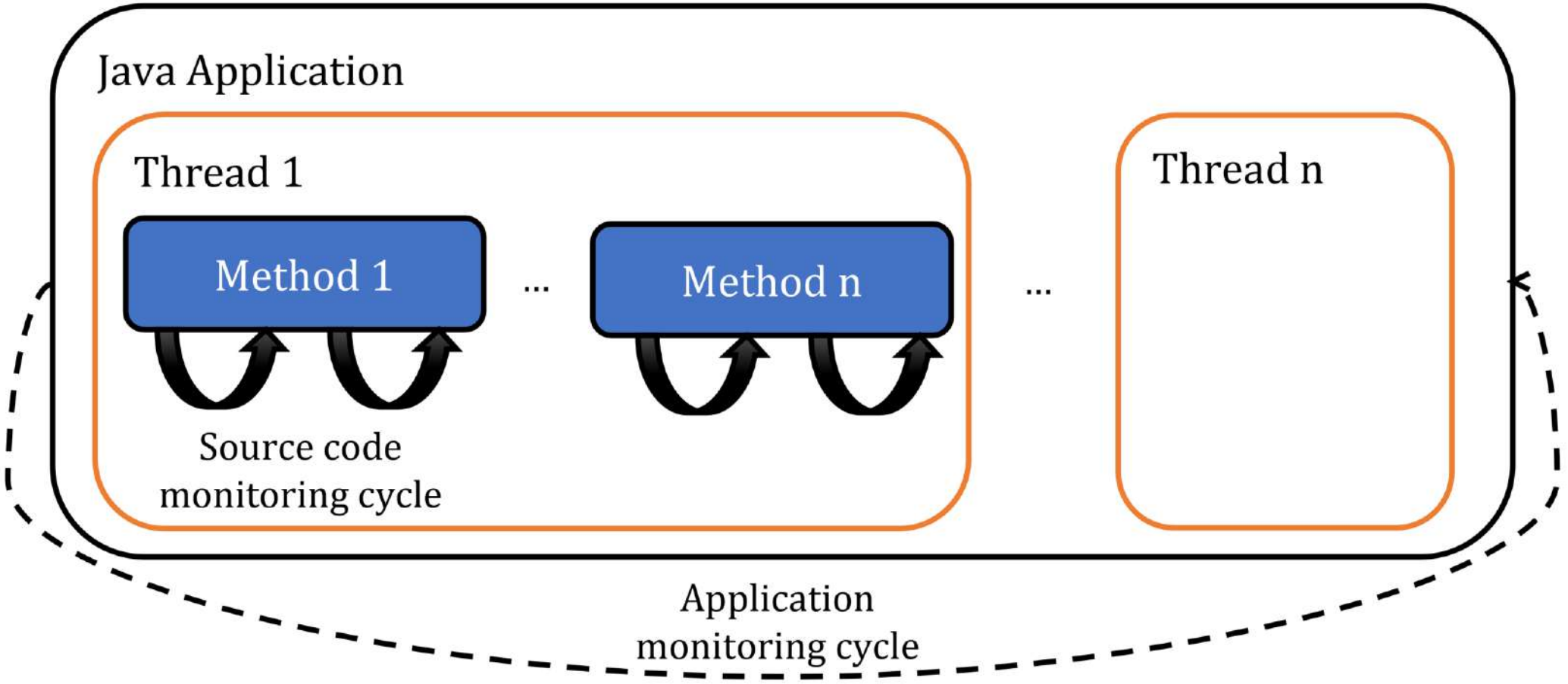
Joular Code for Java applications

- A new Java agent based on the lessons learned from JoularJX
- From legacy of our first proposal back in 2011: Jalen
- Monitors power of every execution branch in realtime
- Works everywhere: Windows, macOS, Linux, Raspberry Pi
- Samples the JVM stack at high frequency *(default: every 10 ms)* and attributes energy every second
- Generates CSV output files per run
- Gets CPU power from Joular Core backend
- LGPL 3 license, https://github.com/joular/joularcode-java

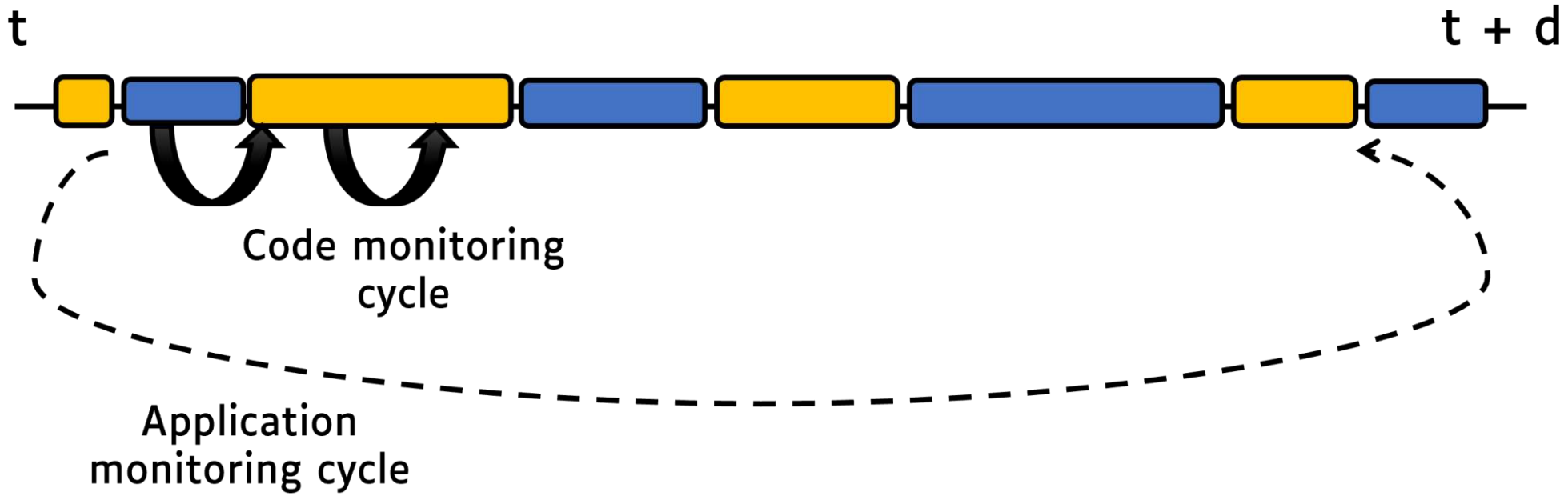
Power sampling

- A lighter, yet effective, alternative is power sampling from data collected from the application, OS and context
- Most accurate in virtual machine-based languages, such as Java
- Java Virtual Machine (JVM) provides profiling features that allows to collect additional metrics software behavior
- No source code or binary modification of the application
- Approach adopted in: Jalen (2012), JoularJX (2021), and Joular Code Java (2026)

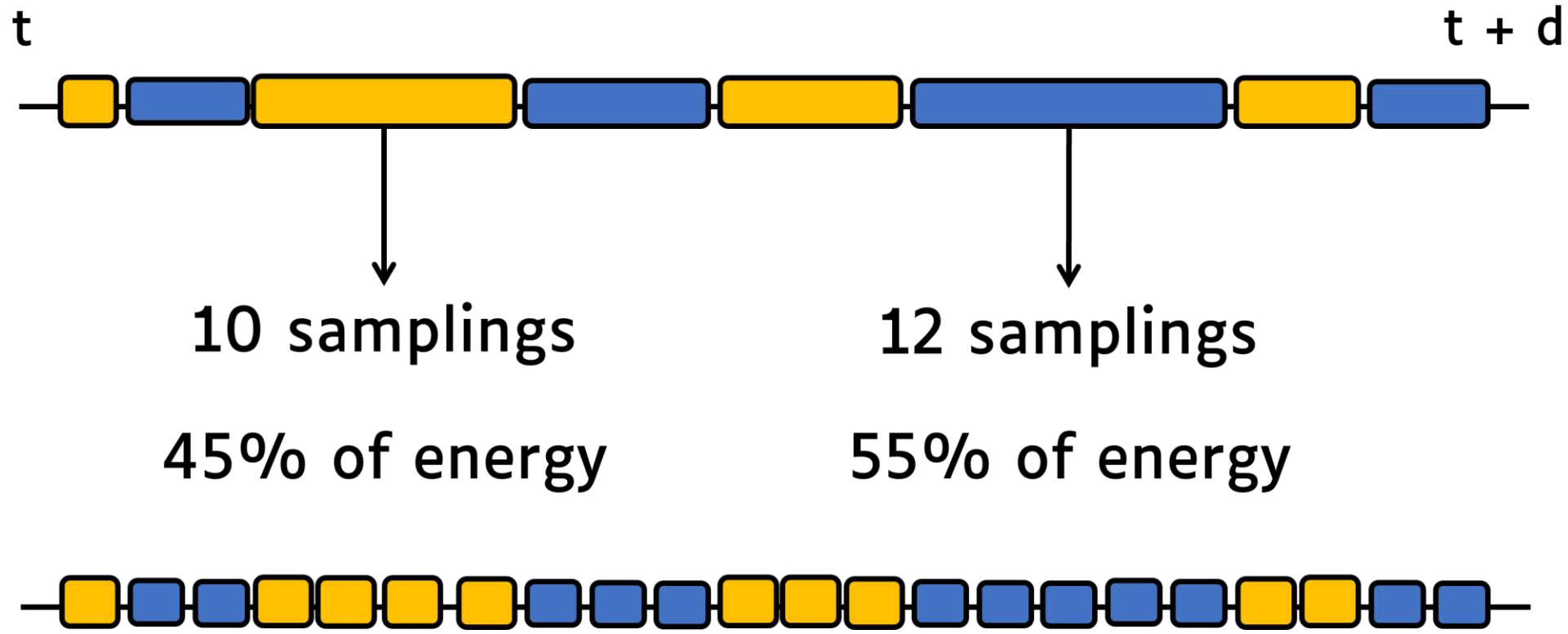
JoularJX and Joular Code Java approach



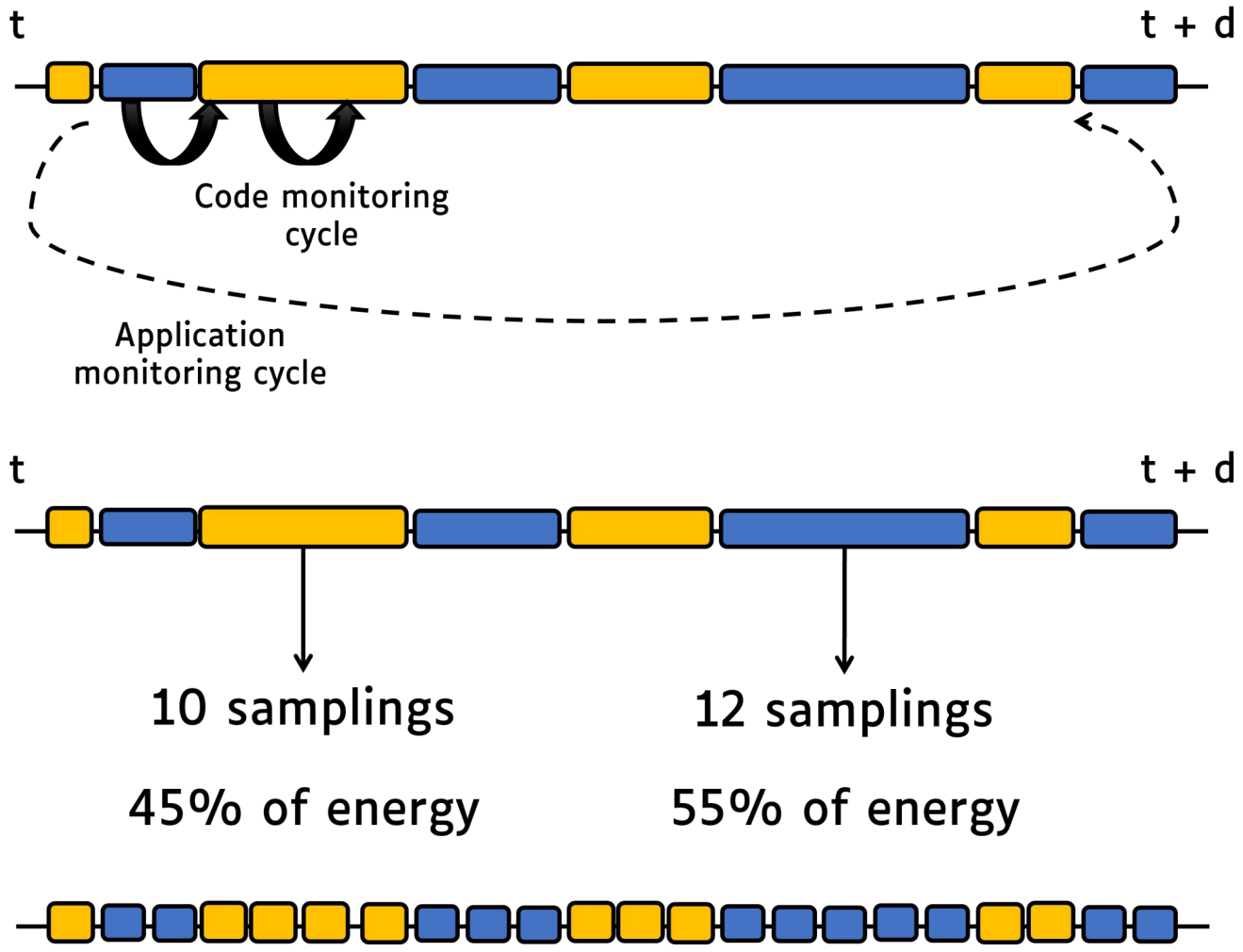
JoularJX and Joular Code Java approach



JoularJX and Joular Code Java approach



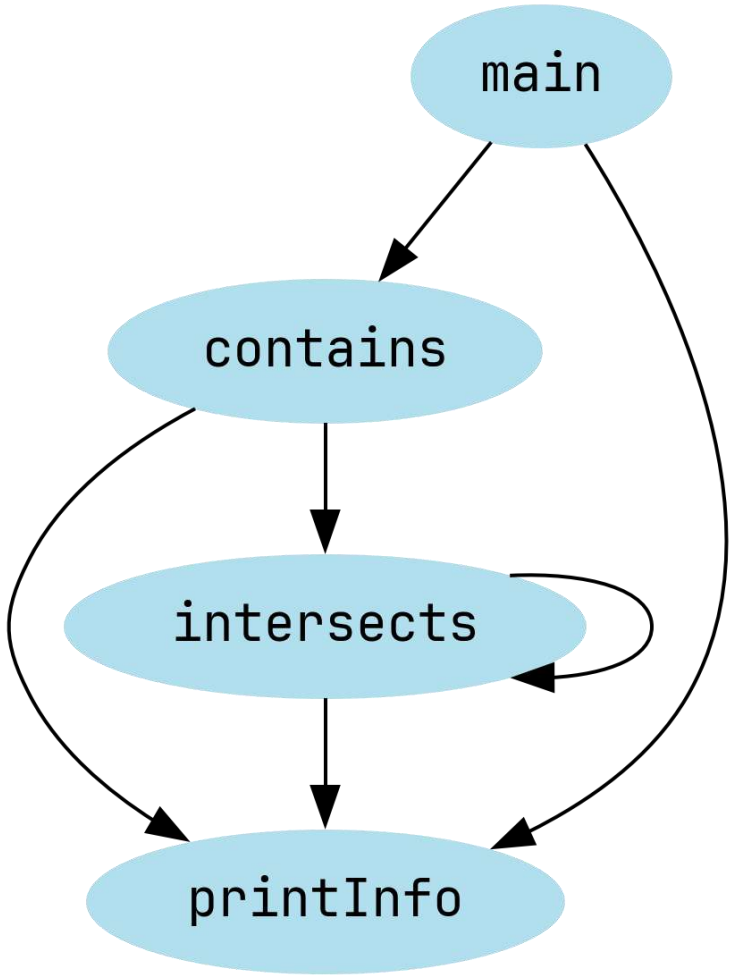
JoularJX and Joular Code Java approach



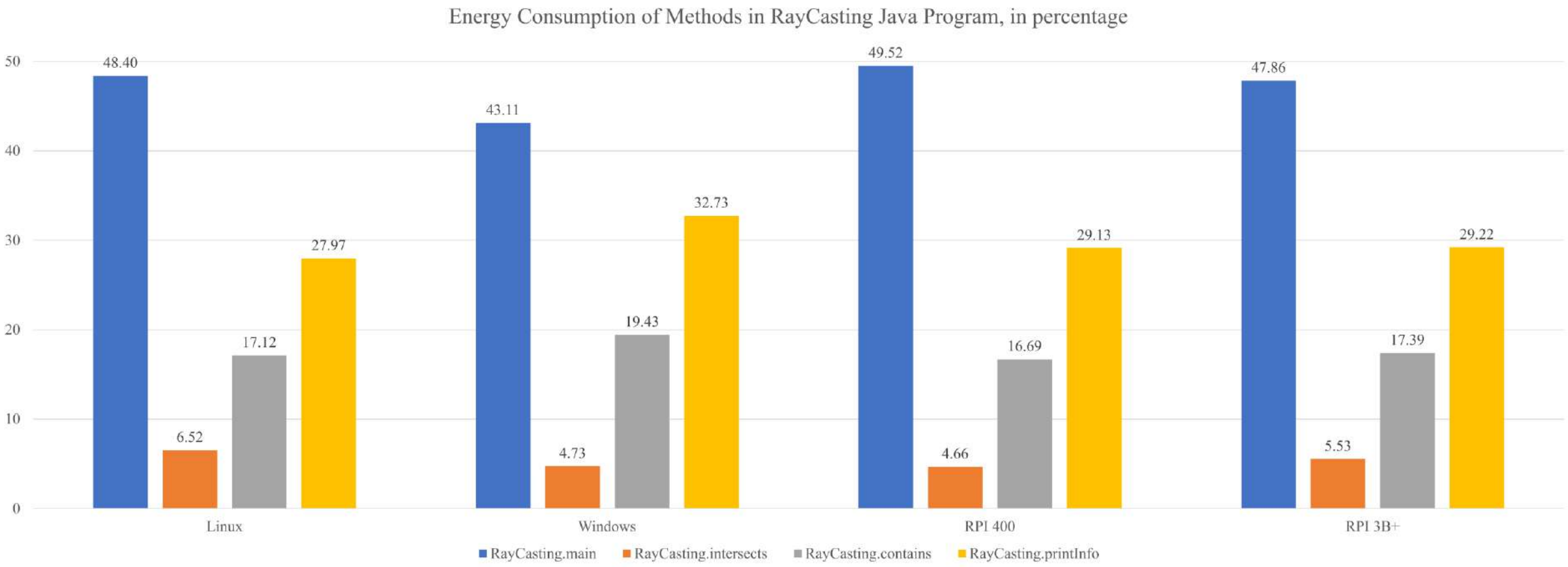
Joular Code Java

- Monitor power consumption and energy of each method and execution branch at runtime
- Works on all platforms, operating systems and architectures (same as Joular Core)
- Tracks the power consumption of every branch of the call tree through time

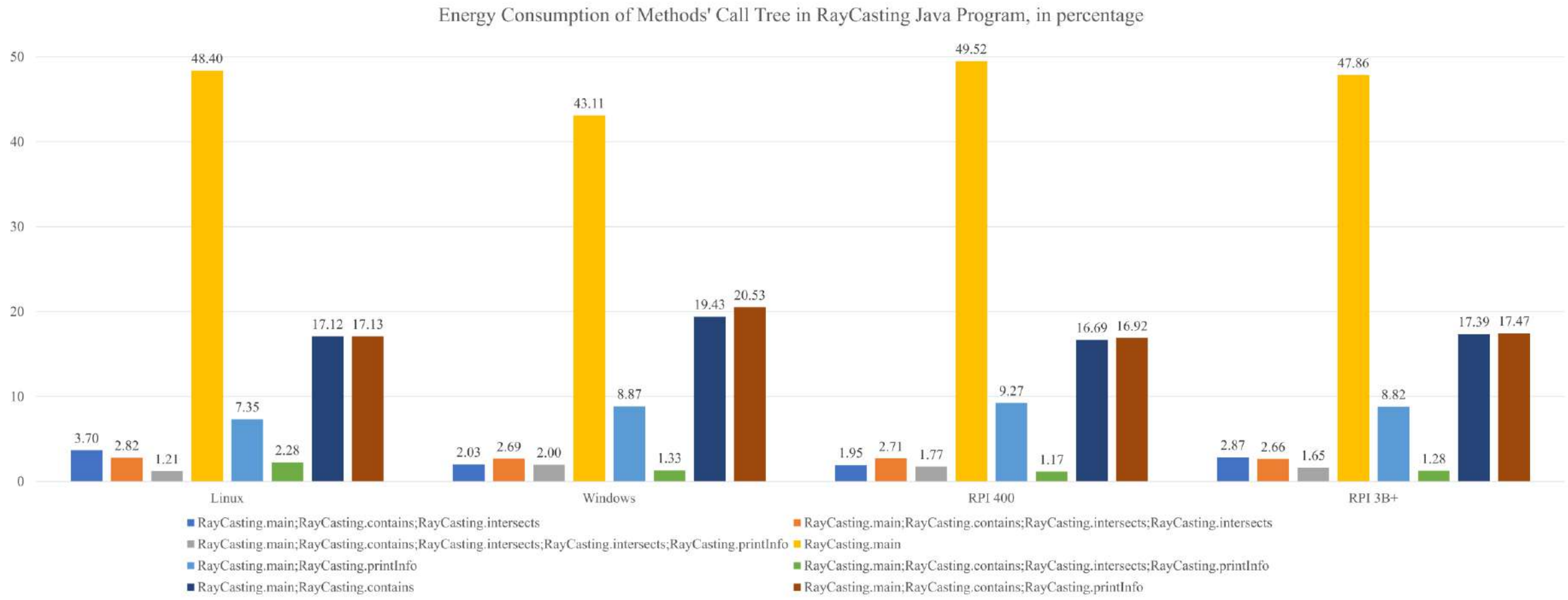
→ We can compute power per execution branch, method, class/package, …



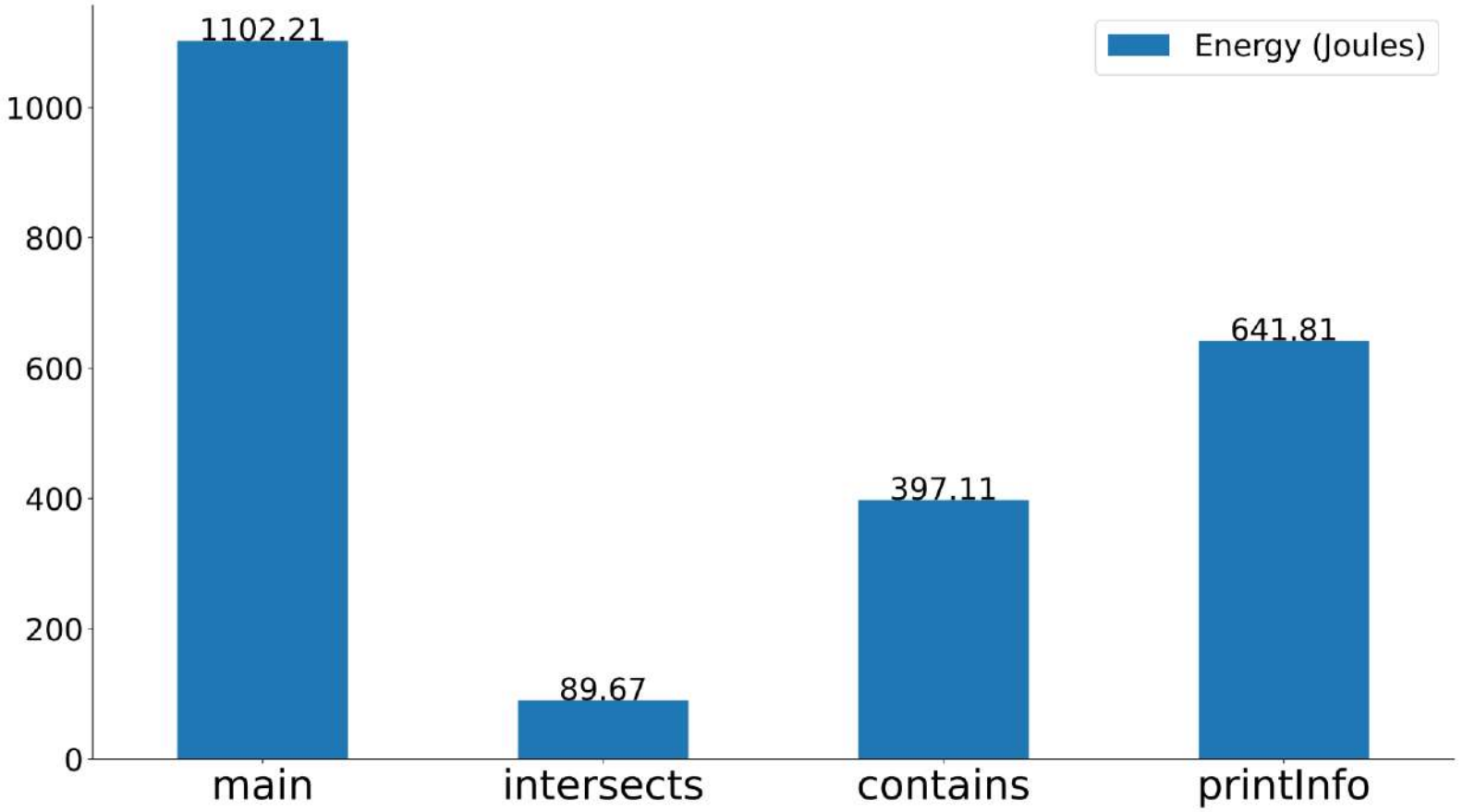
Example: Ray casting algorithm

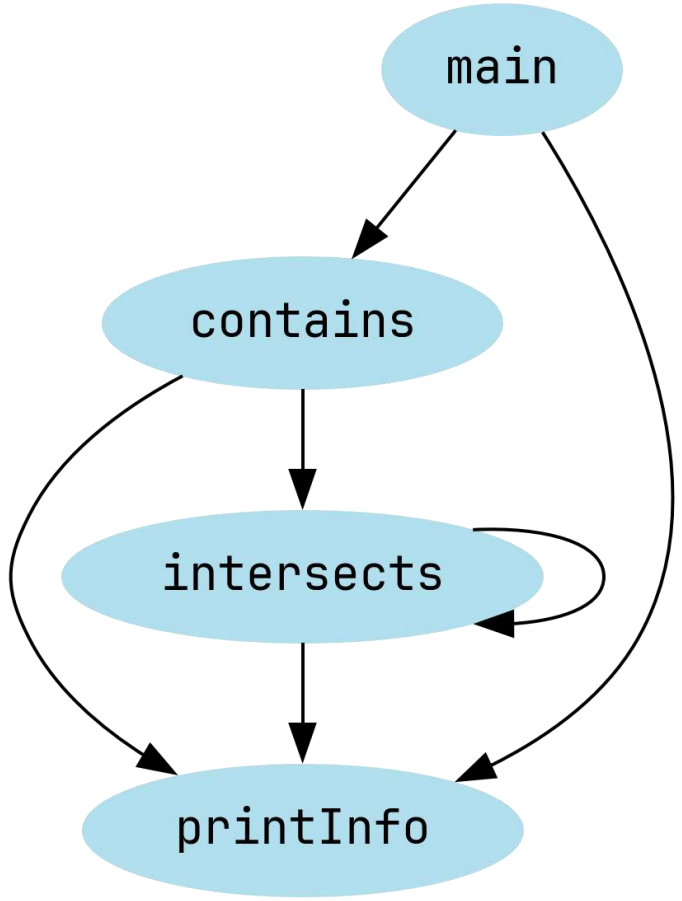


Example: Ray casting algorithm

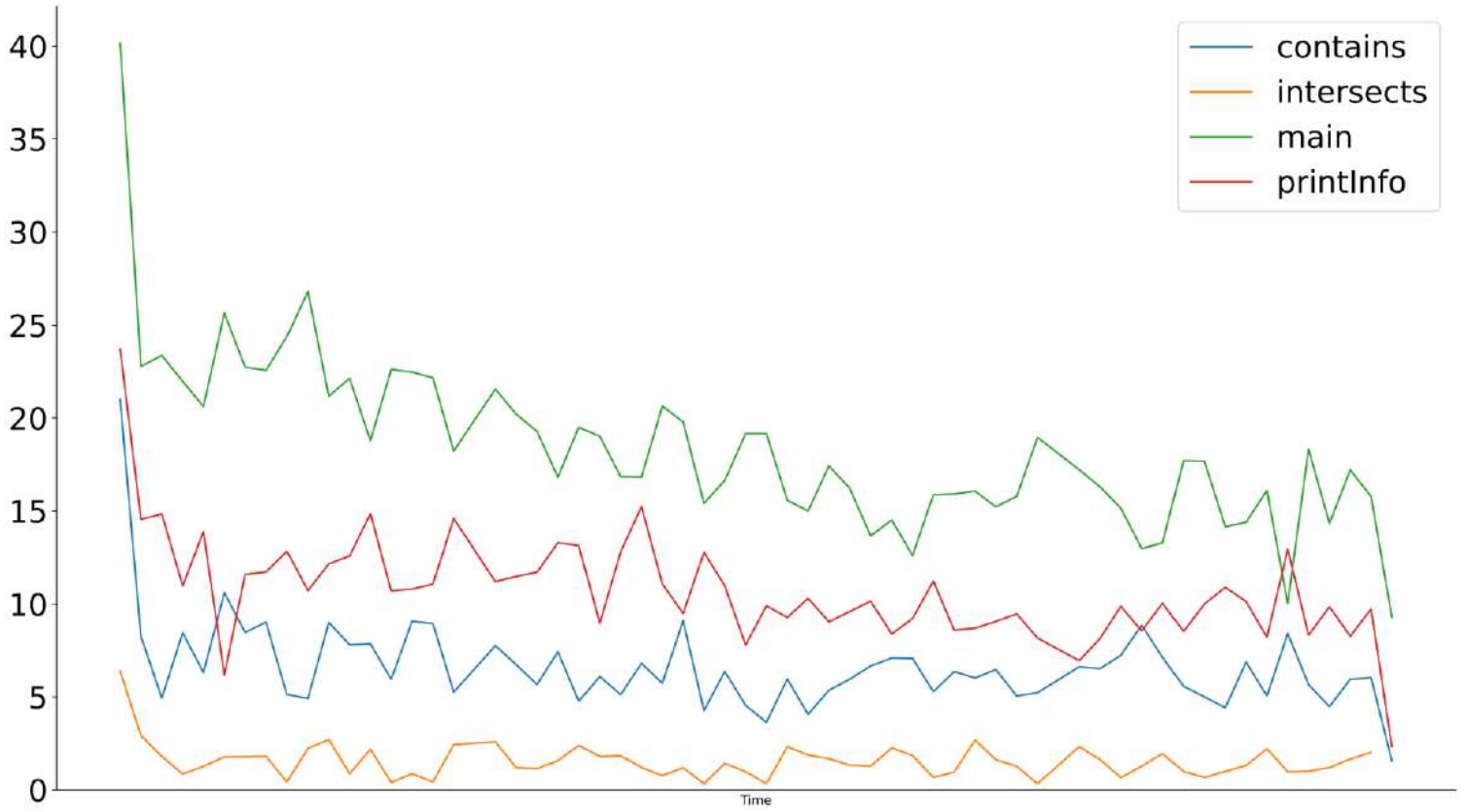


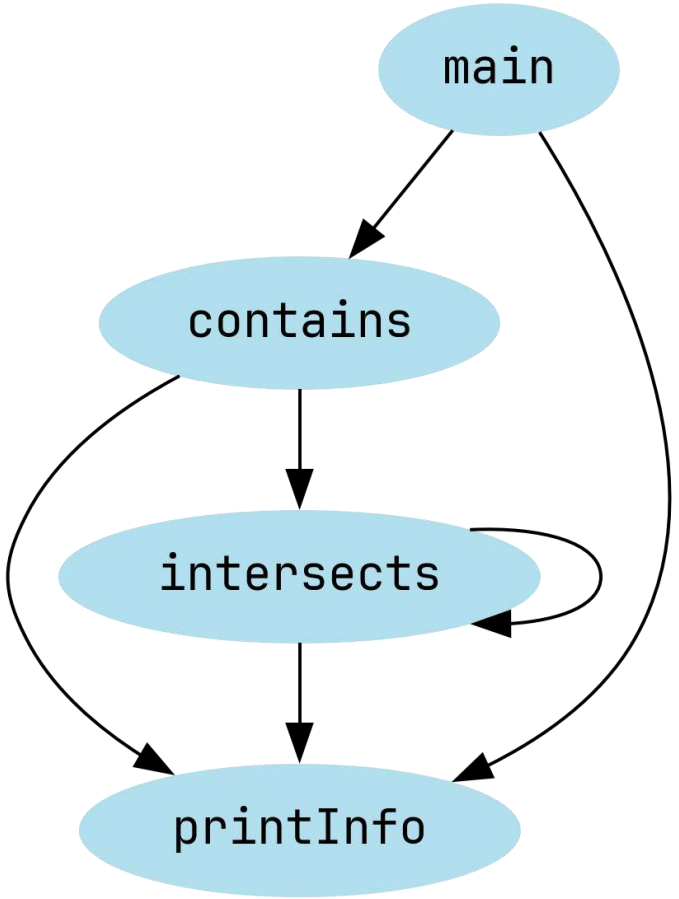
Example: Ray casting algorithm



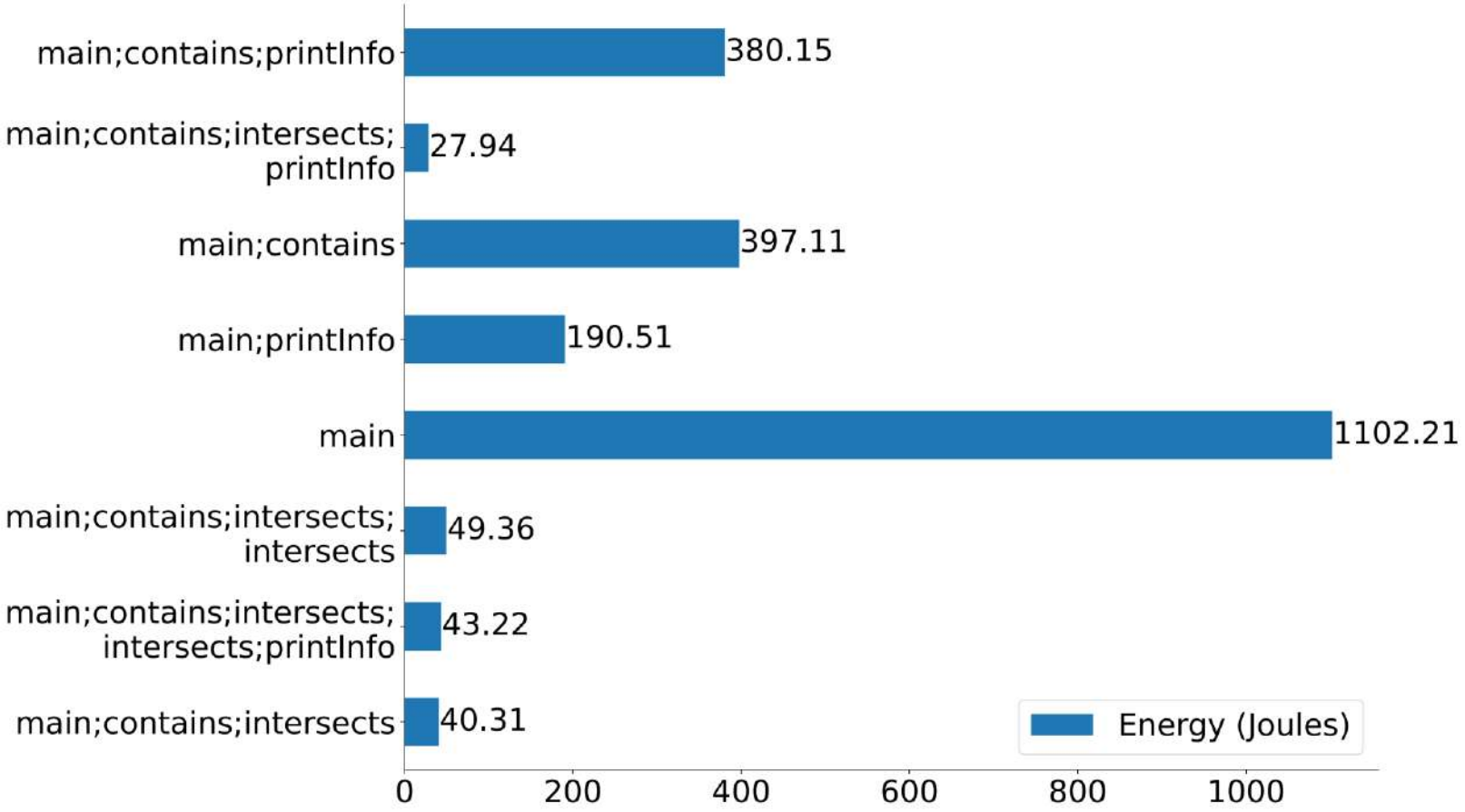


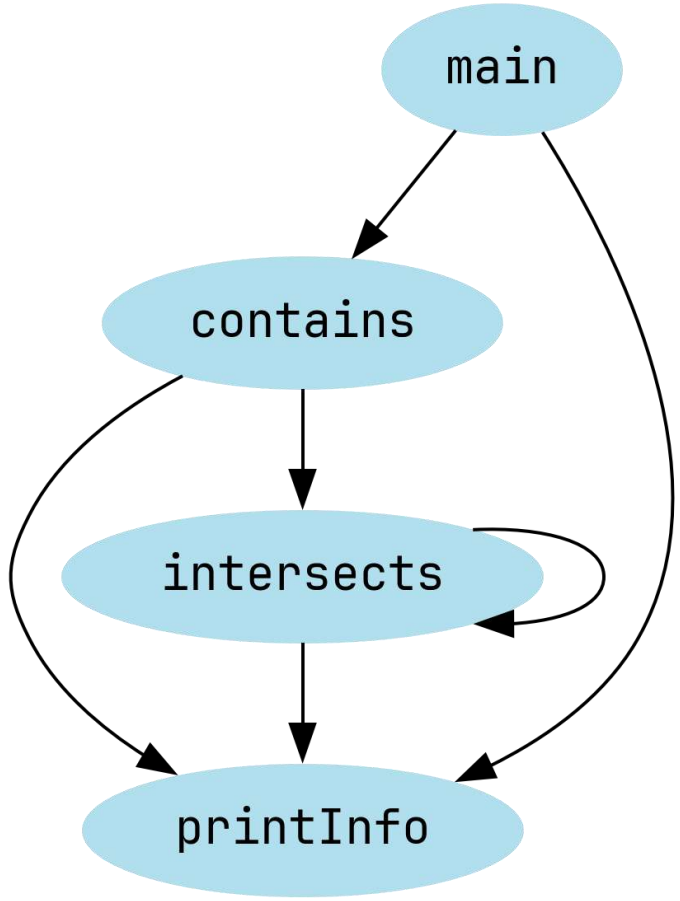
Example: Ray casting algorithm





Example: Ray casting algorithm





Joular Code for Other Languages

- First implementation for Java as we have the experience from Jalen and JoularJX

Joular Code for Other Languages

- First implementation for Java as we have the experience from Jalen and JoularJX

Next: optimize for JVM-based languages (Scala, Kotlin)

- Technically, Joular Code – Java works for these languages
- But many abstractions, methods, etc. → monitoring need to make sense for the developer/tester

Joular Code for Other Languages

- First implementation for Java as we have the experience from Jalen and JoularJX

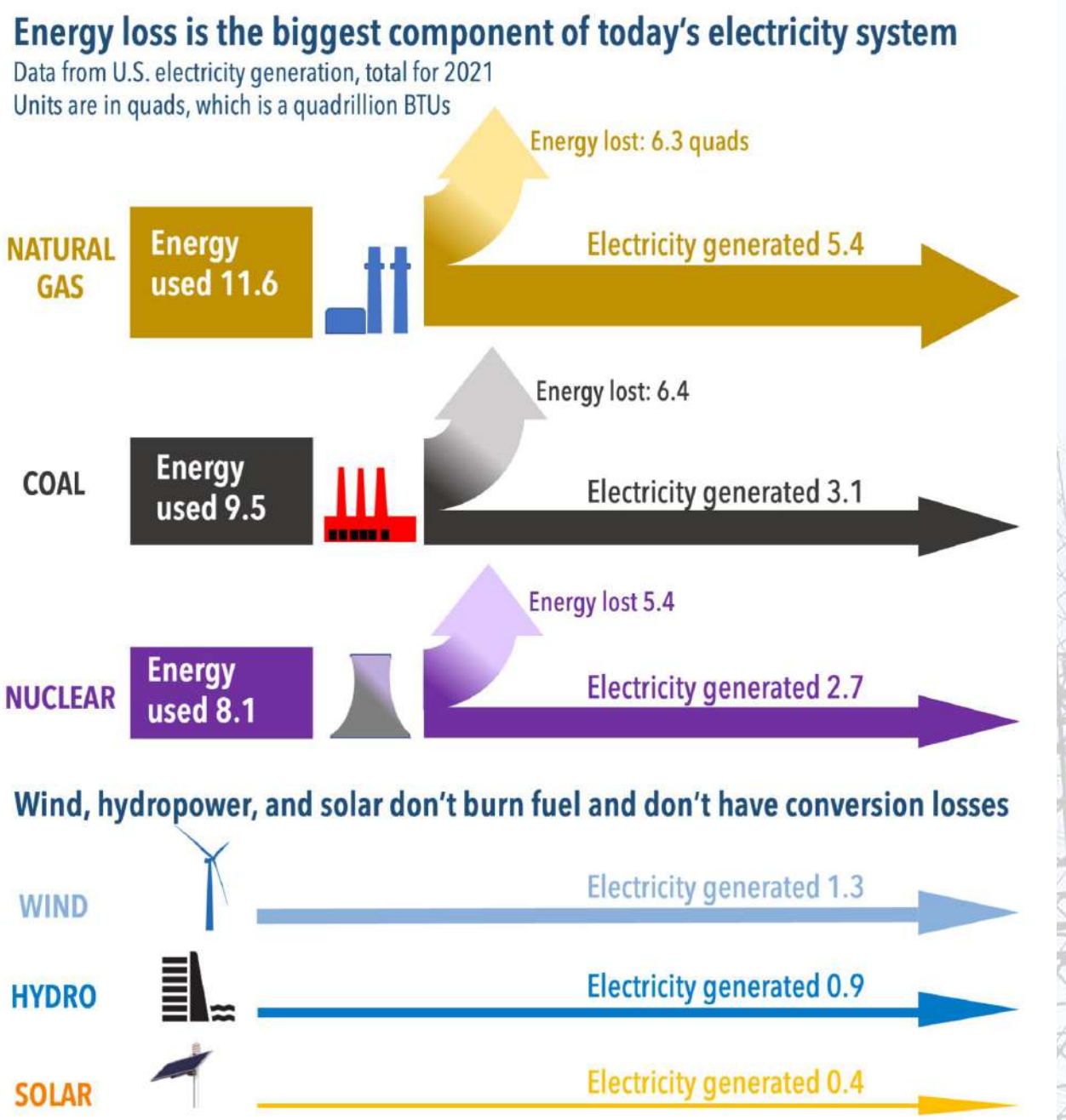
Next: optimize for JVM-based languages (Scala, Kotlin)

- Technically, Joular Code – Java works for these languages
- But many abstractions, methods, etc. → monitoring need to make sense for the developer/tester

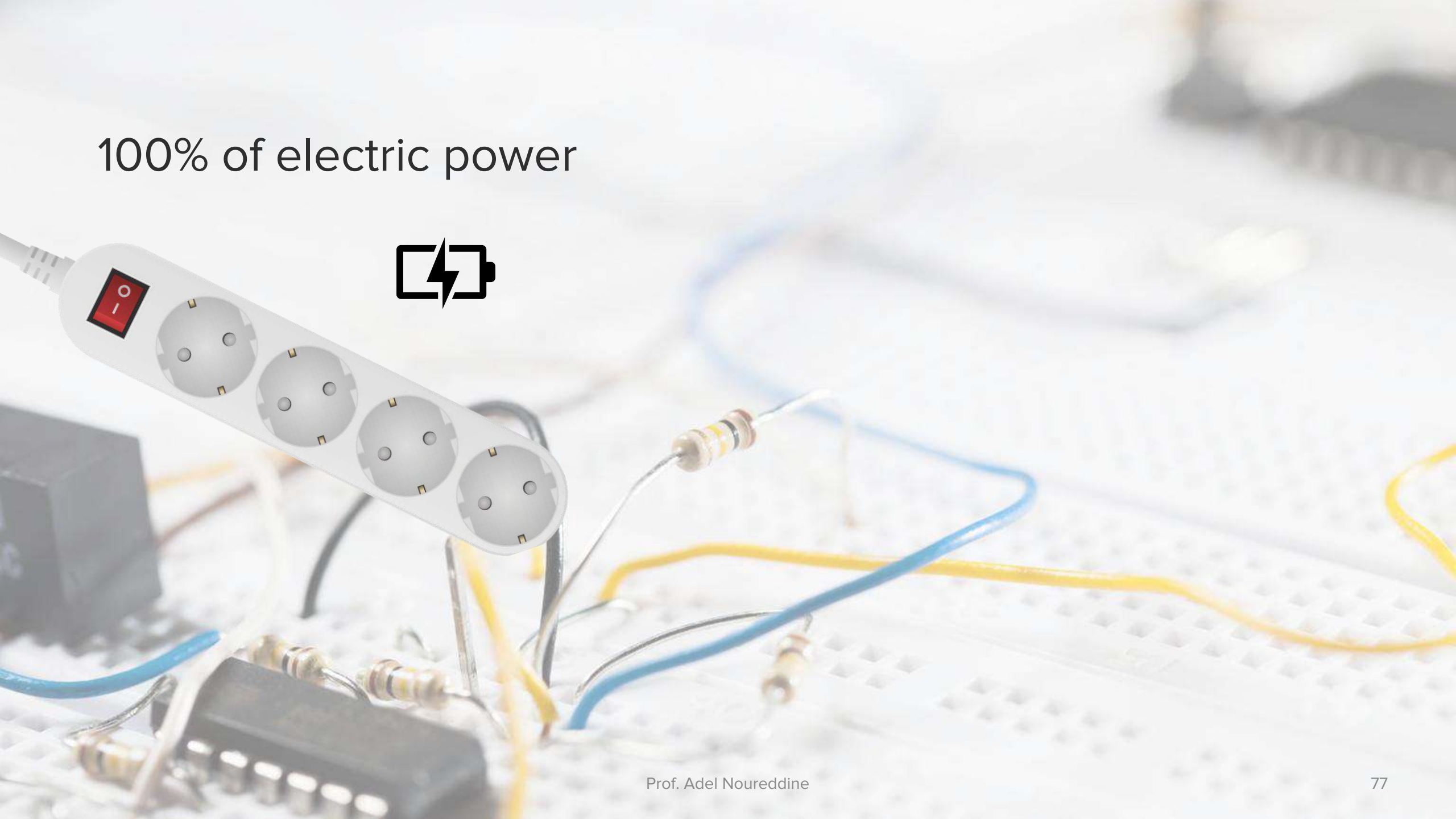
Roadmap:

- VM-based languages: .NET, SmallTalk
- Interpreter languages: Python, JavaScript
- Native languages: C, Rust, C++

Limits of Monitoring

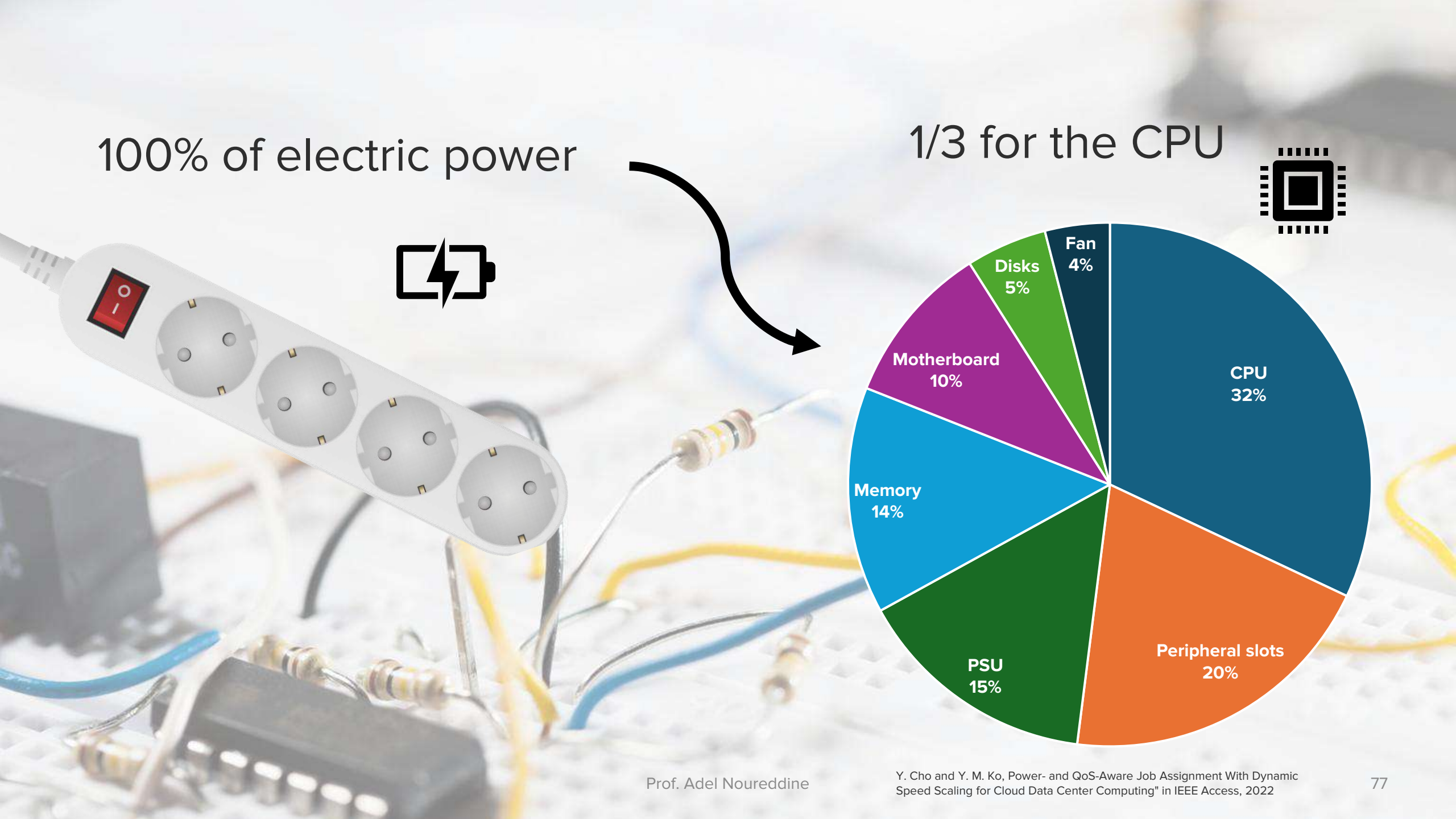

Energy loss is the biggest component of today's electricity system
Data from U.S. electricity generation, total for 2021
Units are in quads, which is a quadrillion BTUs
Energy lost: 6.3 quads
NATURAL GAS
Energy used 11.6
Electricity generated 5.4
Energy lost: 6.4
COAL
Energy used 9.5
Electricity generated 3.1
Energy lost 5.4
NUCLEAR
Energy used 8.1
Electricity generated 2.7
Wind, hydropower, and solar don't burn fuel and don't have conversion losses
WIND
Electricity generated 1.3
HYDRO
Electricity generated 0.9
SOLAR
Electricity generated 0.4
Does not include transmission losses, which are around 5%
Data from the Energy Information Administration
Image by Karin Kirk for Yale Climate Connections

100% of electric power



100% of electric power

1/3 for the CPU



Y. Cho and Y. M. Ko, Power- and QoS-Aware Job Assignment With Dynamic Speed Scaling for Cloud Data Center Computing" in IEEE Access, 2022

RAPL on Sandy Bridge

Modeling approach

~4% error
compared to plug power

RAPL since Haswell

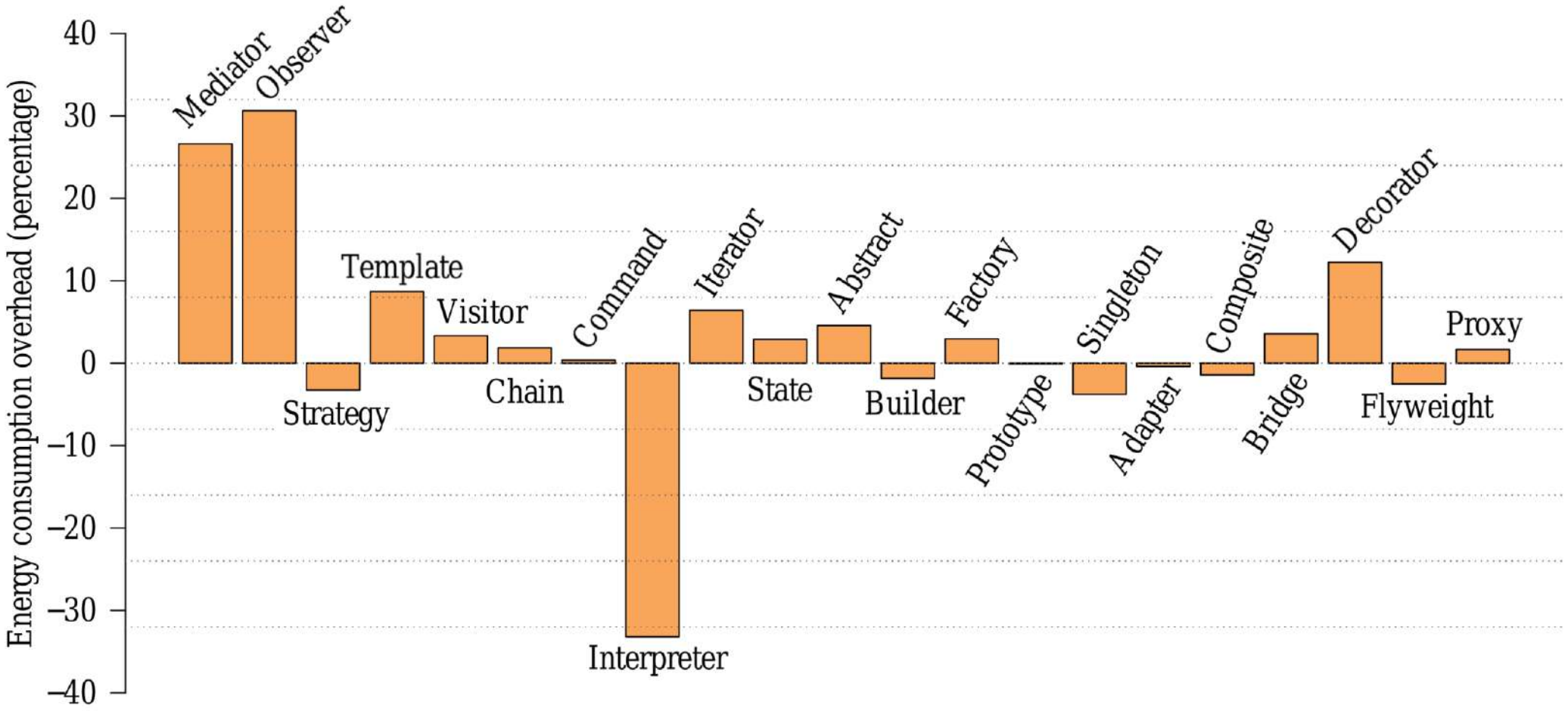
Integrated voltage regulators (FIVRs)

~1.8% error
compared to plug power

D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart and R. Geyer, "An Energy Efficiency Feature Survey of the Intel Haswell Processor," 2015 IEEE International Parallel and Distributed Processing Symposium Workshop, Hyderabad, India, 2015

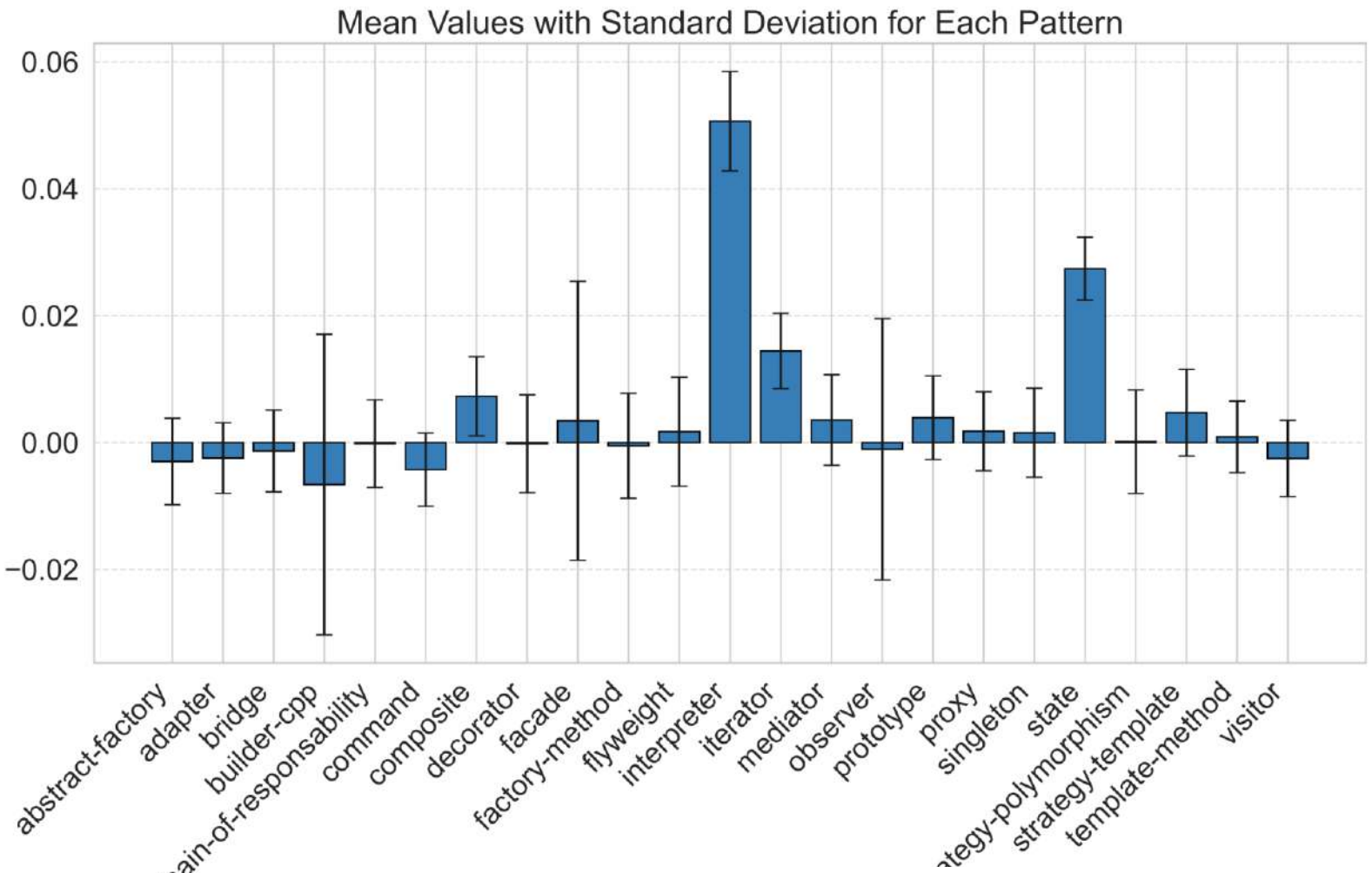
Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. 2018. RAPL in Action: Experiences in Using RAPL for Power Measurements. ACM Transactions on Modeling and Performance Evaluation of Computing Systems, June 2018

Energy consumption of design patterns



Optimising Energy Consumption of Design Patterns. Adel Noureddine, and Ajitha Rajan. In New Ideas and Emerging Results (NIER) of the 37th International Conference on Software Engineering (ICSE'15). Florence, Italy, May 2015.

Energy consumption of design patterns



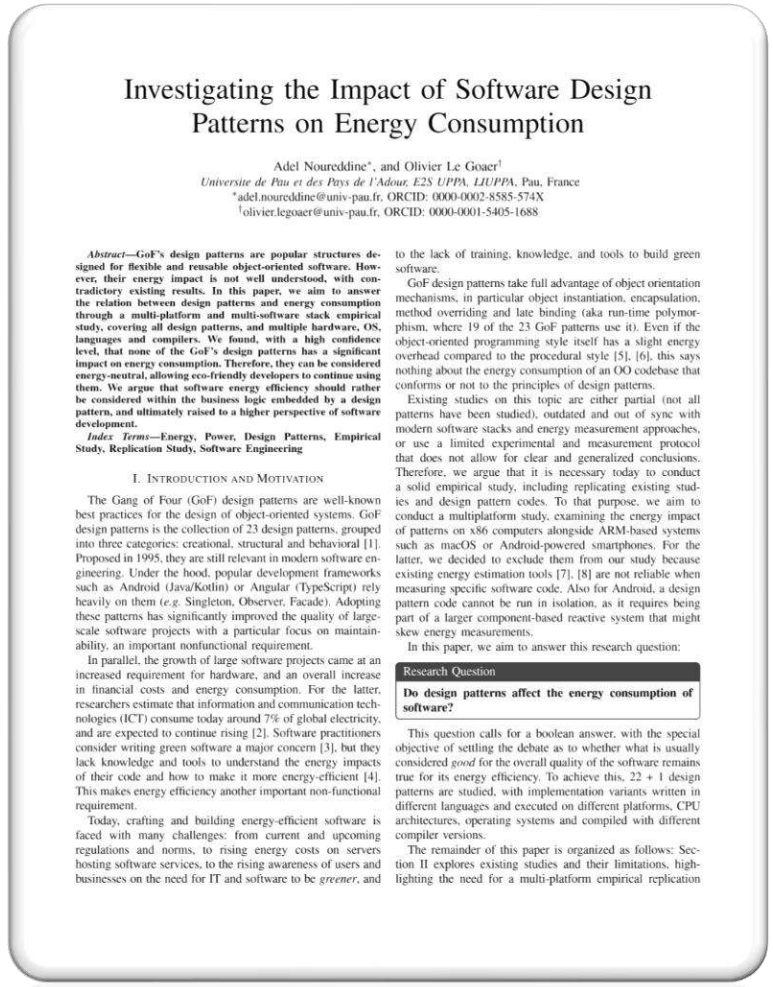
Energy-neutral across operating systems, programming languages, compilers, software stacks, and devices

Statistically representative & high confidence

Investigating the Impact of Software Design Patterns on Energy Consumption. Adel Noureddine, and Olivier Le Goaër. In the 22nd IEEE International Conference on Software Architecture (ICSA 2025). Odense, Denmark, April 2025.

Energy of design patterns

- *Gang of four* design patterns
- Comprehensive empirical study
- **Design patterns are energy neutral** across various platforms, OS and languages
- Influence of design patterns < 5% overall
- Negligible and imperceptible compared to the energy of the entire software
- Hardware architectures & software stacks have little impact on design patterns energy



Investigating the Impact of Software Design Patterns on Energy Consumption

Adel Noureddine*, and Olivier Le Goaer†
Universite de Pau et des Pays de l'Adour, E2S UPPA, LIUPPA, Pau, France
*adel.noureddine@univ-pau.fr, ORCID: 0000-0002-8585-574X
†olivier.legoaer@univ-pau.fr, ORCID: 0000-0001-5405-1688

***Abstract*—GoF's design patterns are popular structures designed for flexible and reusable object-oriented software. However, their energy impact is not well understood, with contradictory existing results. In this paper, we aim to answer the relation between design patterns and energy consumption through a multi-platform and multi-software stack empirical study, covering all design patterns, and multiple hardware, OS, languages and compilers. We found, with a high confidence level, that none of the GoF's design patterns has a significant impact on energy consumption. Therefore, they can be considered energy-neutral, allowing eco-friendly developers to continue using them. We argue that software energy efficiency should rather be considered within the business logic embedded by a design pattern, and ultimately raised to a higher perspective of software development.**

***Index Terms*—Energy, Power, Design Patterns, Empirical Study, Replication Study, Software Engineering**

I. Introduction and Motivation

The Gang of Four (GoF) design patterns are well-known best practices for the design of object-oriented systems. GoF design patterns is the collection of 23 design patterns, grouped into three categories: creational, structural and behavioral [1]. Proposed in 1995, they are still relevant in modern software engineering. Under the hood, popular development frameworks such as Android (Java/Kotlin) or Angular (TypeScript) rely heavily on them (*e.g.* Singleton, Observer, Facade). Adopting these patterns has significantly improved the quality of large-scale software projects with a particular focus on maintainability, an important nonfunctional requirement.

In parallel, the growth of large software projects came at an increased requirement for hardware, and an overall increase in financial costs and energy consumption. For the latter, researchers estimate that information and communication technologies (ICT) consume today around 7% of global electricity, and are expected to continue rising [2]. Software practitioners consider writing green software a major concern [3], but they lack knowledge and tools to understand the energy impacts of their code and how to make it more energy-efficient [4]. This makes energy efficiency another important non-functional requirement.

Today, crafting and building energy-efficient software is faced with many challenges: from current and upcoming regulations and norms, to rising energy costs on servers hosting software services, to the rising awareness of users and businesses on the need for IT and software to be *greener*, and to the lack of training, knowledge, and tools to build green software.

GoF design patterns take full advantage of object orientation mechanisms, in particular object instantiation, encapsulation, method overriding and late binding (aka run-time polymorphism, where 19 of the 23 GoF patterns use it). Even if the object-oriented programming style itself has a slight energy overhead compared to the procedural style [5], [6], this says nothing about the energy consumption of an OO codebase that conforms or not to the principles of design patterns.

Existing studies on this topic are either partial (not all patterns have been studied), outdated and out of sync with modern software stacks and energy measurement approaches, or use a limited experimental and measurement protocol that does not allow for clear and generalized conclusions. Therefore, we argue that it is necessary today to conduct a solid empirical study, including replicating existing studies and design pattern codes. To that purpose, we aim to conduct a multiplatform study, examining the energy impact of patterns on x86 computers alongside ARM-based systems such as macOS or Android-powered smartphones. For the latter, we decided to exclude them from our study because existing energy estimation tools [7], [8] are not reliable when measuring specific software code. Also for Android, a design pattern code cannot be run in isolation, as it requires being part of a larger component-based reactive system that might skew energy measurements.

In this paper, we aim to answer this research question:

Research Question

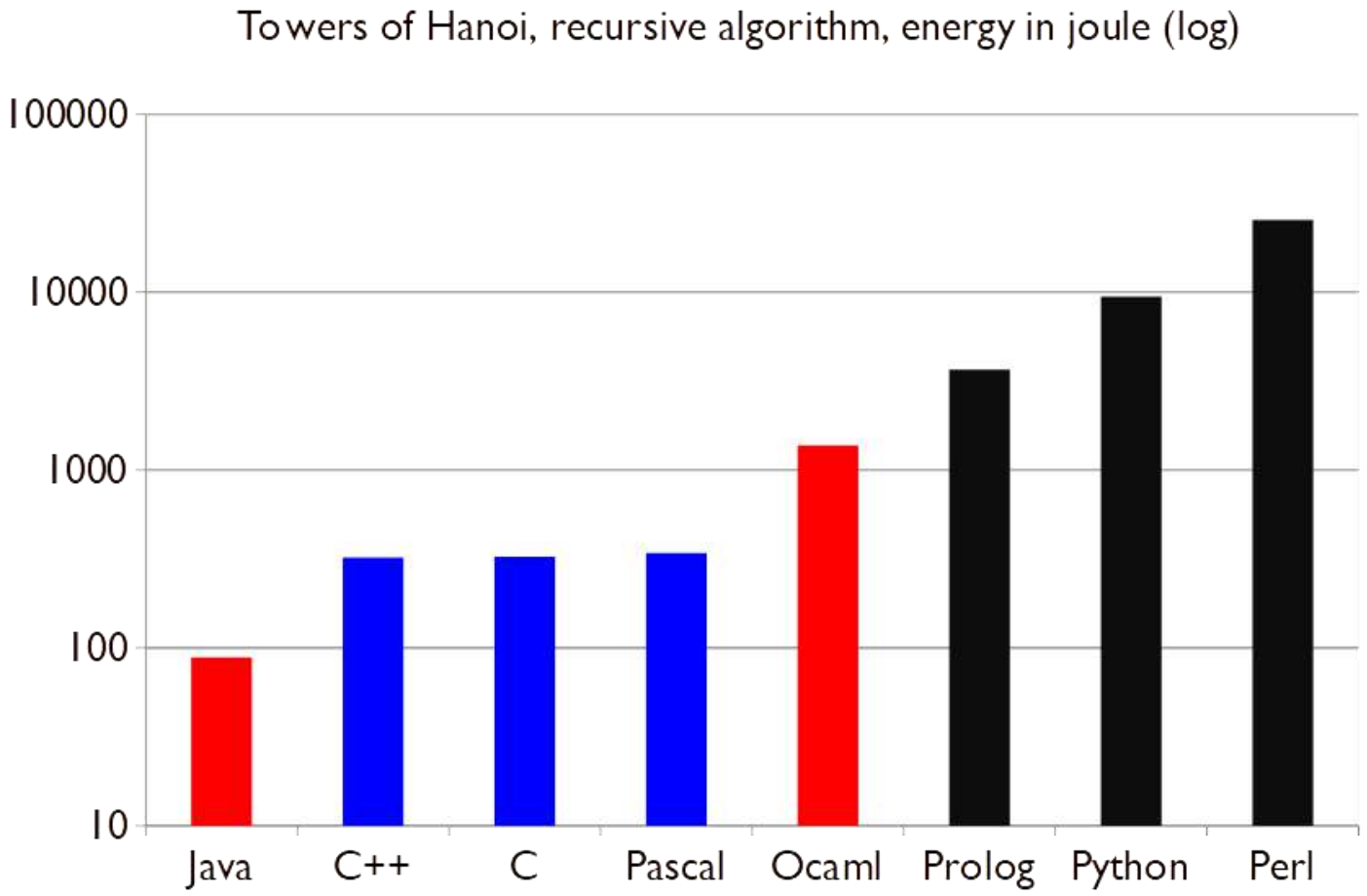
Do design patterns affect the energy consumption of software?

This question calls for a boolean answer, with the special objective of settling the debate as to whether what is usually considered *good* for the overall quality of the software remains true for its energy efficiency. To achieve this, 22 + 1 design patterns are studied, with implementation variants written in different languages and executed on different platforms, CPU architectures, operating systems and compiled with different compiler versions.

The remainder of this paper is organized as follows: Section II explores existing studies and their limitations, highlighting the need for a multi-platform empirical replication

Investigating the Impact of Software Design Patterns on Energy Consumption. Adel Noureddine, and Olivier Le Goaër. In the 22nd IEEE International Conference on Software Architecture (ICSA 2025). Odense, Denmark, April 2025.

Energy of programming languages



A Preliminary Study of the Impact of Software Engineering on GreenIT

Adel Noureddine[1,2], Aurelien Bourdon[1,2], Romain Rouvoy[1,2], Lionel Seinturier[1,2,3]
[1] *Inria Lille – Nord Europe*
[2] *University Lille 1 - LIFL CNRS UMR 8022, France*
[3] *Institut Universitaire de France*
firstname.lastname@inria.fr

***Abstract*—GreenIT has emerged as a discipline concerned with the optimization of software solutions with regards to their energy consumption. In this domain, most of state-of-the-art solutions offer limited or constraining approaches to monitor the energy consumption of a device or a process. In this paper, we therefore report on a runtime energy monitoring framework we developed to easily report on the energy consumption of system processes. Concretely, our approach adopts an OS-level library, called POWERAPI, which estimates the power consumption of processes according to different dimensions (CPU, network, etc.). In order to better understand potential energy leaks of legacy software, we use this library to study the impact of programming languages and algorithmic choices on the energy consumption. This preliminary study is based on an empirical evaluation of a eight implementations of the Towers of Hanoi problem.**

***Keywords*-Performance; Measurement; Experimentation; Energy; Power Model; Monitoring**

I. INTRODUCTION

Energy-aware software solutions and approaches are becoming broadly available, as energy concerns are becoming mainstream. The increasing usage of computers and other electronic devices (*e.g.*, smartphones, sensors) is continuously impacting our overall energy consumption. *Information and Communications Technology* (ICT) accounted for 2% of global carbon emissions in 2007 [1] or 830 $MtCO_2e$ [2], and is expected to grow to 1,430 $MtCO_2e$ in 2020 [2]. These values illustrate the opportunities for efficient ICT solutions to reduce carbon emissions and energy consumption.

Rising energy costs in computers and mobile devices implies the optimization and the adaptation of computer systems. In this domain, research in GreenIT already proposes various approaches aiming at achieving energy savings in computers and software. In particular, monitoring the energy consumption of the system is a requirement to achieve such savings. However, most of the state-of-the-art approaches either focus on the hardware [3], or offer coarse-grained energy feedbacks of software [4], [5].

In this paper, we therefore propose to gather applications energy feedback information at runtime and with similar accuracy as hardware equipments while using only a software approach. Our approach consists of a system monitoring library (at the operating system level), called POWERAPI. POWERAPI estimates the energy consumption of running processes, in real-time, based on raw information collected from hardware devices (*e.g.*, CPU, network card) through the operating system. We use both state-of-the-art energy models and propose new models to compute the energy consumption of software.

Using this monitoring framework, we compare the energy footprint of eight implementations of the Towers of Hanoi program. Our preliminary results demonstrate that we can clearly observe the impact of different implementations of algorithms on the energy consumption, thus providing the opportunity at a later stage to reduce their energy footprint.

The remainder of this paper is organized as follows. In Section II, we describe our motivations and the main challenges we tackle. Section III describes our approach, the design of our proposed architecture and our energy models. Section IV details the implementation and validation of our prototype. In Section V, we report and discuss the preliminary results we obtained from comparing the Towers of Hanoi implementations. Related work is discussed in Section VI, while we conclude in Section VII.

II. MOTIVATION AND CHALLENGES

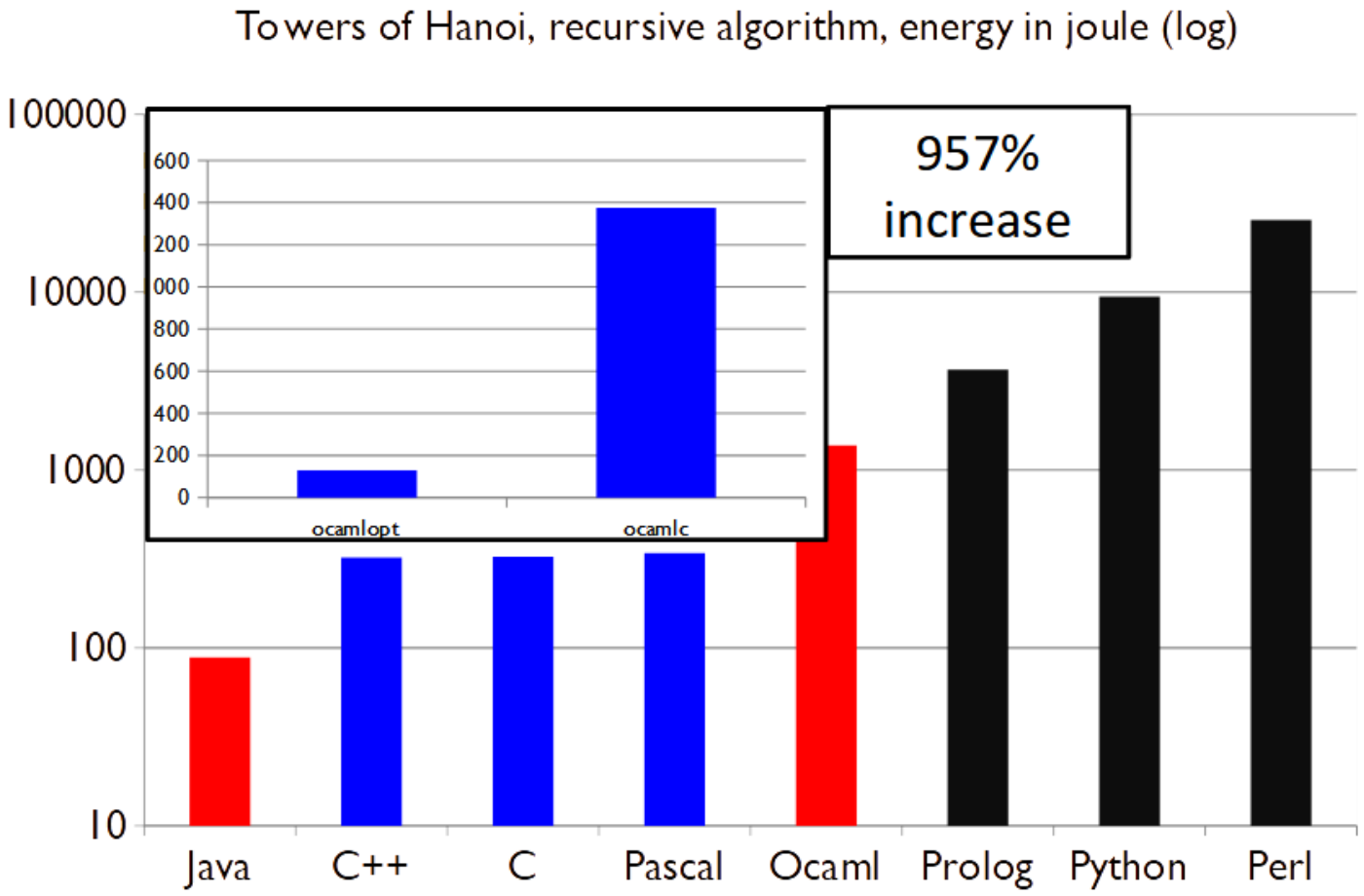
According to [6], ICT consumes up to 7% of global power consumption (or 168 GW) in 2008. This number is predicted to grow and double to 433 GW in 2020 or more than 14.5% of worldwide power consumption [6]. With the rise of ICT power consumption, power-aware optimizations and management cannot be done efficiently without accurate power measurements. In particular, we argue that monitoring the power consumption of applications (and not only hardware) has become a requirement for power-aware *Software as a Service* (SaaS) and power-aware software adaptation.

Hardware monitoring is usually achieved through additional hardware measurement equipments, such as multimeters or specialized integrated circuits (cf. Section VI). This approach offers a precise and accurate measurement of the energy consumption of hardware components, but at the cost of an additional investment. Furthermore, it cannot monitor the energy consumption of software components and running applications.

We rather believe that a scalable approach can be achieved by a software-centric approach. Monitoring the energy con-

A Preliminary Study of the Impact of Software Engineering on GreenIT. Adel Noureddine, Aurélien Bourdon, Romain Rouvoy, and Lionel Seinturier. In the First International Workshop on Green and Sustainable Software (GREENS'12)/ICSE'12,), Zurich, Switzerland, June 2012.

Energy of programming languages



A Preliminary Study of the Impact of Software Engineering on GreenIT

Adel Noureddine[1,2], Aurelien Bourdon[1,2], Romain Rouvoy[1,2], Lionel Seinturier[1,2,3]
[1] *Inria Lille – Nord Europe*
[2] *University Lille 1 - LIFL CNRS UMR 8022, France*
[3] *Institut Universitaire de France*
firstname.lastname@inria.fr

***Abstract*—GreenIT has emerged as a discipline concerned with the optimization of software solutions with regards to their energy consumption. In this domain, most of state-of-the-art solutions offer limited or constraining approaches to monitor the energy consumption of a device or a process. In this paper, we therefore report on a runtime energy monitoring framework we developed to easily report on the energy consumption of system processes. Concretely, our approach adopts an OS-level library, called PowerAPI, which estimates the power consumption of processes according to different dimensions (CPU, network, etc.). In order to better understand potential energy leaks of legacy software, we use this library to study the impact of programming languages and algorithmic choices on the energy consumption. This preliminary study is based on an empirical evaluation of a eight implementations of the Towers of Hanoi problem.**

***Keywords*-Performance; Measurement; Experimentation; Energy; Power Model; Monitoring**

I. Introduction

Energy-aware software solutions and approaches are becoming broadly available, as energy concerns are becoming mainstream. The increasing usage of computers and other electronic devices (*e.g.*, smartphones, sensors) is continuously impacting our overall energy consumption. *Information and Communications Technology* (ICT) accounted for 2% of global carbon emissions in 2007 [1] or 830 $MtCO_2e$ [2], and is expected to grow to 1,430 $MtCO_2e$ in 2020 [2]. These values illustrate the opportunities for efficient ICT solutions to reduce carbon emissions and energy consumption.

Rising energy costs in computers and mobile devices implies the optimization and the adaptation of computer systems. In this domain, research in GreenIT already proposes various approaches aiming at achieving energy savings in computers and software. In particular, monitoring the energy consumption of the system is a requirement to achieve such savings. However, most of the state-of-the-art approaches either focus on the hardware [3], or offer coarse-grained energy feedbacks of software [4], [5].

In this paper, we therefore propose to gather applications energy feedback information at runtime and with similar accuracy as hardware equipments while using only a software approach. Our approach consists of a system monitoring library (at the operating system level), called PowerAPI. PowerAPI estimates the energy consumption of running processes, in real-time, based on raw information collected from hardware devices (*e.g.*, CPU, network card) through the operating system. We use both state-of-the-art energy models and propose new models to compute the energy consumption of software.

Using this monitoring framework, we compare the energy footprint of eight implementations of the Towers of Hanoi program. Our preliminary results demonstrate that we can clearly observe the impact of different implementations of algorithms on the energy consumption, thus providing the opportunity at a later stage to reduce their energy footprint.

The remainder of this paper is organized as follows. In Section II, we describe our motivations and the main challenges we tackle. Section III describes our approach, the design of our proposed architecture and our energy models. Section IV details the implementation and validation of our prototype. In Section V, we report and discuss the preliminary results we obtained from comparing the Towers of Hanoi implementations. Related work is discussed in Section VI, while we conclude in Section VII.

II. Motivation and Challenges

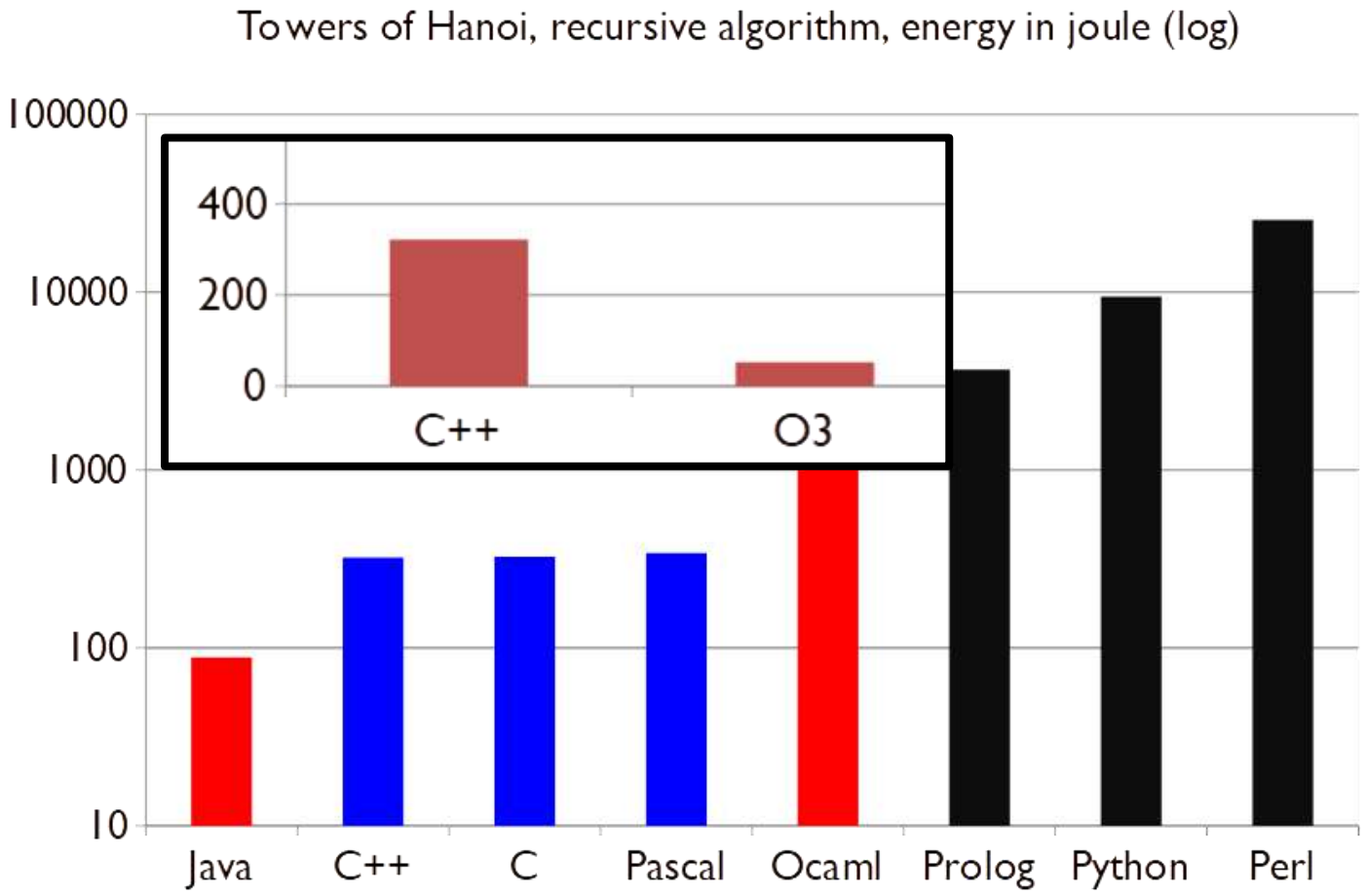
According to [6], ICT consumes up to 7% of global power consumption (or 168 GW) in 2008. This number is predicted to grow and double to 433 GW in 2020 or more than 14.5% of worldwide power consumption [6]. With the rise of ICT power consumption, power-aware optimizations and management cannot be done efficiently without accurate power measurements. In particular, we argue that monitoring the power consumption of applications (and not only hardware) has become a requirement for power-aware *Software as a Service* (SaaS) and power-aware software adaptation.

Hardware monitoring is usually achieved through additional hardware measurement equipments, such as multimeters or specialized integrated circuits (cf. Section VI). This approach offers a precise and accurate measurement of the energy consumption of hardware components, but at the cost of an additional investment. Furthermore, it cannot monitor the energy consumption of software components and running applications.

We rather believe that a scalable approach can be achieved by a software-centric approach. Monitoring the energy con-

A Preliminary Study of the Impact of Software Engineering on GreenIT. Adel Noureddine, Aurélien Bourdon, Romain Rouvoy, and Lionel Seinturier. In the First International Workshop on Green and Sustainable Software (GREENS'12)/ICSE'12,), Zurich, Switzerland, June 2012.

Energy of programming languages



A Preliminary Study of the Impact of Software Engineering on GreenIT

Adel Noureddine[1,2], Aurelien Bourdon[1,2], Romain Rouvoy[1,2], Lionel Seinturier[1,2,3]
[1] *Inria Lille – Nord Europe*
[2] *University Lille 1 - LIFL CNRS UMR 8022, France*
[3] *Institut Universitaire de France*
firstname.lastname@inria.fr

***Abstract*—GreenIT has emerged as a discipline concerned with the optimization of software solutions with regards to their energy consumption. In this domain, most of state-of-the-art solutions offer limited or constraining approaches to monitor the energy consumption of a device or a process. In this paper, we therefore report on a runtime energy monitoring framework we developed to easily report on the energy consumption of system processes. Concretely, our approach adopts an OS-level library, called PowerAPI, which estimates the power consumption of processes according to different dimensions (CPU, network, etc.). In order to better understand potential energy leaks of legacy software, we use this library to study the impact of programming languages and algorithmic choices on the energy consumption. This preliminary study is based on an empirical evaluation of a eight implementations of the Towers of Hanoi problem.**

***Keywords*-Performance; Measurement; Experimentation; Energy; Power Model; Monitoring**

I. Introduction

Energy-aware software solutions and approaches are becoming broadly available, as energy concerns are becoming mainstream. The increasing usage of computers and other electronic devices (*e.g.*, smartphones, sensors) is continuously impacting our overall energy consumption. *Information and Communications Technology* (ICT) accounted for 2% of global carbon emissions in 2007 [1] or 830 $MtCO_2e$ [2], and is expected to grow to 1,430 $MtCO_2e$ in 2020 [2]. These values illustrate the opportunities for efficient ICT solutions to reduce carbon emissions and energy consumption.

Rising energy costs in computers and mobile devices implies the optimization and the adaptation of computer systems. In this domain, research in GreenIT already proposes various approaches aiming at achieving energy savings in computers and software. In particular, monitoring the energy consumption of the system is a requirement to achieve such savings. However, most of the state-of-the-art approaches either focus on the hardware [3], or offer coarse-grained energy feedbacks of software [4], [5].

In this paper, we therefore propose to gather applications energy feedback information at runtime and with similar accuracy as hardware equipments while using only a software approach. Our approach consists of a system monitoring library (at the operating system level), called PowerAPI. PowerAPI estimates the energy consumption of running processes, in real-time, based on raw information collected from hardware devices (*e.g.*, CPU, network card) through the operating system. We use both state-of-the-art energy models and propose new models to compute the energy consumption of software.

Using this monitoring framework, we compare the energy footprint of eight implementations of the Towers of Hanoi program. Our preliminary results demonstrate that we can clearly observe the impact of different implementations of algorithms on the energy consumption, thus providing the opportunity at a later stage to reduce their energy footprint.

The remainder of this paper is organized as follows. In Section II, we describe our motivations and the main challenges we tackle. Section III describes our approach, the design of our proposed architecture and our energy models. Section IV details the implementation and validation of our prototype. In Section V, we report and discuss the preliminary results we obtained from comparing the Towers of Hanoi implementations. Related work is discussed in Section VI, while we conclude in Section VII.

II. Motivation and Challenges

According to [6], ICT consumes up to 7% of global power consumption (or 168 GW) in 2008. This number is predicted to grow and double to 433 GW in 2020 or more than 14.5% of worldwide power consumption [6]. With the rise of ICT power consumption, power-aware optimizations and management cannot be done efficiently without accurate power measurements. In particular, we argue that monitoring the power consumption of applications (and not only hardware) has become a requirement for power-aware *Software as a Service* (SaaS) and power-aware software adaptation.

Hardware monitoring is usually achieved through additional hardware measurement equipments, such as multimeters or specialized integrated circuits (cf. Section VI). This approach offers a precise and accurate measurement of the energy consumption of hardware components, but at the cost of an additional investment. Furthermore, it cannot monitor the energy consumption of software components and running applications.

We rather believe that a scalable approach can be achieved by a software-centric approach. Monitoring the energy con-

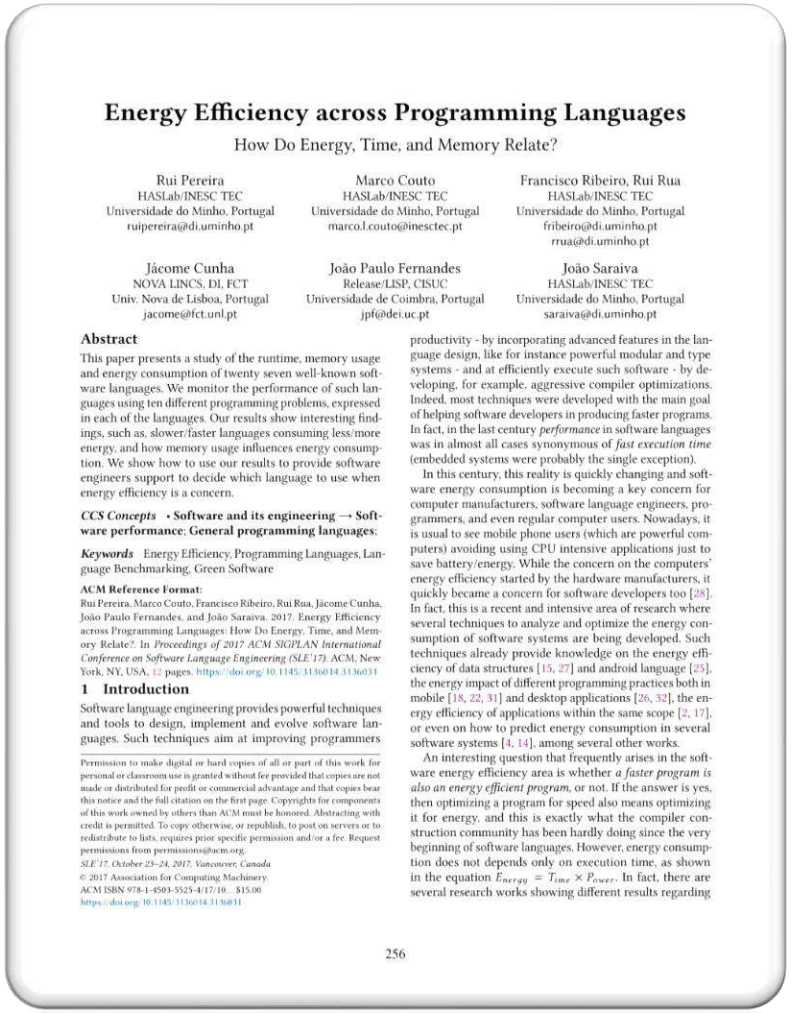
A Preliminary Study of the Impact of Software Engineering on GreenIT. Adel Noureddine, Aurélien Bourdon, Romain Rouvoy, and Lionel Seinturier. In the First International Workshop on Green and Sustainable Software (GREENS'12)/ICSE'12,), Zurich, Switzerland, June 2012.

Energy of programming languages

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84



Energy Efficiency across Programming Languages

How Do Energy, Time, and Memory Relate?

Rui Pereira
HASLab/INESC TEC
Universidade do Minho, Portugal
ruipereira@di.uminho.pt

Marco Couto
HASLab/INESC TEC
Universidade do Minho, Portugal
marco.l.couto@inesctec.pt

Francisco Ribeiro, Rui Rua
HASLab/INESC TEC
Universidade do Minho, Portugal
fribeiro@di.uminho.pt
rrua@di.uminho.pt

Jácome Cunha
NOVA LINCS, DI, FCT
Univ. Nova de Lisboa, Portugal
jacome@fct.unl.pt

João Paulo Fernandes
Release/LISP, CISUC
Universidade de Coimbra, Portugal
jpf@dei.uc.pt

João Saraiva
HASLab/INESC TEC
Universidade do Minho, Portugal
saraiva@di.uminho.pt

Abstract

This paper presents a study of the runtime, memory usage and energy consumption of twenty seven well-known software languages. We monitor the performance of such languages using ten different programming problems, expressed in each of the languages. Our results show interesting findings, such as, slower/faster languages consuming less/more energy, and how memory usage influences energy consumption. We show how to use our results to provide software engineers support to decide which language to use when energy efficiency is a concern.

CCS Concepts • **Software and its engineering → Software performance; General programming languages;**

Keywords Energy Efficiency, Programming Languages, Language Benchmarking, Green Software

ACM Reference Format:
Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. 2017. Energy Efficiency across Programming Languages: How Do Energy, Time, and Memory Relate?. In *Proceedings of 2017 ACM SIGPLAN International Conference on Software Language Engineering (SLE'17).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3136014.3136031

1 Introduction

Software language engineering provides powerful techniques and tools to design, implement and evolve software languages. Such techniques aim at improving programmers productivity - by incorporating advanced features in the language design, like for instance powerful modular and type systems - and at efficiently execute such software - by developing, for example, aggressive compiler optimizations. Indeed, most techniques were developed with the main goal of helping software developers in producing faster programs. In fact, in the last century *performance* in software languages was in almost all cases synonymous of *fast execution time* (embedded systems were probably the single exception).

In this century, this reality is quickly changing and software energy consumption is becoming a key concern for computer manufacturers, software language engineers, programmers, and even regular computer users. Nowadays, it is usual to see mobile phone users (which are powerful computers) avoiding using CPU intensive applications just to save battery/energy. While the concern on the computers' energy efficiency started by the hardware manufacturers, it quickly became a concern for software developers too [28]. In fact, this is a recent and intensive area of research where several techniques to analyze and optimize the energy consumption of software systems are being developed. Such techniques already provide knowledge on the energy efficiency of data structures [15, 27] and android language [25], the energy impact of different programming practices both in mobile [18, 22, 31] and desktop applications [26, 32], the energy efficiency of applications within the same scope [2, 17], or even on how to predict energy consumption in several software systems [4, 14], among several other works.

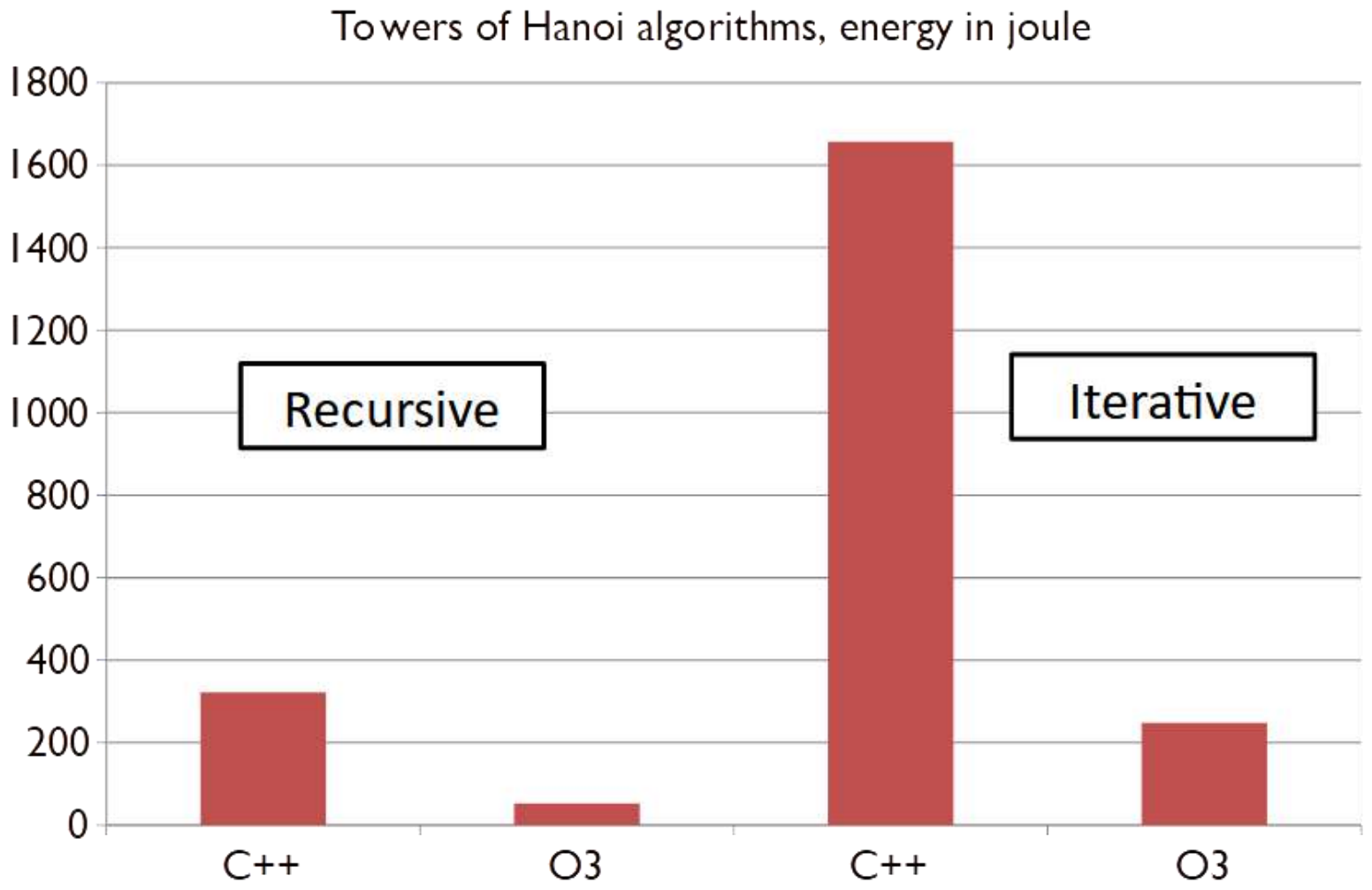
An interesting question that frequently arises in the software energy efficiency area is whether *a faster program is also an energy efficient program*, or not. If the answer is yes, then optimizing a program for speed also means optimizing it for energy, and this is exactly what the compiler construction community has been hardly doing since the very beginning of software languages. However, energy consumption does not depends only on execution time, as shown in the equation $Energy = Time \times Power$. In fact, there are several research works showing different results regarding

SLE'17, October 23–24, 2017, Vancouver, Canada

ACM ISBN 978-1-4503-5525-4/17/10…$15.00
https://doi.org/10.1145/3136014.3136031

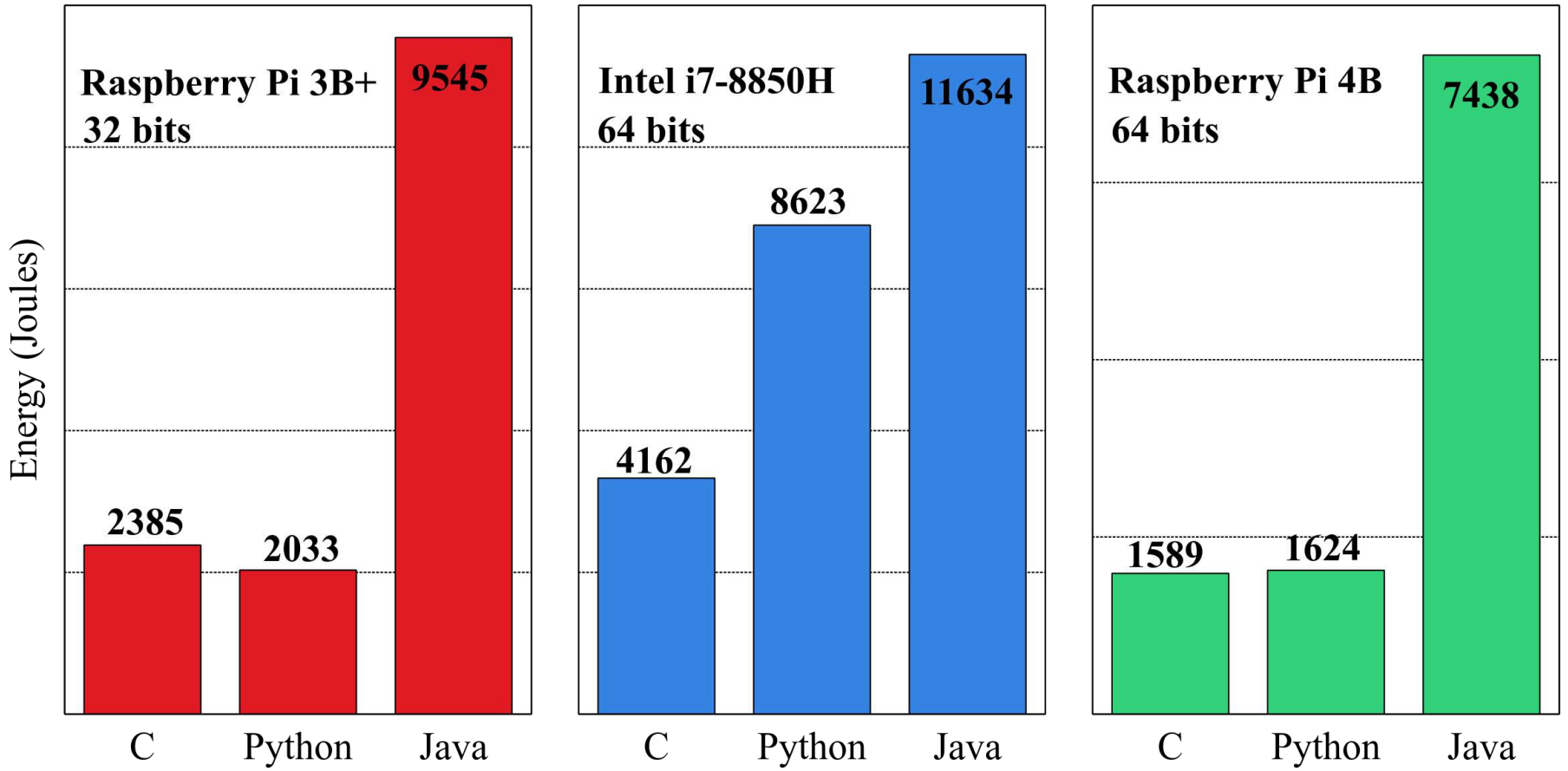
Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. 2017. Energy efficiency across programming languages: how do energy, time, and memory relate? In Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2017). Association for Computing Machinery, New York, NY, USA, 256–267.

Impact of optimization, algorithm, …



Biases in platforms, configurations, …

Ray casting algorithm: each version has different print output on the terminal, different OS/kernel, compiler & language versions



Don’t forget end users!

Awarness & Motivation

Green IT optimizations

- **Infrastructure optimizations:** data centers, servers, cooling, etc.
- **Hardware optimizations:** energy-efficient CPU architectures, low-power modes, efficient chip design, etc.
- **Software deployment and scheduling:** workload consolidation, energy-aware task scheduling, etc.
- **Coding optimization (Green Coding):** energy-aware programming, efficient algorithm design, green patterns, etc.

Green IT optimizations

- **Infrastructure optimizations:** data centers, servers, cooling, etc.
- **Hardware optimizations:** energy-efficient [illegible] ctures, low-power modes, efficient chip desi[illegible]
- **Software deploym[illegible]** consolidation, energy-aw[illegible]
- **Coding [illegible]ding):** energy-aware programming, efficient a[illegible], green patterns, etc.

End Users are absent from software optimizations

User behavior shapes software energy

Context	High-impact choice	Impact	Low-impact alternative
Browser (Chrome)	36 open tabs, active background scripts	13–28% faster battery drain	Throttle background scripts
Communication (Teams)	Audio + video (camera on)	up to 285%	Audio-only (camera off)
Productivity (Word)	Full IDE for a plain-text note	up to 97%	Lightweight editor
Streaming (YouTube)	720p →2160p (4K)	up to 214%	Lower playback quality

Adaptive Software Energy Feedback for Long-Term User Engagement. Thomas Zaragoza, Adel Noureddine, and Ernesto Exposito. In the 18th Symposium on Search-Based Software Engineering (SSBSE 2026). Montreal, Canada, July 2026.

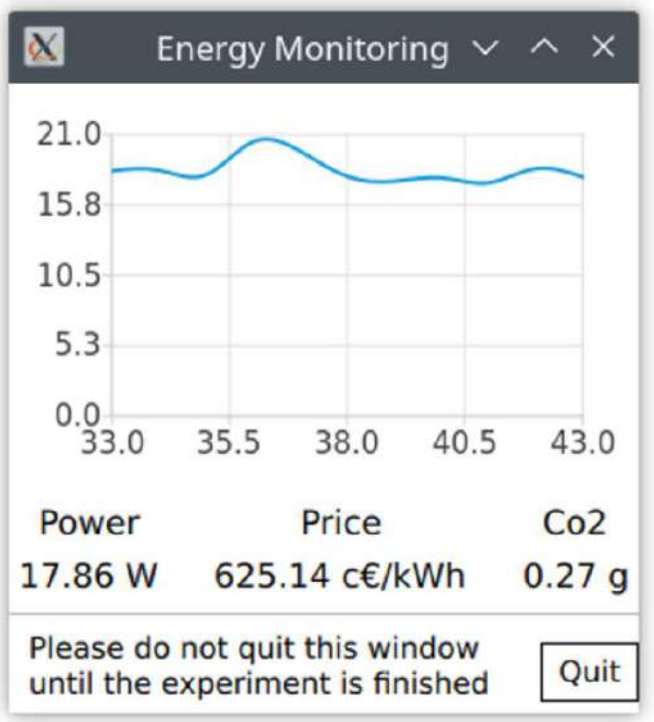
User behavior shapes software energy

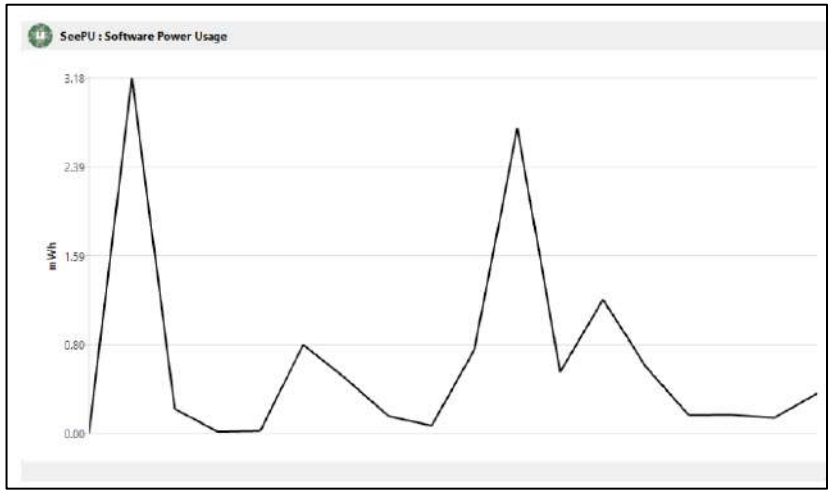
Context	High-impact choice	Impact	Low-impact alternative
Browser (Chrome)	36 open tabs, active background scripts	13–28% faster battery drain	Throttle background scripts
Communi (Teams)			era off)
Productivi (Word)	note		tor
Streaming (YouTube)	720p →2160p (4K)	up to 214%	Lower playback quality

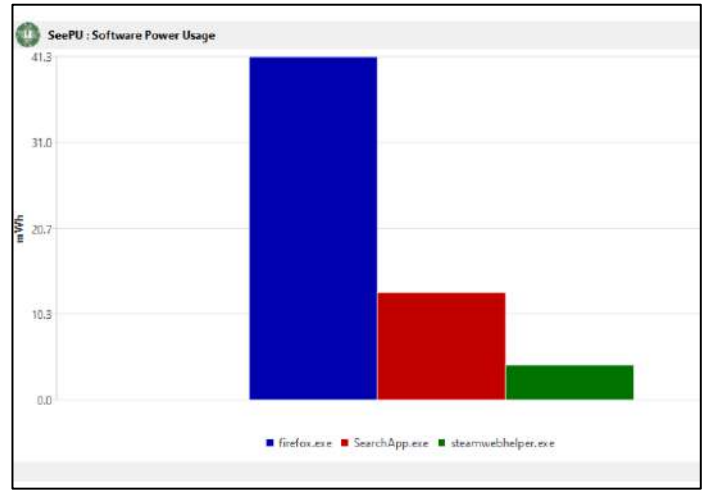
Same hardware, same task: user choices alone determine energy outcome

Adaptive Software Energy Feedback for Long-Term User Engagement. Thomas Zaragoza, Adel Noureddine, and Ernesto Exposito. In the 18th Symposium on Search-Based Software Engineering (SSBSE 2026). Montreal, Canada, July 2026.

Green feedback







The Impact of Green Feedback on Users' Software Usage. Adel Noureddine, Martín Diéguez Lodeiro, Noelle Bru, and Richard Chbeir. In IEEE Transactions on Sustainable Computing journal (T-SUSC). Volume 8, number 2, pages 280-292. April-June 2023.

Understanding and Influencing End-User Behavior in Software Energy Consumption. Thomas Zaragoza, Thibault Soulance, Adel Noureddine, and Ernesto Exposito. In the 29th International Conference on Evaluation and Assessment in Software Engineering (EASE 2025). Istanbul, Turkey, June 2025.

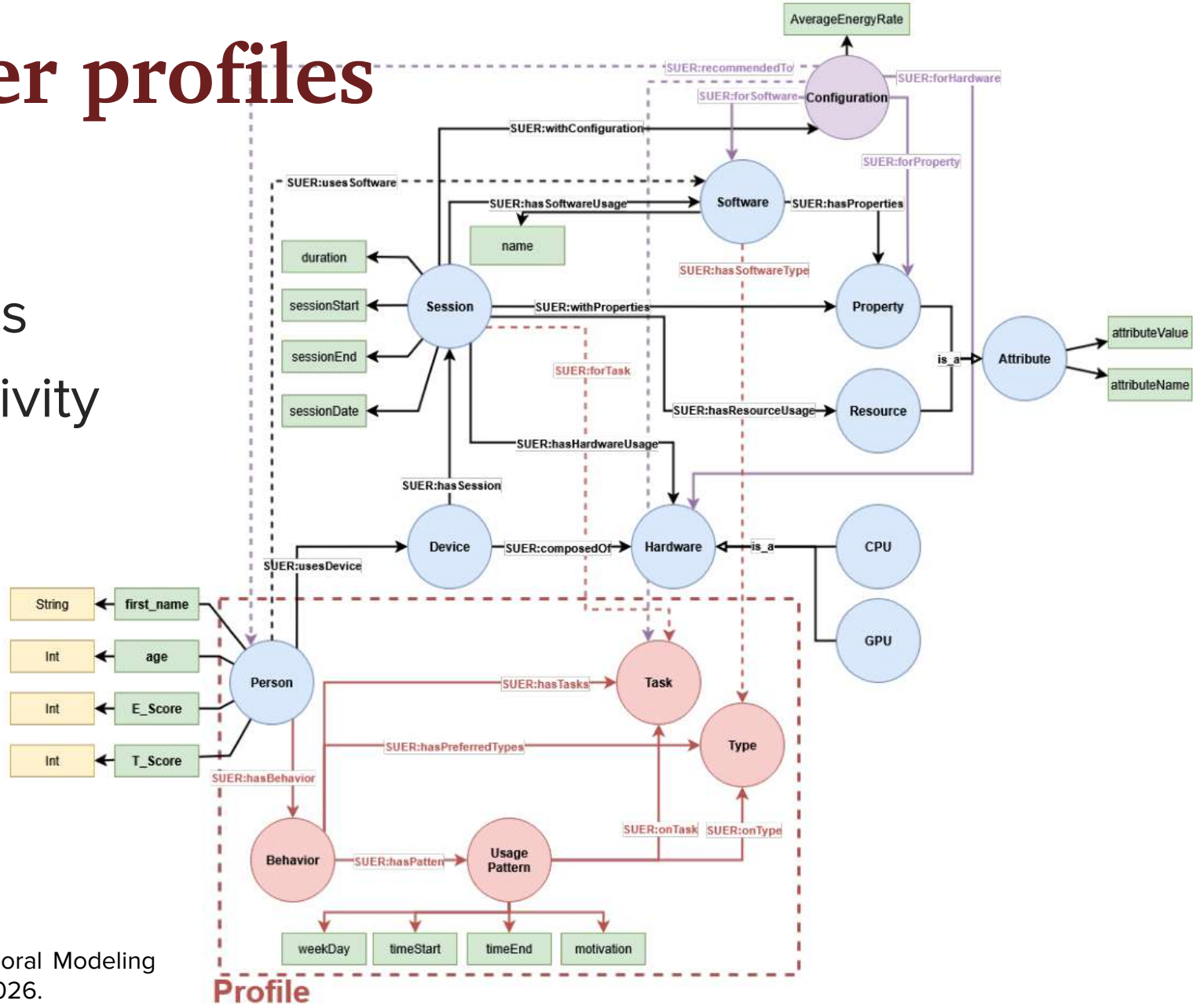
Personalized feedback…

- **Narrative**: *ex. historical self-comparison*
- **Frequency**: *weekly digest*
- **Representation**: *visual chart*
- **Metric**: *carbon-equivalent*
- **Visualization**: *bar chart*
- **Features**: *social comparison*

Adaptive Software Energy Feedback for Long-Term User Engagement. Thomas Zaragoza, Adel Noureddine, and Ernesto Exposito. In the 18th Symposium on Search-Based Software Engineering (SSBSE 2026). Montreal, Canada, July 2026.

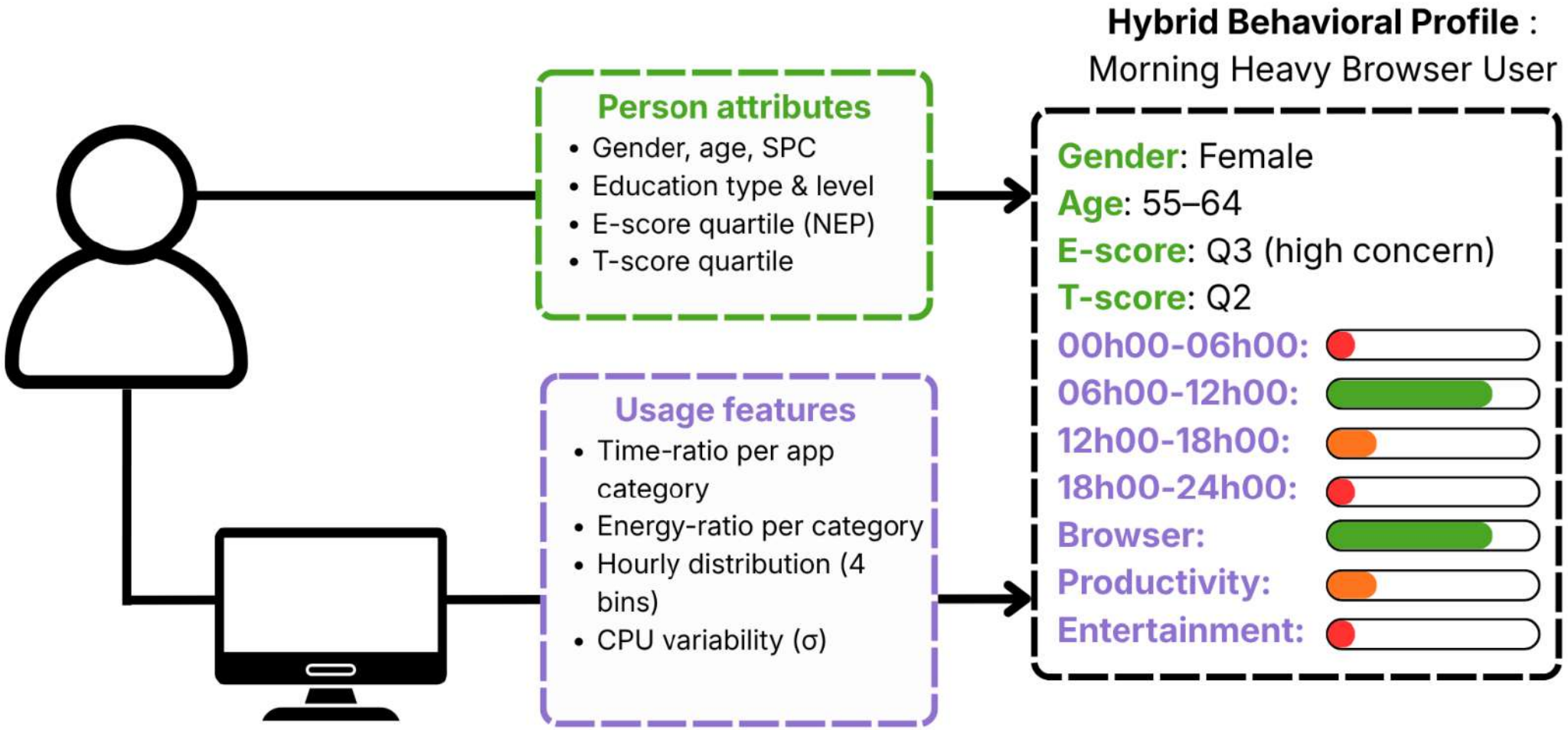
… to specific user profiles

- Socio-demographic
- Software usage patterns
- Environnemental sensitivity



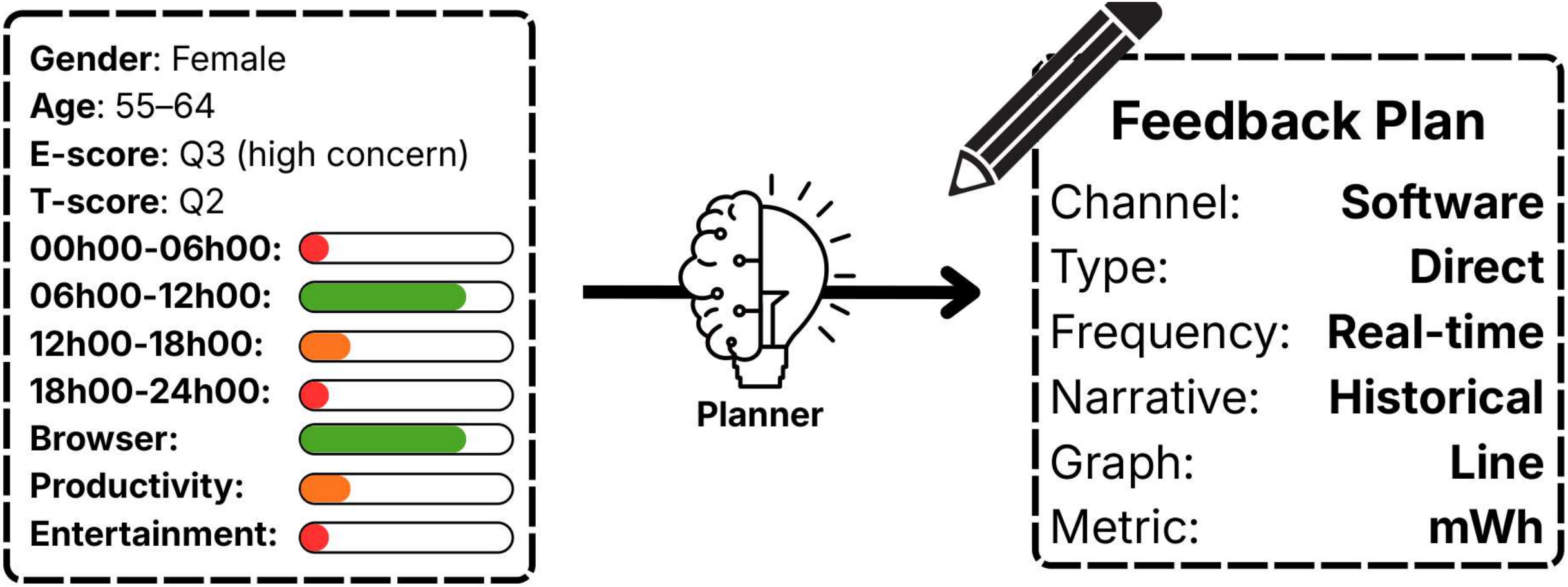
Adaptive Approach for Software Energy Reduction from Behavioral Modeling to Personalized Feedback. Thomas Zaragoza. PhD thesis, May 2026.

… to specific user profiles

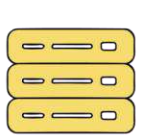


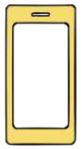
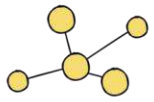
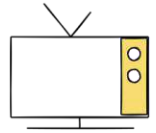
Adaptive Approach for Software Energy Reduction from Behavioral Modeling to Personalized Feedback. Thomas Zaragoza. PhD thesis, May 2026.

Personalized green feedback



Adaptive Approach for Software Energy Reduction from Behavioral Modeling to Personalized Feedback. Thomas Zaragoza. PhD thesis, May 2026.



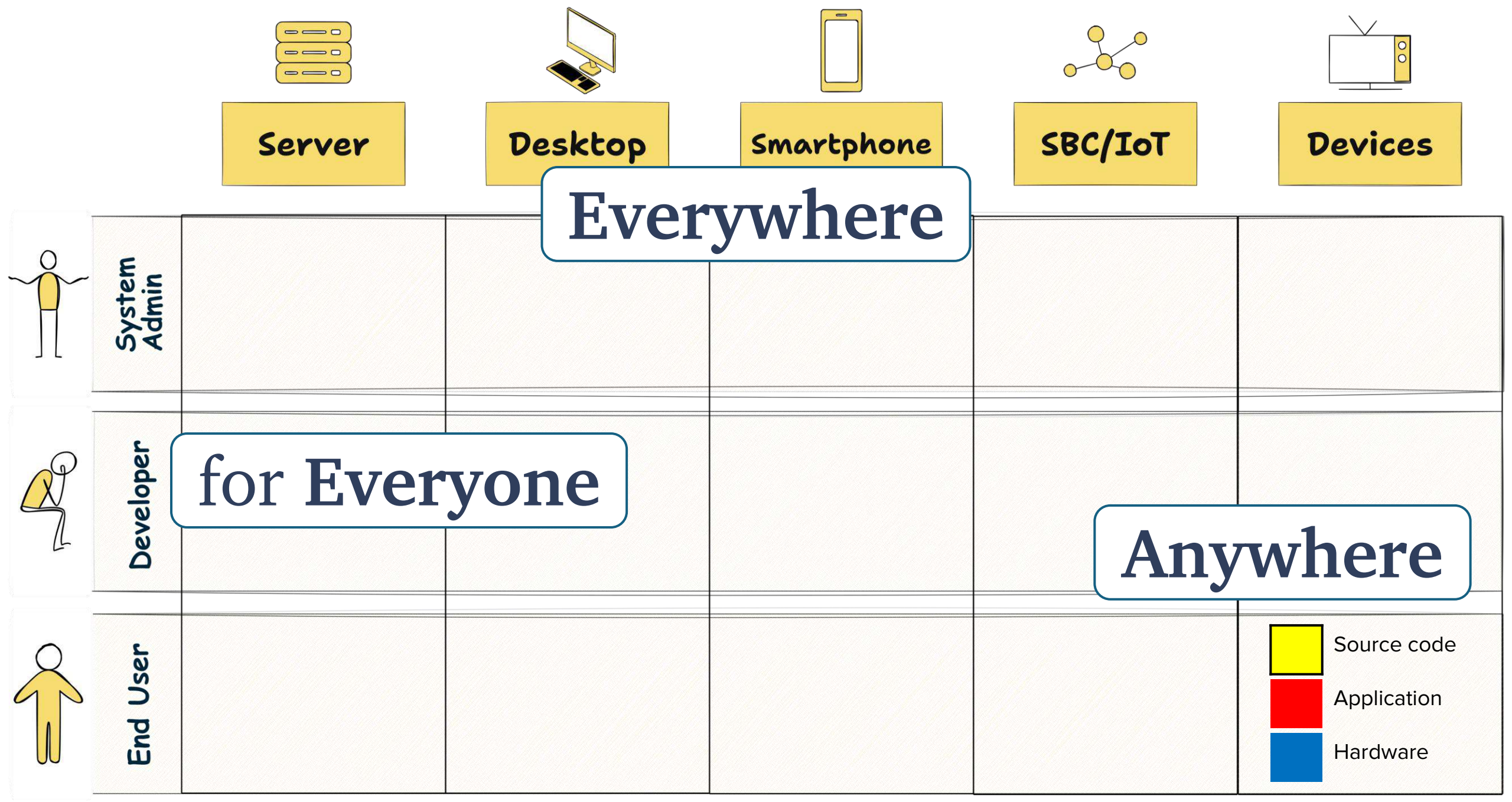




	Server	Desktop	Smartphone	SBC/IoT	Devices
System Admin					
Developer					
End User					

Source code

Application

Hardware


Server
Desktop
Smartphone
SBC/IoT
Devices
Everywhere
System Admin
Developer
for Everyone
Anywhere
End User
Source code
Application
Hardware

Monitor software energy (& environmental impacts):

Everywhere
Anywhere
for Everyone

Monitoring Energy Across Hardware, Software, and Humans

Prof. Adel Noureddine

University Paris Nanterre
Sorbonne University, CNRS, LIP6

DevOpsSustain workshop, @FSE 2026
Montreal, Canada – 05 July 2026

noureddine.org

Everywhere
Anywhere
for Everyone

Université Paris Nanterre · LIP6 · Sorbonne Université · cnrs