

GREEN SOFTWARE MODEL: GUIDANCE FOR PRODUCERS

Version 1.0 (December 2024)

Colophon

This document was reviewed and approved by the following people:

Chushu Gao
Senior Researcher

Pepijn van de Kamp
Head of Research and Innovation

chushu.gao@softwareimprovementgroup.com

pepijn.vandekamp@softwareimprovementgroup.com

TABLE OF CONTENTS

1.	Introduction	3
2.	Green Software Consultancy Model	4
3.	System properties	5
3.1	Code Efficiency.....	5
3.2	Technology Efficiency	5
3.3	Location Awareness	5
3.4	Elastic Scalability	5
3.5	Hosting Strategy.....	6
3.6	Architecture Efficiency.....	6
3.7	Embodied Carbon Strategy	6
3.8	Observability	7
4.	References	8

1. INTRODUCTION

Software systems contribute to carbon emissions through the energy consumed by the hardware they operate on and the emissions associated with manufacturing this hardware. As the environmental impact of digital technologies continues to grow, it becomes imperative to design and develop software systems with sustainability in mind.

The primary motivation for developing a green software model is to empower developers, architects, and other stakeholders to make informed decisions about tools, architectures, and practices that minimize carbon emissions. This document describes the SIG evaluation criteria for sustainability of software systems.

2. GREEN SOFTWARE CONSULTANCY MODEL

The newly published ISO/IEC 21031:2024 standard, Software Carbon Intensity (SCI) Specification [SCI 2024], establishes a methodology for calculating the carbon emission rate of a software system.

SCI categorizes the software sustainability actions into following three categories:

- **Energy Efficiency:** Actions taken to make software use less electricity to perform the same function.
- **Hardware Efficiency:** Actions taken to make software use fewer physical resources to perform the same function.
- **Carbon Awareness:** Actions taken to time- or region-shift software computation to take advantage of cleaner, more renewable or lower carbon sources of electricity.

The model selects the sub-characteristics which are in line with SCI specification. The model uses eight system properties that are investigated by SIG and explained in more detail.

ISO/IEC 21031 Software Carbon Intensity	Code efficiency	Technology efficiency	Location awareness	Resource elasticity	Hosting strategy	Architecture efficiency	Embodied carbon strategy	Observability
Energy Efficiency	■	■		■		■		■
Hardware Efficiency	■			■	■	■	■	■
Carbon Awareness			■	■				■

3. SYSTEM PROPERTIES

3.1 CODE EFFICIENCY

Code Efficiency focuses on the degree of optimization applied to speed up individual steps or reduce the number of steps or resources used by a single transaction. It contributes to sub-characteristics of Energy Efficiency and Hardware Efficiency:

- **Energy Efficiency:** Reducing the execution time of single transaction usually results in reducing energy consumption overall.
- **Hardware Efficiency:** Reducing resources used by a single transaction also reduces amortized embodied carbon.

The degree of code efficiency is evaluated based on the findings of common inefficient code patterns. A low rating will typically result from many inefficient code patterns found in the code base.

3.2 TECHNOLOGY EFFICIENCY

Different programming languages, when solving the same problem, often exhibit significant performance differences due to their language characteristics, runtime mechanisms, and design philosophies. This is also hold for Energy Efficiency. Technology Efficiency depicts the degree of Energy Efficiency of using different programming language to accomplish the same task.

Technology Efficiency is based on the energy profiling and technology distributions. A low rating will result from a large portion of the system implemented by a less energy efficient language.

3.3 LOCATION AWARENESS

Where in the world and when the software system is hosted has a significant impact on the carbon emission. For example, electricity consumed in North EU has a 17 times lower carbon intensity than consumed in Central EU. Location Awareness indicates the level of carbon emission rate of electricity used for hosting and running the software.

A low rating will result from the software system mostly hosted on a high carbon intensity location.

3.4 ELASTIC SCALABILITY

Elastic Scalability represents the ability to dynamically scale its resource consumption based on the loads of the application. It contributes to all three sub-characteristics:

- **Energy Efficiency:** Switching off or scale down hardware reduces the idle energy consumption.
- **Hardware Efficiency:** Elastic scaling means hardware resources can be switched off when load is low, while they are switched off they do not count towards the amortization of embodied carbon.
- **Carbon Awareness:** Elastic scaling is a prerequisite for being able to time-shift or region-shift workloads.

A low rating will result from the lack of dynamically configurable resource provision.

3.5 HOSTING STRATEGY

Data centers come in various types, ranging from on-premises facilities to public clouds, each hosting and managing computing resources based on distinct design and management strategies. The hosting strategy can involve any type of data center or a combination of them, with dedicated or shared infrastructure. Choosing different hosting strategies can significantly impact hardware efficiency. It is widely acknowledged that using public clouds and shared infrastructure will increase hardware efficiency and reduce the overall energy impacts.

To enable flexibility in hosting strategy, software systems should be designed to allow easy deployment or migration across various hosting environments. For example, hardcoded configurations can make it difficult for an application to migrate to greener cloud services, hence leading to a lower rating.

3.6 ARCHITECTURE EFFICIENCY

Architecture Efficiency will take into account architecture quality of software systems, architecture styles and architectural tactics and patterns applied to the software systems.

The architecture quality defines the degree to which an architecture is flexible and adaptable to change. Architecture quality contributes to the degree of easiness for a software system to adapt for improved sustainability.

Architecture style encompasses various paradigms for organizing software systems, ranging from monolithic and layered architectures to microservices architecture. The architecture style can significantly impact energy efficiency and hardware efficiency. For example, microservice architecture significantly enhances scalability and resource efficiency, as each microservice can be deployed and scaled independently.

Architectural tactics and patterns are design decisions aimed at achieving a specific quality attribute. These tactics and patterns focus on a single quality attribute and can be applied at any stage of the software development process. When targeting sustainability, architectural tactics and patterns, such as caching mechanisms help developers design software that minimizes environmental impact by optimizing energy efficiency and resource utilization.

Higher architectural quality rating suggests software systems to adapt more easily toward energy efficiency and hardware efficiency, for example, by incorporating architectural tactics and patterns.

3.7 EMBODIED CARBON STRATEGY

Embodied carbon emission is the total carbon emissions associated with the production, transportation, and assembly of a hardware device before it is operational. The embodied carbon cost of a software system is amortized over the hardware's operational lifespan.

For software practitioner, they can decrease the embodied carbon cost of software systems by extend the lifetime of the hardware or increase the utilization of the hardware.

3.8 OBSERVABILITY

To accurately assess software sustainability and support problem analysis and resolution, it is crucial to measure and monitor the software system. The system's design and operation should facilitate this process.

A low rating results if essential data on the system behavior is not collected or not preserved over longer period for trend analysis. Also all relevant types of data should be observable: energy consumption, resource consumption, response times, throughput and workload fluctuations.


4. REFERENCES

[SCI 2024] ISO/IEC. Information technology - Software Carbon Intensity (SCI) specification (ISO/IEC 21031:2024), <https://github.com/Green-Software-Foundation/sci/blob/main/SPEC.md>



Fred. Roeskestraat 115
1076 EE Amsterdam
The Netherlands

www.softwareimprovementgroup.com
marketing@softwareimprovementgroup.com

 Getting software right for a healthier digital world