UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

Course Introduction

Seminar in Green Software Engineering

Dr. Pooja Rani
rani@ifi.uzh.ch, pooja.rani@uni-mannheim.de
University of Zurich, Switzerland
University of Mannheim, Germany

Overview

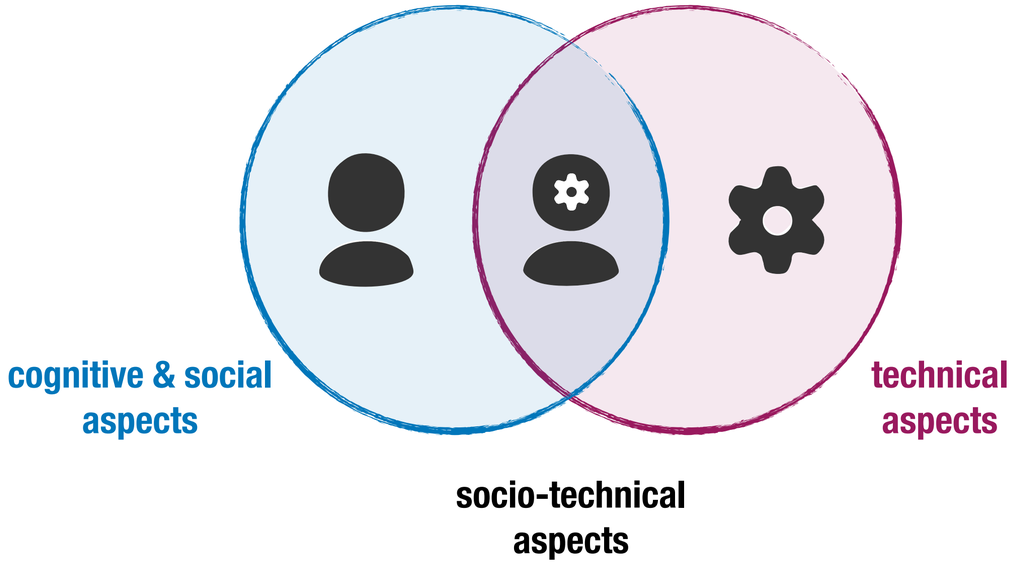
- How to do research (for the seminar and beyond)
- Process, topic, and evaluation of the course

Goal

You will learn to **analyze, summarize, and critique** scientific research, provide **constructive feedback**, and deliver **structured scientific reports and presentations** while evaluating current approaches and future directions.

Software Engineering: Challenges

- Creates practical, cost-effective software solutions
- People involved face various challenges



Research Work

- We try to understand how developers/ designers work and what problems they face.
- Identified the problems and now designing the possible solutions (e.g., tools, plugins)
- The solution is built, and now looking to evaluate it

How to collect and analyze data?

How to find out which approach is better?

How to validate the solution?

source: https://github.com/bvasiles/empirical-methods

Software Engineering

- Scientific use of quantitative and qualitative data to understand and improve software processes

- Starts with a good question:

  How do you support developers in following the coding guidelines?

- Leads to actionable results:

  Creation of tools, metrics, plugins, or improvement in processes

Understanding Research World

- Research papers
- Software engineering research fields
- Scientific conferences and journals
- Peer reviewing process
- Recognizing top conferences and journals

Producing a Literature Review

- Main sources for scientific publications
- Access to papers
- Search engines
- Mapping and systematic literature studies

Latex World

- Setup of an LATEX environment
- Online and offline editors
- Text formatting
- Figures and tables
- Structure a document
- Applying a template

source: https://github.com/bvasiles/empirical-methods

Evaluation

- Individual Report: 60%
- Peer Review: 20%
- Presentation: 20%

Platforms from University

Zoom/MS team: meetings

ILIAS: Study material, Question & answers, notifications

Schedule: Registration

- Fill out the Topic Selection form to register
- Indicate a ranking of preferences for the topics you would like to discover.
- Also, indicate some motivations for your choices.

Report

- Produce a report about a literature review on an assigned topic
- We present three main topics
- You have to come up with a subtopic to focus on

Report: Search related work

- Use sources for scientific research papers.
- Look at the proceedings of the main conferences and journals in software engineering.
- Apply the snowballing: starting from a target paper, follow the references/ citations.
- Always keep track of what you do.

Report: Search related work

- Google Scholar
- DBLP
- ACM Digital Library
- IEEE Explore
- Citeseer
- Elsevier (Science Direct)

search online by author, keyword, topic etc. on

Finding GOOD Papers

- Journal, conference, workshop
- Identify top venues in the field ICSE, FSE, ASE, MSR, ICPC, ICSME, GREENS
- Check the research track and length Research track, Industry track, Vision/NIER
- The CORE ranking is one way ICT4S, GREENS

CORE rankings for journals:http://portal.core.edu.au/jnl-ranks/?search=software+engineering&by=all&source=CORE2020&sort=atitle&page=1

CORE rankings for conference: http://portal.core.edu.au/conf-ranks/?search=software+engineering&by=all&source=CORE2021&sort=atitle&page=1

Reading Papers

- Read *critically and creatively*: be suspicious, and ask appropriate questions:
  - Are the authors solving the right problem? What are the limitations? Are the assumptions reasonable?
  - Can you trust its results? How are the results evaluated?
  - Can the results be generalised to other systems?
- What are the research methods used? Methodology? How is the dataset built? Can you summarise the paper?

https://www.eecs.harvard.edu/~michaelm/postscripts/ReadPaper.pdf

Report: Format

- The report must be written by using LaTeX and the IEEE double-column template
- Find good structure for sections:

  Abstract, Introduction, Related Work, Discussion, Future Work, Conclusion, References
- We expect correct and understandable English

Report: Anti-plagiarism check

- No room for cheating!
- We will run an anti-plagiarism tool. Better safe than sorry. . .

Literature review

- It’s now time to work on the literature review.
- You will have to upload your report to HotCRP for review. Upload an anonymous version of your document.

Paper bidding

- We’ll simulate a blind peer-reviewing process.
- First, you have to select the papers you seem to like the most (bidding).
- Only the titles and abstract will be visible.

Reviewing

- Second, you can start doing your 3 reviews
- Brief Summary for the report (3-4 lines)
- Technical Quality, Originality, and Significance
- Logical Structure, Presentation, and Style
- Overall Feedback

Discussion period

- Third, you will discuss with other anonymous reviewers to find a consensus.
- You'll have to decide on indications of revision to the author of the paper.
- It'll follow the notification to the authors.

Presentation

- We’ll have one or two presentation days. This is part of the grading.
- You need to present for 12 minutes with an additional 5 minutes for Q&A.

Final manuscript revision

- You’ll have some time to finalize the report. Follow the indications of the reviewers.
- Submit it as an assessment on HotCRP.

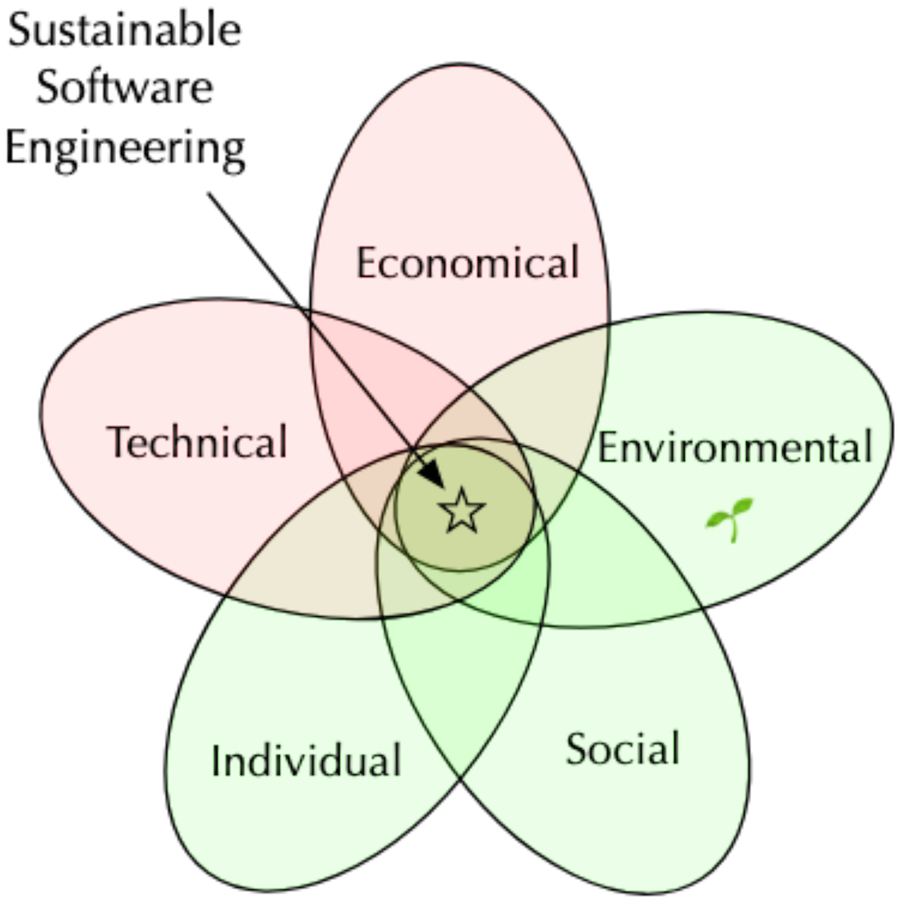
Topic and Process

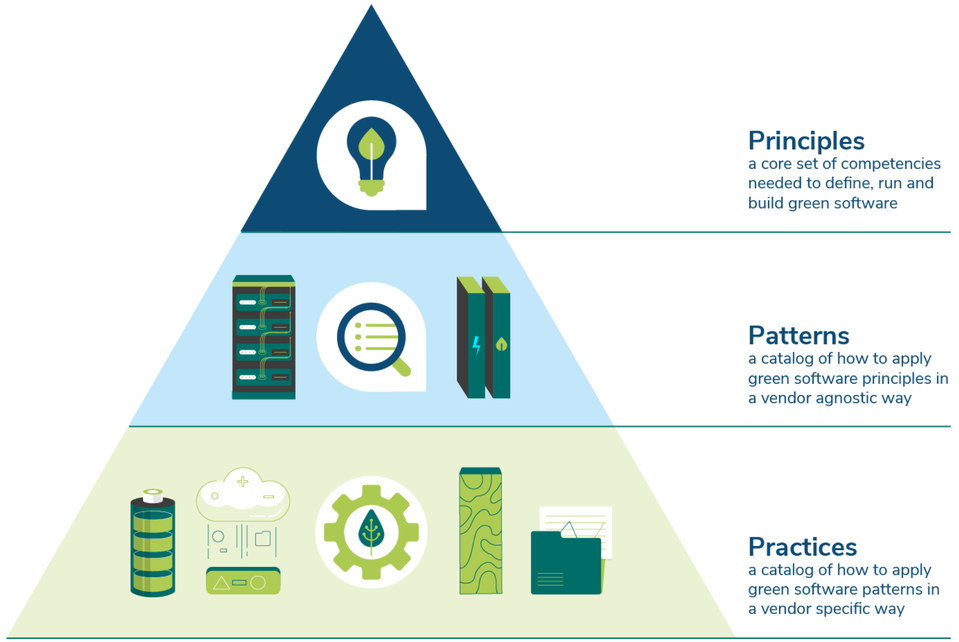
Energy patterns for popular programming languages

Energy-efficient LLMs for software developers

Software-only sustainability guidelines for greener software

Further projects: OLAT





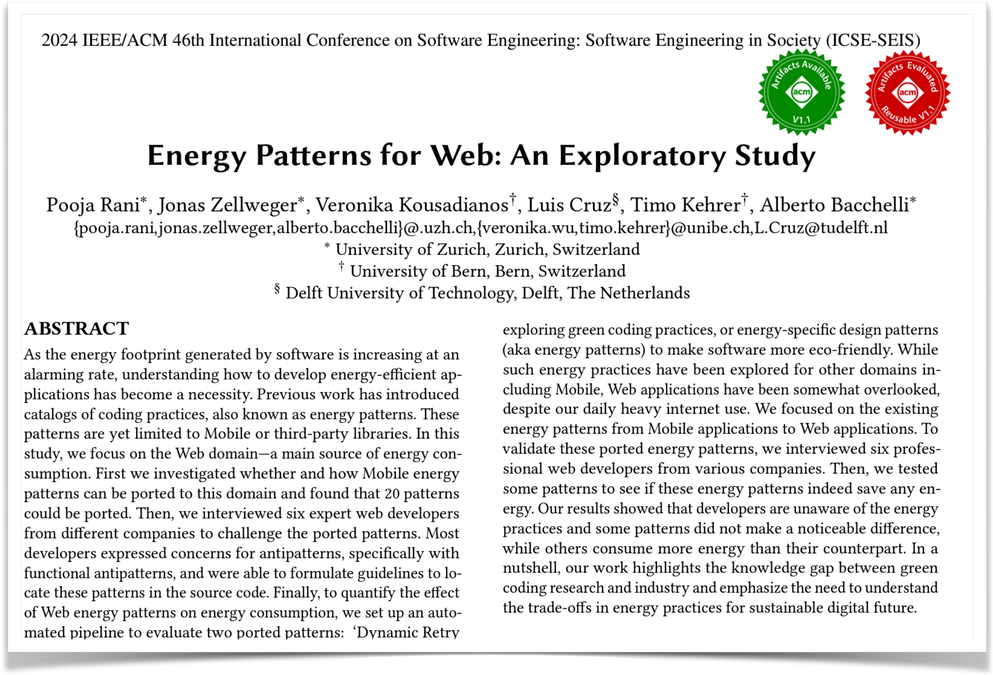
Principles
a core set of competencies needed to define, run and build green software

Patterns
a catalog of how to apply green software principles in a vendor agnostic way

Practices
a catalog of how to apply green software patterns in a vendor specific way

Green Software Foundation
greensoftware.org

Energy patterns for web applications



2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)

Energy Patterns for Web: An Exploratory Study

Pooja Rani*, Jonas Zellweger*, Veronika Kousadianos†, Luis Cruz§, Timo Kehrer†, Alberto Bacchelli*
{pooja.rani,jonas.zellweger,alberto.bacchelli}@.uzh.ch,{veronika.wu,timo.kehrer}@unibe.ch,L.Cruz@tudelft.nl
* University of Zurich, Zurich, Switzerland
† University of Bern, Bern, Switzerland
§ Delft University of Technology, Delft, The Netherlands

ABSTRACT

As the energy footprint generated by software is increasing at an alarming rate, understanding how to develop energy-efficient applications has become a necessity. Previous work has introduced catalogs of coding practices, also known as energy patterns. These patterns are yet limited to Mobile or third-party libraries. In this study, we focus on the Web domain—a main source of energy consumption. First we investigated whether and how Mobile energy patterns can be ported to this domain and found that 20 patterns could be ported. Then, we interviewed six expert web developers from different companies to challenge the ported patterns. Most developers expressed concerns for antipatterns, specifically with functional antipatterns, and were able to formulate guidelines to locate these patterns in the source code. Finally, to quantify the effect of Web energy patterns on energy consumption, we set up an automated pipeline to evaluate two ported patterns: 'Dynamic Retry exploring green coding practices, or energy-specific design patterns (aka energy patterns) to make software more eco-friendly. While such energy practices have been explored for other domains including Mobile, Web applications have been somewhat overlooked, despite our daily heavy internet use. We focused on the existing energy patterns from Mobile applications to Web applications. To validate these ported energy patterns, we interviewed six professional web developers from various companies. Then, we tested some patterns to see if these energy patterns indeed save any energy. Our results showed that developers are unaware of the energy practices and some patterns did not make a noticeable difference, while others consume more energy than their counterpart. In a nutshell, our work highlights the knowledge gap between green coding research and industry and emphasize the need to understand the trade-offs in energy practices for sustainable digital future.

Dark UI Colors

Provide a dark UI colour theme to save battery.

Dynamic Retry Delay

Whenever an attempt to access a resource has failed, increase the interval of time waited before asking it

Open Only When Necessary

Open/start resources/services only when they are strictly necessary.

Push Over Poll

Use push notifications to receive updates from resources instead of actively querying resources (polling)

Power Awareness

Have a different behavior when device is connected/disconnected to a power station

https://tqrg.github.io/energy-patterns/#/

Energy patterns for popular programming languages

GreenPy: Evaluating Application-Level Energy Efficiency in Python for Green Computing

Nurzihan Fatema Reya, Abtahi Ahmed, Tashfia Zaman and Md. Motaharul Islam*

United International University, Dhaka, Bangladesh
nreya201085@bscse.uiu.ac.bd; aahmed202247@bscse.uiu.ac.bd; tzaman201141@bscse.uiu.ac.bd;
motaharul@cse.uiu.ac.bd
*Correspondence: motaharul@cse.uiu.ac.bd

Received: 5th May 2023; Accepted: 27th June 2023; Published: 1st July 2023

Abstract: The increased use of software applications has resulted in a surge in energy demand, particularly in data centers and IT infrastructures. As global energy consumption is projected to surpass supply by 2030, the need to optimize energy consumption in programming has become imperative. Our study explores the energy efficiency of various coding patterns and techniques in Python, with the objective of guiding programmers to a more informed and energy-conscious coding practices. The research investigates the energy consumption of a comprehensive range of topics, including data initialization, access patterns, structures, string formatting, sorting algorithms, dynamic programming and performance comparisons between NumPy and Pandas, and personal computers versus cloud computing. The major findings of our research include the advantages of using efficient data structures, the benefits of dynamic programming in certain scenarios that saves up to 0.128J of energy, and the energy efficiency of NumPy over Pandas for numerical calculations. Additionally, the study also shows that assignment operator, sequential read, sequential write and string concatenation are 2.2 times, 1.05 times, 1.3 times and 1.01 times more energy-efficient choices, respectively, compared to their alternatives for data initialization, data access patterns, and string formatting. Our findings offer guidance for developers to optimize code for energy efficiency and inspire sustainable software development practices, contributing to a greener computing industry.

Sorting Algorithm

Which version of the sorting algorithm

Loop operations

For loop vs list operations

Data types

Data structures

Built-in Functions vs custom ones

Which functions to use?

Reya et al., 2023, ""GreenPy: Evaluating Application-Level Energy Efficiency in Python for Green Computing"

https://data-intelligence.hashnode.dev/how-to-use-python-dict-in-energy-efficient-way-some-useful-tips

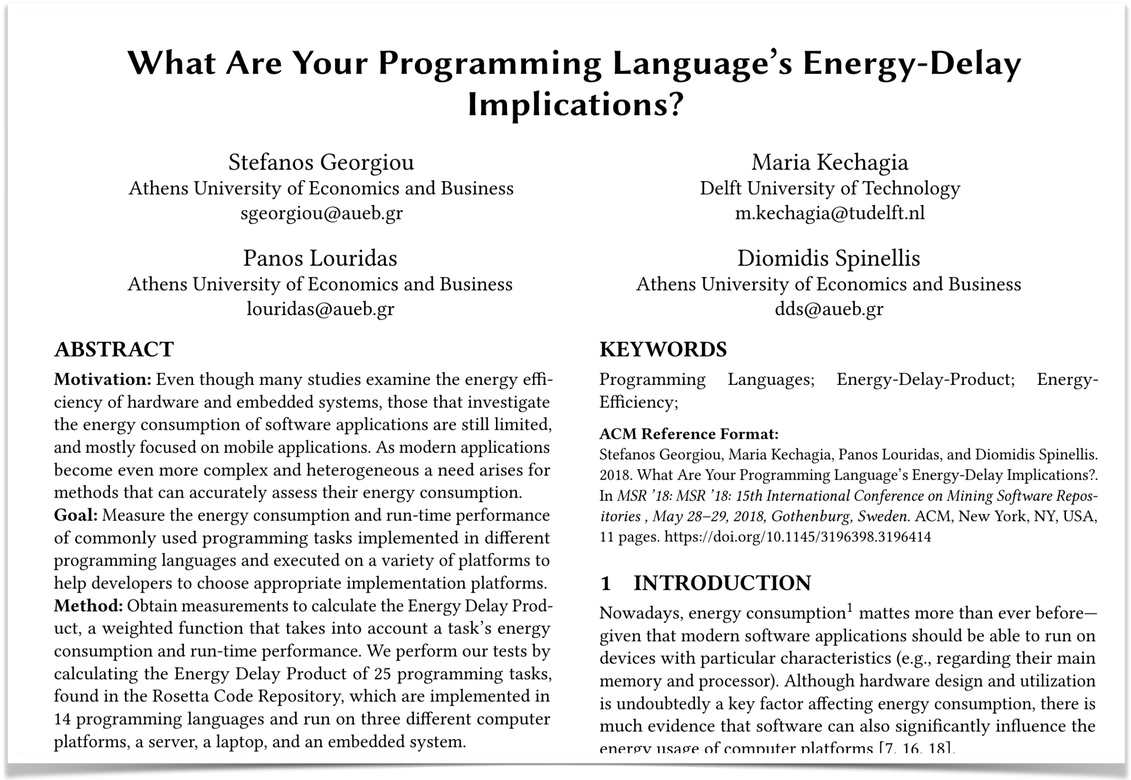
Energy patterns for popular programming languages

What programming concepts play a role in writing efficient code (energy efficiency, time efficiency)?

What kind of empirical research methods (controlled experiment, survey) are used to assess the impact of these coding practices?

How do these coding practices or patterns impact the code's sustainability?

Energy patterns for popular programming languages



What Are Your Programming Language's Energy-Delay Implications?

Stefanos Georgiou
Athens University of Economics and Business
sgeorgiou@aueb.gr

Maria Kechagia
Delft University of Technology
m.kechagia@tudelft.nl

Panos Louridas
Athens University of Economics and Business
louridas@aueb.gr

Diomidis Spinellis
Athens University of Economics and Business
dds@aueb.gr

ABSTRACT

Motivation: Even though many studies examine the energy efficiency of hardware and embedded systems, those that investigate the energy consumption of software applications are still limited, and mostly focused on mobile applications. As modern applications become even more complex and heterogeneous a need arises for methods that can accurately assess their energy consumption.
Goal: Measure the energy consumption and run-time performance of commonly used programming tasks implemented in different programming languages and executed on a variety of platforms to help developers to choose appropriate implementation platforms.
Method: Obtain measurements to calculate the Energy Delay Product, a weighted function that takes into account a task's energy consumption and run-time performance. We perform our tests by calculating the Energy Delay Product of 25 programming tasks, found in the Rosetta Code Repository, which are implemented in 14 programming languages and run on three different computer platforms, a server, a laptop, and an embedded system.

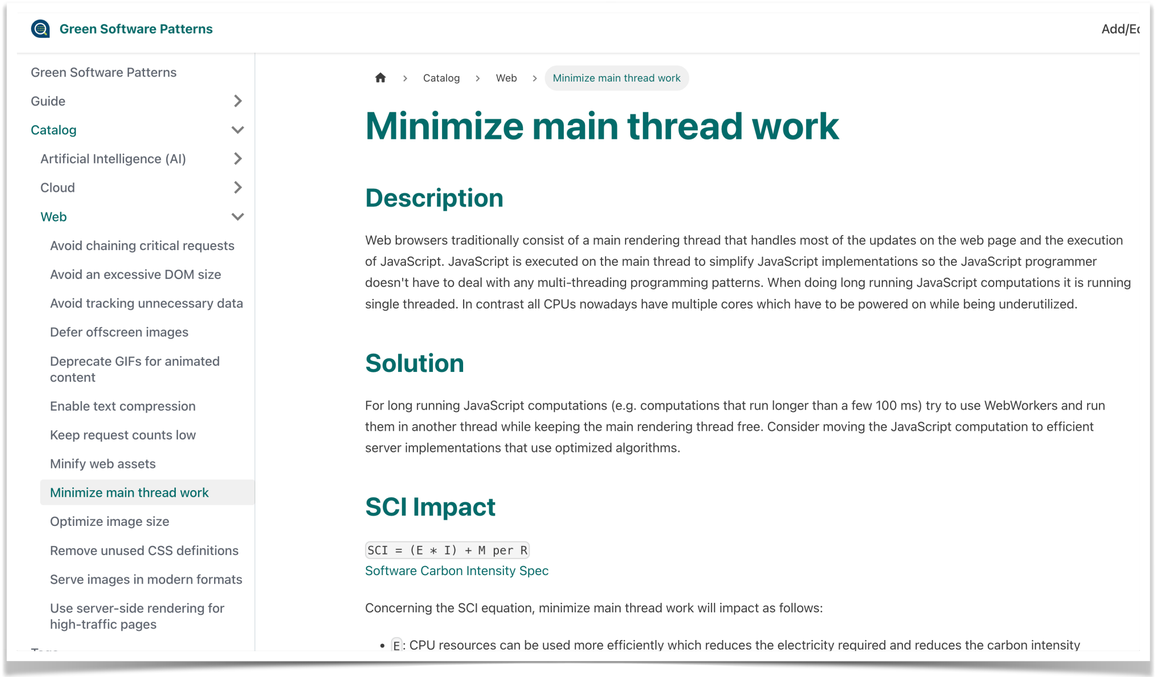
KEYWORDS

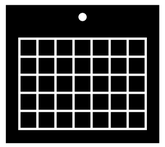
Programming Languages; Energy-Delay-Product; Energy-Efficiency;

ACM Reference Format:
Stefanos Georgiou, Maria Kechagia, Panos Louridas, and Diomidis Spinellis. 2018. What Are Your Programming Language's Energy-Delay Implications?. In *MSR '18: MSR '18: 15th International Conference on Mining Software Repositories , May 28–29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3196398.3196414

1 INTRODUCTION

Nowadays, energy consumption[1] mattes more than ever before—given that modern software applications should be able to run on devices with particular characteristics (e.g., regarding their main memory and processor). Although hardware design and utilization is undoubtedly a key factor affecting energy consumption, there is much evidence that software can also significantly influence the energy usage of computer platforms [7, 16, 18].





Energy-efficient LLMs for software developers

Understand a LLM’s performance

through key metrics for green software development

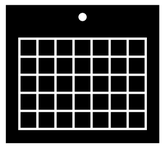
Help developers and managers make informed decisions

through key metrics for green software development

Collect green practices for LLMs for SE tasks

bring evidence for the LLMs

https://github.com/Green-Software-Foundation/awesome-green-software



Energy-efficient LLMs for software developers

Which software engineering tasks are often studied for energy-efficient LLM use?

What efficiency techniques have been proposed for reducing energy in developer-facing LLM tools?

What actionable recommendations can you provide to developers for choosing energy-efficient designs for developer tools?

https://github.com/Green-Software-Foundation/awesome-green-software

https://luiscruz.github.io/2023/05/13/energy-units.html

Energy-efficient LLMs for software developers

Paula, E., Soni, J., Upadhyay, H. et al. Comparative analysis of model compression techniques for achieving carbon-efficient AI. Sci Rep 15, 23461 (2025). https://doi.org/10.1038/s41598-025-07821-w

Walkowiak, T., 2025, May. Energy Efficiency in Large Language Models: An Empirical Study. In *International Conference on Dependability and Complex Systems* (pp. 221-228). Cham: Springer Nature Switzerland.

Ilager, S., Florian Briem, L. and Brandic, I., 2025. Green-code: Optimizing energy efficiency in large language models for code generation. *arXiv e-prints*, pp.arXiv-2501.

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

Choose your topic by filling the google form

Dr. Pooja Rani
rani@ifi.uzh.ch, pooja.rani@uni-mannheim.de
University of Zurich, Switzerland
University of Mannheim, Germany