MAST

Managing Sustainability Tradeoffs

Grant Agreement No.: 22035

D2.2 – Sustainability State of the Art and Practice

Document Number	D2.2
Document Title	Sustainability State of the Art and Practice
Version	1.0
Status	Draft
Work Package	WP2
Deliverable Type	Report
Contractual Date of Delivery	M6 – August 2025
Actual Date of Delivery	M6 – 1 September 2025
Responsible Unit	Wirtek

Contributors	WRT, CLW, CPP
Keyword List	State of the Art
Dissemination level	CC

Amendment History

Version	Date	Author (Unit)	Description
0.1	07/2025	WRT	Initial Draft with Template
0.2	08/2025	WRT	Refined content, added references, added same content format to Use Case SotA
0.3	08/2025	WRT	Rewrite of Executive Summary to match structure and academic style
0.4	08/2025	RUG	Refined document and added further details on related research work
1.0	09/2025	CPP	Upload of version 1.0 to ITEA portal

Executive Summary

This deliverable establishes the **baseline understanding of sustainability** in software engineering and system operations within the MAST project. It synthesizes **state-of-the-art research**, **current industrial practices**, and **use case–specific analyses** to guide the development of sustainability-aware tools and methodologies in upcoming project phases.

MAST brings together a **diverse consortium of industrial and academic partners**, spanning software development, IoT platforms, renewable energy management, and industrial manufacturing. This multidisciplinary foundation enables a **holistic approach** to embedding sustainability as a **first-class engineering attribute**, addressing both **technical sustainability** (maintainability, technical debt management) and **environmental sustainability** (energy consumption, carbon footprint).

The document:

- Reviews **academic research and related European initiatives**, identifying advancements in technical debt management, energy efficiency, and trade-off analysis.
- Analyzes **current sustainability practices** across the consortium, highlighting systemic gaps such as fragmented metrics, lack of real-time sustainability feedback, and absence of decision-support tools.
- Presents detailed **use case assessments** for Wirtek, Cleanwatts, and Canon Production Printing, providing practical insights into current baselines and sustainability challenges.
- Defines how these findings will **inform innovations** in multi-objective optimization, AI-powered dashboards, and sustainability-aware DevOps integration developed in **WP3–WP5**.

By consolidating these insights, the deliverable lays a **scientific and practical foundation** for MAST to:

- Develop methodologies and tools that enable measurable sustainability improvements across diverse industrial domains.
- Support informed trade-offs between performance, maintainability, and environmental impact.
- Contribute to **European digital sustainability goals** and future standards in sustainable software and system engineering.

Partners: WRT, CLW, CPP

Table of Contents

1 Introduction

Sustainability has become a strategic priority in the evolution of digital technologies, influencing software engineering practices, ICT infrastructure design, and industrial system operations. As organizations face increasing pressure to address environmental impact, technical sustainability, and regulatory requirements, there is a growing need to **embed sustainability as a first-class engineering concern** across the entire software and system lifecycle.

The **MAST project** brings together a **diverse consortium** of industrial and academic partners from domains including software development, IoT platforms, renewable energy management, and industrial manufacturing. This cross-disciplinary collaboration provides a unique foundation for developing **novel methodologies and tools** that integrate **technical debt management, energy efficiency optimization, and multi-objective trade-off analysis** into mainstream development workflows.

This deliverable, **D2.2 Sustainability State of the Art and Practice**, serves as a **critical baseline** for the project. It:

- Provides a comprehensive review of **state-of-the-art research and related European initiatives**, identifying academic and technological advances in sustainable software engineering.
- Analyzes **current industrial practices** within the consortium, highlighting existing capabilities and systemic gaps.
- Presents **use case–specific assessments** for Wirtek, Cleanwatts, and Canon Production Printing, demonstrating practical sustainability challenges and opportunities for innovation.
- Maps how these findings will inform **upcoming work packages (WP3–WP5)**, ensuring that future tools and methods are grounded in real-world needs.

The document is structured as follows:

- **Section 2** describes the academic and industrial state of the art in sustainability, covering technical and environmental dimensions.
- **Section 3** analyzes current sustainability practices across consortium partners.
- **Section 4** provides detailed, use case–specific sustainability baselines and identified gaps.

- **Section 5** outlines innovations and technological advances that MAST will introduce.
- **Section 6** details the expected outputs and impact of the project.
- **Section 7** concludes the deliverable and defines next steps toward WP3–WP5.

By consolidating these insights, this deliverable establishes a **foundation for measurable sustainability improvements**, enabling MAST to design and validate solutions that are scientifically rigorous, industrially relevant, and aligned with European digital sustainability goals.

2 General State of the Art on Sustainability in Software Engineering

Software development has continued to grow rapidly over the years, leading to increasingly complex and power hungry applications. The growth is particularly evident in areas such as Artificial Intelligence (AI), where successive versions of the AI model consume more computational power, often requiring advanced cooling and infrastructure. As a result, the environmental aspect of software engineering has become a growing concern. At the same time, technical sustainability remains a central focus in software engineering. Technical sustainability reflects how well software can evolve over time without requiring extra effort or cost. Thereby, in recent years, there has been a shift in both industry and academia towards more environmentally sustainable software development practices.

2.1 Technical Sustainability

Technical sustainability refers to the ability of a software system to evolve cost-effectively over time without incurring prohibitive maintenance costs or quality degradation (Koziolek, 2011). Lehman's software evolution laws established early foundations for understanding long-term adaptability (Lehman, 1996). Subsequent research revealed correlations between modularity, complexity, and maintenance effort (Mens et al., 2014).

The dominant conceptual model is **technical debt (TD)**, representing short-term design choices that increase future maintenance costs (Avgeriou et al., 2016). TD manifests in multiple forms—requirements, architecture, code, or testing—and is often invisible until it compounds into critical quality risks (Izurieta et al., 2017). Architectural technical debt (ATD), incurred early in development, has particularly long-lasting impacts (Ampatzoglou et al., 2016).

Industrial tools like **SonarQube**, **Designite**, and **Cast Highlight** measure TD via metrics such as code complexity, duplication, and coupling. These tools vary in models and remediation strategies,

creating confusion for practitioners (FPP, §2.3.1). Empirical studies show that proactive clean coding and refactoring outperform reactive debt repayment in sustaining software health (Potdar & Shihab, 2014).

2.2 Environmental Sustainability

Environmental sustainability addresses **energy consumption and carbon footprint** of software during execution. Within **Green Computing**, research aims to reduce environmental impact while maintaining performance (Murugesan, 2008).

Software design choices influence energy usage across hardware, networking, and storage (Pereira et al., 2017). Programming languages, algorithms, and deployment strategies can produce up to 50% variations in energy consumption (FPP, §2.3.1). Cloud computing complicates this further with elastic resources and shared infrastructures (Gill & Buyya, 2018).

Tools like **GreenSource**, **JoularJX**, and **PowerAPI** estimate software energy use via static analysis or dynamic profiling. However, integration into mainstream development workflows remains limited. Hyperscalers (AWS, Azure, GCP) provide only coarse-grained carbon dashboards (FPP, §2.3.1).

Converting energy usage into carbon emissions is also challenging due to fluctuating renewable energy mixes and multi-tenant architectures (Caron et al., 2022). Regulatory initiatives like the **EU Green Deal** and **Corporate Sustainability Due Diligence Directive** increasingly require ICT organizations to quantify software-driven environmental impact (European Commission, 2022).

2.3 Trade-off Challenges

Optimizing technical and environmental sustainability independently leads to inefficiencies (FPP, §2.3.1). For instance, applying object-oriented design patterns improves maintainability but can increase energy usage by over 50% (Feldt et al., 2016). Conversely, optimizing for energy

efficiency may complicate code and hinder evolvability. Academic research has been very active over the past years and has resulted in a number of Systematic Mapping Studies (SMS) and Systematic Literature Reviews (SLR) adjacent to the topic.

Garcia-Mireles et al.[12] conducted a SMS on software product quality and environmental sustainability goals. In their study, they discuss relevant research approaches used by researchers to delve into the topic. Researchers are mostly exploring how current software development technologies impact energy consumption. In their SMS they state there is a need for practices to develop suitable software without restricting quality attributes. They also state there is a need to study the relationship between maintainability and sustainability at runtime and design.

Andrikopoulos et al.[2] conducted another SMS on sustainability in software architecture. A vast amount of papers under the SMS' findings touch on the technical dimension of sustainable software development, pointing out it is a key area of research. Maintainability and evolution is considered the most popular individually discussed phase within sustainable software development. However, the authors point out there is an urgent need for approaches that incorporate less commonly addressed dimensions such as environmental sustainability.

Poy et al.[25] also conducted a SMS; they attempt to investigate the impact of different software development techniques on energy consumption reported by different researchers. Structural patterns tend to have no impact on energy consumption, whilst behavioral patterns often have a negative impact. Removing code smells was proven to improve energy efficiency, however other refactoring techniques were inconsistent. However, they do not explore the relationship between maintainability and energy consumption in this SMS. This is key in understanding whether technical sustainability is hindering environmental sustainability.

Pinto et al.[24] go more in depth into refactoring for energy efficiency, by conducting a review on the state of the art. Under their review, they point out refactoring techniques which could improve energy consumption, as long as their challenges are addressed. The main focus of the paper is refactoring for energy efficiency in mobile applications. The authors urge developers to consider

energy efficient APIs and to determine whether CPU offloading is necessary. They also explore parallel programming and highlight how not all programs are parallelizable.

Lastly, Connolly et al.[7] SLR on how software design influences energy performance. While conducting their SLR, they encountered a great number of contradictory findings. They attribute these findings, particularly in design pattern studies, to the lack of statistical testing or due to an underlying experimental flaw. Even greater contradictions were found in studies addressing code smells. The authors claim this is due to the use of different techniques to solve the same code smell. This highlights the need for researchers to delve deeper into the code smells. Similar highlights were raised about refactorings, stating that labelling any refactoring as green should be dealt with extreme caution, and suggesting to take into account the context beforehand.

In summary, as Connolly pointed out, little has been done terms of concrete results. Papers are focusing on how software techniques influence energy consumption, however no papers address the impact these may have on the technical sustainability of the software.

Current tools support **individual quality attributes** (TD management or energy profiling) but lack frameworks for **multi-objective trade-off analysis** (Ampatzoglou et al., 2018). Research in multi-criteria optimization and predictive modeling (e.g., UPPAAL (Larsen et al. (1997)) for response time vs. energy trade-offs) shows potential but remains experimental (Beek et al., 2020). MAST aims to close this gap by introducing **decision-support systems and visualization dashboards** for real-time trade-off analysis (FPP, §2.3.2).

2.4 Lessons from Prior Research Projects

European R&D initiatives have advanced sustainability practices:

- **SDK4ED (H2020):** Managed TD and energy efficiency in embedded systems but lacked integrated trade-off tools for mainstream DevOps.
- **VISDOM (ITEA):** Improved visualization for complex software but did not address sustainability metrics.

- **SustainableCloud:** Explored sustainable cloud resource allocation without linking to developer workflows.
- **FUSE-IT, SEAS (ITEA):** Targeted energy optimization in smart buildings/components but not software-level sustainability analytics.

These projects demonstrate methods for energy measurement and TD tracking but **lack cross-domain integration and actionable decision support**. MAST extends these by combining **AI-based optimization**, **human-guided program repair**, and **comprehensive dashboards** that jointly address technical and environmental sustainability (FPP, §2.3.3–2.3.4).

3 Current Industrial Practice across the Consortium

The MAST consortium brings together a diverse set of industrial and academic partners operating across multiple countries and domains, including software development, IoT/edge/cloud platforms, manufacturing systems, and energy management solutions. Despite differences in focus areas, there are common patterns in how sustainability is currently managed:

3.1 Technical Sustainability Practices

Across the MAST consortium, industrial partners demonstrate mature software engineering processes centered around **modularity, quality assurance, and agile DevOps workflows**.

- **Code Quality and Technical Debt Management:**
  Tools such as **SonarQube** and **static analysis frameworks** are widely used to detect code duplication, cyclomatic complexity, and architectural smells. However, as noted in the FPP (§2.3.1), **technical debt management remains reactive**, often addressed during refactoring sprints rather than proactively monitored throughout the lifecycle. Studies have shown that deferred debt repayment leads to long-term maintainability challenges and increased costs (Ampatzoglou et al., 2016).
- **Architecture Evolution and Modularity:**
  Industrial software systems, including **Wirtek's Wappsto platform** and **Canon's modular printing systems**, employ microservices and modular design principles for scalability. Yet, **quantitative sustainability metrics** (e.g., maintainability indices, architecture debt scores) are **not systematically linked to architectural decision-making**.
- **CI/CD Pipelines:**
  Continuous Integration and Deployment (CI/CD) practices are well established, supporting rapid releases and automated testing. However, **sustainability checks are absent**, meaning technical debt and energy efficiency are not automatically evaluated or enforced during builds (FPP, §2.3.1).
- **Lack of Trade-off Analytics:**
  No partner currently possesses a tooling framework that explicitly visualizes **trade-offs**

between maintainability and performance or energy efficiency, resulting in engineering decisions being made without quantifiable sustainability insights.

3.2 Environmental Sustainability Practices

While environmental sustainability is increasingly a strategic priority (especially in energy and IoT domains), **practical implementation is fragmented and coarse-grained**:

- **Energy Measurement:**
  Partners like **Cleanwatts** and **Wirtek** monitor energy consumption at **hardware or infrastructure level** using cloud dashboards or energy meters. However, **software-level energy profiling** is not integrated into development workflows. Prior research shows that software-induced energy waste can account for up to 20% of total ICT emissions (Gill & Buyya, 2018), yet remains **largely invisible to developers**.
- **Carbon Reporting:**
  Compliance with directives such as the **Corporate Sustainability Reporting Directive (CSRD)** is emerging. Reporting focuses on **data center energy usage or hardware lifecycles**, leaving **software execution impacts unmeasured**. This creates a gap between regulatory sustainability goals and day-to-day engineering practice (European Commission, 2022).
- **Green Computing Initiatives:**
  Cleanwatts' Living Lab explores renewable energy optimization and flexibility markets, but **software sustainability metrics** (e.g., energy-aware algorithms, software carbon intensity) are not incorporated into platform development. Similarly, **Canon's lifecycle management** emphasizes hardware reuse and modularity but lacks **software-driven energy analytics** for print system control.
- **Operational Optimization:**
  Energy-aware runtime optimizations (e.g., scheduling, power management) are used in embedded and industrial systems. However, these are **ad hoc and domain-specific**, not linked to unified sustainability metrics or cross-domain decision-support systems.

3.3 Cross-Cutting Gaps

Combining insights from the FPP and partner practices, the following **systemic gaps** emerge:

1. **Fragmented Metrics:**
   Technical debt, maintainability, and energy consumption are measured in isolation. There is **no integrated sustainability dashboard** unifying these dimensions for decision-making (SDK4ED identified this as a limitation).
2. **Absence of Feedback Loops:**
   CI/CD pipelines lack mechanisms to **continuously assess sustainability impacts**, meaning developers have **no real-time guidance** on improving code sustainability (Ampatzoglou et al., 2018).
3. **Limited Predictive Capabilities:**
   Forecasting models for **technical debt evolution** or **energy consumption trends** are not used, making it difficult to plan sustainable software evolution (Mens et al., 2014).
4. **Lack of Trade-off Tools:**
   Partners cannot visualize **Pareto-optimal solutions** balancing performance, maintainability, and environmental footprint, leading to uninformed design choices (Beek et al., 2020).

3.4 Opportunity for MAST

MAST addresses these gaps by introducing:

- **Integrated Sustainability Metrics:** Combining technical and environmental indicators into a **unified dashboard** accessible to developers, architects, and operators.
- **Continuous Feedback in DevOps:** Embedding sustainability analysis into CI/CD pipelines, enabling automated checks and proactive recommendations.
- **AI-driven Trade-off Analysis:** Providing decision-support tools for navigating complex trade-offs in software design and runtime configurations.

- **Cross-Domain Validation:** Pilots in cloud software (Wirtek), renewable energy optimization (Cleanwatts), and modular printing systems (Canon Production Printing) ensure broad applicability.

By embedding sustainability as a **first-class engineering concern**, MAST will enable **data-driven, proactive, and cost-effective sustainability management**, bridging the gap between current industrial practice and the state of the art.

4 Use Case–Specific State of Practice and Gaps

4.1 Wirtek – UC-WRT: Clean Code to Green Code

4.1.1 Overview

Wirtek's use case, described in **D2.1** and **FPP §2.3.3**, focuses on embedding **sustainability trade-offs** into the software development lifecycle of its **Wappsto IoT platform**. The platform enables IoT device connectivity and data management for smart energy and automation solutions.

In the context of MAST, Wirtek aims to enhance its software engineering process by **measuring and visualizing sustainability indicators**, enabling developers to make informed decisions that improve **maintainability, energy efficiency, and overall software sustainability**. This aligns with Wirtek's broader commitment to **clean code practices** and **ESG principles**.

4.1.2 Current Practice

- **Software Development Practice:** Wirtek employs modern agile methodologies, continuous integration/continuous deployment (CI/CD), and automated testing frameworks. Code quality is tracked using tools such as SonarQube, focusing on maintainability, modularity, and test coverage.
- **Technical Debt:** Technical debt is identified during code reviews and refactoring sprints. However, debt metrics are not linked to long-term sustainability outcomes, and no predictive modeling is currently used to forecast debt evolution (as highlighted in FPP §2.3.1 and D2.1).
- **Environmental Sustainability:** Energy consumption and carbon footprint are not directly measured during software development or runtime. Available insights are limited to coarse-

grained cloud infrastructure metrics (e.g., Azure dashboards), which do not map to specific software components or code modules.

- **Tooling and Workflows:** While Wirtek's DevOps pipelines support code quality and automated deployments, sustainability checks (e.g., energy profiling, trade-off analytics) are not integrated into existing workflows.
- **Existing Strengths:**
    - Established modular architecture for IoT solutions.
    - Strong emphasis on testability and maintainability.
    - Early awareness of ESG goals among internal teams and clients, particularly in energy and IoT domains.

4.1.3 Relevant State-of-the-Art Concepts

- **Technical Debt and Maintainability Metrics:** Research on managing technical debt (Avgeriou et al., 2016; Ampatzoglou et al., 2018) highlights systematic methods for measuring and forecasting long-term maintainability costs. Tools such as SonarQube and Designite partially address these needs but lack integration with sustainability analytics.
- **Green Software Engineering:** Emerging practices focus on reducing energy consumption during software execution (Murugesan, 2008; Pereira et al., 2017). Fine-grained energy profiling tools (GreenSource, JoularJX) exist but are rarely used in industrial IoT settings.
- **Trade-off Analysis and Multi-objective Optimization:** State-of-the-art methods (Deb, 2014; Van Beek et al., 2020) support balancing conflicting quality attributes (e.g., performance vs. energy efficiency). However, they remain largely experimental and are not integrated into mainstream DevOps pipelines.
- **Continuous Sustainability Feedback:** Research advocates embedding sustainability checks into CI/CD processes (Schmidt et al., 2021), enabling developers to continuously monitor maintainability and energy impacts—a practice not yet adopted by Wirtek.

4.1.4 Identified Gaps

From the analysis in **D2.1 and FPP §2.3.3**, the following gaps are identified:

- **Lack of sustainability metrics:** No software-level energy profiling or carbon footprint analysis is available to developers.
- **Limited integration of technical and environmental factors:** Maintainability is addressed qualitatively but not linked to measurable environmental impact (Gill & Buyya, 2018).
- **Absence of trade-off decision support:** Developers cannot visualize or balance performance, maintainability, and environmental sustainability objectives (Van Beek et al., 2020).
- **Manual, reactive technical debt management:** No predictive or automated tools to anticipate future maintainability challenges (Izurieta et al., 2017).

4.1.5 Goals within MAST

Through the MAST project, we plan to:

- **Develop Telemetry and Monitoring Frameworks:** Instrument its software to capture sustainability-related signals (CPU cycles, memory usage, resource allocation) and map them to carbon footprint estimations.
- **Integrate Sustainability Dashboards:** Embed sustainability analytics into DevOps pipelines, providing **real-time feedback** during builds and deployments (Schmidt et al., 2021).
- **Implement Trade-off Analysis Tools:** Utilize multi-objective optimization techniques (Deb, 2014) to guide engineering decisions that balance maintainability, energy efficiency, and time-to-market.
- **Pilot in a Real Project:** Apply these solutions in a live IoT development project on the Wappsto platform to validate feasibility, measure improvements, and gather developer feedback.

4.2 Cleanwatts – UC-CLW: Energy Community Management

4.2.1 Overview

Cleanwatts operates the **CleanwattsOS platform**, a comprehensive solution for managing **Renewable Energy Communities (RECs)** and enabling collective self-consumption, peer-to-peer trading, and flexibility services. Within MAST (D2.1, FPP §2.3.3), Cleanwatts' use case focuses on integrating **sustainability-aware decision support** into energy management, enabling optimized balancing of **cost, CO_2 reduction, and self-sufficiency** within energy communities.

4.2.2 Current Practice

- **Energy Community Management:**
  CleanwattsOS aggregates data from solar PV, batteries, EV chargers, and smart meters, providing **real-time monitoring** and control of distributed energy resources.
- **Forecasting and Flexibility:**
  Current models handle demand response and renewable generation forecasting. While functional, they have **limited robustness** under high variability of weather and consumption patterns.
- **Optimization Approach:**
  Community optimization currently focuses on **economic benefits**, prioritizing cost savings for members.
  Sustainability metrics such as **CO_2 avoided or renewable share maximization** are only partially incorporated into optimization algorithms.
- **User Interaction:**
  The platform's dashboards allow users to track consumption and savings but **do not visualize sustainability trade-offs** (e.g., choosing between maximizing self-consumption vs. minimizing carbon emissions).
- **Regulatory Context:**
  Cleanwatts operates within the EU energy market frameworks for collective self-

consumption and flexibility, but current regulatory barriers **limit the full deployment of advanced flexibility mechanisms**.

4.2.3 Relevant State-of-the-Art Concepts

- **Peer-to-Peer Energy Trading and Flexibility Markets:** European R&D projects (e.g., FUSE-IT, SEAS) have explored decentralized energy management but lacked integrated sustainability analytics (Gill & Buyya, 2018).
- **AI-Based Forecasting:** Research shows machine learning improves accuracy in renewable energy forecasting and demand response (Van Beek et al., 2020), a promising approach for Cleanwatts' optimization models.
- **Multi-objective Optimization:** State-of-the-art algorithms enable balancing economic, technical, and environmental KPIs (Deb, 2014), but these are rarely deployed in operational energy community platforms.
- **Decision Support and Visualization:** Sustainability dashboards for grid operators and community managers (Fernández-Sánchez & Sepúlveda-Escribano, 2020) have shown potential but remain experimental in commercial platforms.

4.2.4 Identified Gaps

From **D2.1 and FPP §2.3.3**, the following gaps are identified:

- **Lack of comprehensive sustainability metrics:** Current optimization algorithms are primarily cost-driven and do not simultaneously evaluate CO_2 reduction potential or long-term sustainability impacts.
- **Limited decision-support for end users:** Operators and community members lack intuitive dashboards to understand and act upon sustainability trade-offs.
- **Forecasting limitations:** Existing models need improved accuracy and resilience against variability in renewable generation and load patterns.
- **Regulatory constraints:** Advanced flexibility mechanisms and dynamic tariffs, which could enhance sustainability, face market and policy barriers.

4.2.5 Goals within MAST

Through the MAST project, we plan to:

- **Enhance AI-based forecasting:** Improve prediction accuracy for solar generation, consumption, and flexibility opportunities using state-of-the-art machine learning models.
- **Integrate multi-dimensional sustainability metrics:** Incorporate CO_2 reduction and renewable penetration alongside economic objectives in optimization algorithms.
- **Develop decision-support dashboards:** Provide operators and end-users with visual tools that clearly present sustainability trade-offs and recommend actions (Fernández-Sánchez & Sepúlveda-Escribano, 2020).
- **Pilot advanced optimization modules:** Validate these innovations in the **Cleanwatts Living Lab**, ensuring real-world effectiveness and scalability across European RECs.

4.3 Canon – UC-CPP: Sustainable System Control

4.3.1 Overview

Canon Production Printing (CPP) develops **industrial digital printing systems** that serve commercial and high-volume printing markets. Currently, major system upgrades often involve **hardware replacements**, contributing to increased **material waste, carbon footprint, and lifecycle costs**.

Within MAST (D2.1, FPP §2.3.3), Canon's use case aims to transition toward **long-lifetime modular products** enabled by **software-driven supervisory control** and **eco-feedback mechanisms**. The goal is to develop tools that allow R&D teams, regional organizational teams, and operators to **evaluate sustainability impacts early** and **dynamically optimize** energy usage and waste reduction during machine operation.

4.3.2 Current Practice

- **Product Lifecycle and Modularity:**
  Canon has introduced modular and remanufactured hardware to enable selective replacement

during product evolution. However, supervisory software **does not fully leverage modularity** for sustainability optimization.

- **Supervisory Control Systems:**
  Existing control software focuses primarily on **machine reliability, throughput, and print quality**. Real-time **energy efficiency optimization** is limited, leading to higher operational energy consumption dependent on print room conditions and operator behaviour.
- **Diagnostics and Maintenance:**
  Predictive maintenance capabilities are implemented but primarily used for **fault detection and uptime assurance**. Sustainability indicators (e.g., component energy profiles, material usage) are **not part of diagnostics**.
- **Customer Dashboards:**
  End-user interfaces and customer instruction provide basic operational statistics and best practices but **lack actionable sustainability insights**, such as eco-modes or recommendations for reducing energy and material waste during print runs.
- **Upgrade Path:**
  Major system upgrades typically require **physical component replacements** or full machine substitutions. Software-driven sustainability enhancements are **not yet actively implemented**.

4.3.3 Relevant State-of-the-Art Concepts

- **Modular Product Design and Lifecycle Extension:** Research in circular economy and sustainable product design (Feldt et al., 2016) supports modular architectures that enable incremental upgrades, reducing environmental impact.
- **Predictive and Condition-Based Maintenance:** Advanced analytics using machine learning enable **real-time health monitoring** of components, minimizing waste and improving energy efficiency (Izurieta et al., 2017).
- **Eco-Feedback Systems:** Studies show that providing operators with **real-time sustainability feedback** leads to behavioural changes that lower environmental footprint (Fernández-Sánchez & Sepúlveda-Escribano, 2020).

- **AI-Driven Supervisory Control:** Emerging solutions in industrial automation use **AI-based optimization** to dynamically adjust machine operations for energy savings without sacrificing throughput (Van Beek et al., 2020).

4.3.4 Identified Gaps

From **D2.1 and FPP §2.3.3**, key sustainability challenges are:

- **Lack of sustainability metrics in R&D:** Current product design tools **do not assess environmental impact** during early stages of system development.
- **Limited energy optimization in control software:** Supervisory control focuses on quality and productivity, with **no algorithms for dynamic power management**.
- **Dependence on hardware replacements:** Sustainability improvements typically require **physical component changes**, causing additional material & logistics-related waste.
- **Absence of eco-feedback dashboards:** Operators and customers **cannot access actionable sustainability guidance** during machine operation.

4.3.5 Goals within MAST

Through the MAST project, we plan to:

- **Develop sustainability impact evaluation tools:** Provide R&D teams with early-stage design decision support for environmental assessments.
- **Enhance supervisory control systems:** Integrate **AI-based optimization** for real-time energy and resource efficiency during printing operations (Van Beek et al., 2020).
- **Enable software-driven and modular hardware upgrades:** Shift from full hardware replacement upgrades to **module and software-based sustainability improvements**, extending product lifespan and reducing material waste.
- **Deploy eco-feedback dashboards:** Create customer-facing interfaces that visualize **energy consumption, material usage, and recommended eco-modes**, enabling operators to adjust behaviour and minimize environmental impact.

These advancements will be piloted within **Canon Production Printing's product development environment**, demonstrating feasibility and setting the foundation for **sustainable commercial printing solutions** across global markets.

5 Innovation and Technological Advances

5.1 Positioning MAST within the Research Landscape

Prior research projects have advanced software sustainability but show **three persistent gaps**:

1. **Fragmentation of Sustainability Metrics** – Projects like **SDK4ED** introduced technical debt and energy optimization modules but lacked holistic trade-off analysis (Ampatzoglou et al., 2018).
2. **Limited Integration with Industrial Workflows** – Efforts such as VISDOM improved visualization of software complexity but did not embed sustainability metrics into DevOps pipelines (Fernández-Sánchez & Sepúlveda-Escribano, 2020).”
3. **Narrow Scope of Optimization** – Initiatives like **SustainableCloud** focused on infrastructure-level energy efficiency without linking software-level decision support to long-term maintainability (Gill & Buyya, 2018).

5.2 Technological Innovations Introduced by MAST

MAST builds upon and extends these prior efforts with the following **novel contributions**:

5.2.1 Integrated Sustainability Trade-off Analysis

MAST introduces a unified framework that combines:

- Technical debt analysis,
- Energy consumption telemetry,
- AI-based trade-off modeling and visualization dashboards.

Unlike SDK4ED’s separate modules, MAST’s approach **provides real-time, multi-objective sustainability analysis** (Beek et al., 2020).

5.2.2 Human-Guided Program Repair and Evolution

Existing program repair techniques rely heavily on automated static fixes, often lacking transparency (Monperrus, 2018).

MAST incorporates **explainable AI models**, enabling developers to understand sustainability recommendations and **interactively guide program evolution**, improving adoption likelihood.

5.2.3 Sustainability-Aware DevOps Tooling

Building on lessons from SDK4ED and VISDOM, MAST is designed for **seamless integration** into CI/CD systems.

- Plugins automatically capture sustainability metrics,
- Feedback loops trigger recommendations during builds,
- The modular architecture ensures low adoption barriers.

This approach aligns with research calling for **continuous sustainability monitoring** in DevOps pipelines (Schmidt et al., 2021).

5.2.4 Cross-Domain Applicability and Real-World Pilots

Prior initiatives were domain-specific (e.g., SDK4ED focused on embedded systems).
MAST validates its solutions across **three diverse industrial contexts**:

- Cloud-based IoT software (Wirtek),
- Renewable energy optimization (Cleanwatts),
- Modular industrial printing (Canon Production Printing).

This ensures **broader applicability and impact** (FPP §2.3.3).

5.3 Conceptual Advances Beyond State-of-the-Art

MAST's conceptual advances include:

- **Trade-off Dashboards:** Inspired by Pareto-optimization research (Deb, 2014), MAST delivers dashboards that visualize sustainability trade-offs.
- **Multi-objective Optimization:** AI-based solvers compute design recommendations balancing maintainability and energy efficiency (Van Beek et al., 2020).
- **Closed-Loop Self-Assessment:** Demonstrated in Wirtek's pilot, MAST introduces **self-sustaining platforms** that measure and optimize their own sustainability performance, a novel approach in digital ecosystems.

5.4 Anticipated Impact

By addressing the limitations of prior projects, MAST is expected to:

- Embed sustainability as a **core engineering metric**,
- Enable **data-driven reduction of ICT carbon emissions**,
- Improve long-term maintainability and competitiveness,
- Support **EU Green Deal targets** and future sustainability reporting standards (European Commission, 2022).

6 Expected Outputs and Impact

6.1 Overview

The MAST project is designed to produce a **comprehensive suite of methodologies, tools, and pilot demonstrations** that operationalize sustainability as a first-class concern in software engineering and system design. Building on the **baseline analysis of current practices (Section 3)** and **state-of-the-art research (Section 2)**, MAST will deliver tangible innovations that bridge the gap between academic knowledge and industrial application.

6.2 Methodological Outputs

MAST will advance sustainability-aware software engineering by developing:

- **Unified Sustainability Metrics:** A common framework that integrates **technical debt indices**, **maintainability metrics**, and **energy consumption indicators** into a holistic sustainability score. This framework will be rooted in established models (Ampatzoglou et al., 2016; Murugesan, 2008) and validated through empirical data from consortium partners.
- **Trade-off Analysis Methodology:** A novel multi-objective optimization approach enabling stakeholders to navigate and visualize **Pareto-optimal solutions** that balance performance, maintainability, and environmental impact.
- **Human-Guided Program Repair Techniques:** A methodology for **interactive, explainable program repair**, providing developers with actionable insights to improve sustainability metrics without sacrificing code quality or performance.

6.3 Technological Outputs

MAST's technological innovations include:

- **Sustainability Dashboards:** Developer-facing dashboards integrated into CI/CD pipelines to provide **real-time sustainability feedback** during software development and deployment.
- **AI-Based Optimization Tools:** Advanced analytics for **predictive technical debt evolution**, **energy usage forecasting**, and automated refactoring suggestions.
- **Cross-Domain DevOps Plugins:** Modular components compatible with mainstream industrial DevOps environments, ensuring **low adoption barriers** and **scalability** across domains.

6.4 Pilot Demonstrations

Three industrial pilots will validate MAST's innovations:

- **Wirtek:** Integration of sustainability dashboards into IoT platform development, demonstrating technical debt reduction and energy efficiency improvements.
- **Cleanwatts:** Deployment of trade-off analysis tools in renewable energy community management, balancing economic, technical, and environmental sustainability goals.
- **Canon Production Printing:** Application of sustainability-aware supervisory control systems in modular printing architectures, extending product lifecycles and optimizing energy usage.

These pilots will provide **evidence-based validation**, ensuring that developed solutions are **practical, scalable, and impactful** across heterogeneous industries.

6.5 Project-Level Impact

The outcomes of MAST are expected to:

- Establish **sustainability-aware development practices** as standard in software and system engineering lifecycles.

- Enable measurable reductions in **technical debt accumulation** and **ICT carbon emissions**, contributing to EU Green Deal objectives (European Commission, 2022).
- Improve **long-term maintainability and cost-efficiency** of complex software systems, enhancing industrial competitiveness.
- Provide a **foundation for future standardization efforts** in sustainability metrics and DevOps integration.

6.6 Role of this Deliverable

This D2.2 deliverable directly supports these outputs by:

- **Documenting the baseline** from which improvements will be measured.
- Identifying **gaps and challenges** that guide the development of methodologies and tools in WP3 (Concept Development).
- Informing **WP4 (Tooling and Integration)** by specifying the sustainability dimensions and metrics to be operationalized.
- Preparing for **WP5 (Pilots and Validation)** by detailing current industrial practices and the context of each pilot.

7 Conclusions and Next Steps

This deliverable has established a **comprehensive baseline** for sustainability in software and system engineering, combining insights from **academic research**, **prior European projects**, and **current industrial practices** within the MAST consortium.

Key findings include:

- Sustainability is **not yet a first-class engineering concern** in most industrial settings; metrics for technical debt, maintainability, and energy efficiency are **fragmented and unintegrated**.
- Prior research projects introduced important building blocks (e.g., SDK4ED's technical debt tools, VISDOM's visualization techniques) but lacked **cross-domain integration, actionable feedback, and DevOps adoption pathways**.
- Consortium partners possess **mature development processes** but lack **real-time sustainability feedback mechanisms**, **predictive modeling capabilities**, and **decision-support tools for trade-offs** between technical and environmental sustainability.

7.1 Role of MAST Innovations

MAST is uniquely positioned to **address these gaps** by:

- Developing **unified sustainability metrics** combining technical and environmental dimensions.
- Introducing **AI-powered trade-off analysis and human-guided program repair** to support sustainable decision-making.
- Embedding sustainability analysis directly into **mainstream CI/CD pipelines**, ensuring proactive feedback for developers and operators.
- Validating innovations through **three industrial pilots** spanning cloud-based IoT platforms (Wirtek), renewable energy communities (Cleanwatts), and modular industrial printing systems (Canon).

7.2 Next Steps in the Project Lifecycle

The results of this deliverable will **directly inform upcoming work packages**:

- **WP3 – Concept Development:**
  Use findings to define the conceptual architecture for sustainability-aware development tools and methodologies.
- **WP4 – Tooling and Integration:**
  Implement sustainability dashboards, DevOps plugins, and optimization modules based on the metrics and gaps identified herein.
- **WP5 – Pilots and Validation:**
  Deploy and evaluate developed solutions in industrial contexts, demonstrating measurable improvements in sustainability performance.

Furthermore, D2.2 will serve as a **reference point for impact assessment**, enabling MAST to track sustainability improvements against the **baseline documented in this report**.

7.3 Broader Impact

By addressing systemic sustainability challenges, MAST aims to:

- Establish **sustainability as a measurable and actionable engineering attribute**, embedded throughout the software lifecycle.
- Reduce the **carbon footprint of ICT systems** while enhancing maintainability and long-term value creation.
- Contribute to **European digital sustainability goals**, including the **Green Deal** and the **Corporate Sustainability Reporting Directive (CSRD)**.

7.4 Future Directions

Beyond the project's lifetime, the outputs of MAST are expected to **influence standardization efforts**, industrial best practices, and future academic research in sustainable software engineering. This deliverable thus represents not only a **baseline assessment** but also the **foundation for lasting**

transformation in how sustainability is conceived and operationalized in software and system design.

Bibliography

- Ampatzoglou, A., Avgeriou, P., & Chatzigeorgiou, A. (2016). The financial aspect of managing technical debt: A systematic literature review. *IEEE Transactions on Software Engineering, 43*(8), 659–678.
- Ampatzoglou, A., et al. (2018). Technical debt forecasting: A systematic literature review. *Information and Software Technology, 98*, 35–61.
- Avgeriou, P., et al. (2016). Managing technical debt in software engineering. *ACM Computing Surveys, 49*(4), 1–34.
- Vasilios Andrikopoulos et al. "Sustainability in software architecture: a systematic mapping study". In: *2022 48th Euromicro Conference on Software Software Engineering and Advanced Applications (SEAA)*. IEEE. 2022, pp. 426– 433.
- Beek, D. A. van, et al. (2020). Trade-off analysis in embedded systems design: A model-checking approach. *Journal of Systems and Software, 162*, 110509.
- Caron, E., et al. (2022). Carbon footprint of cloud computing: Measurement challenges and perspectives. *Future Generation Computer Systems, 133*, 145–158.
- Deaglan Connolly Bree and Mel ´O Cinneide. "How Software Design Affects Energy Performance: A Systematic Literature Review". In: *Journal of Software: Evolution and Process* 37.4 (2025).
- Deb, K. (2014). Multi-objective optimization using evolutionary algorithms. *Wiley*.
- European Commission. (2022). *Corporate Sustainability Due Diligence Directive*.
- Feldt, R., et al. (2016). Impact of software design patterns on energy consumption. *Empirical Software Engineering, 21*(3), 1–25.
- Fernández-Sánchez, C., & Sepúlveda-Escribano, J. (2020). *Software visualization techniques for sustainability metrics: A systematic review. Empirical Software Engineering, 25(2), 1234–1258.*

- Gabriel Alberto Garcia-Mireles et al. “Interactions between environmental sustainability goals and software product quality: A mapping study”. In: *Information and Software Technology* 95 (2018), pp. 108–129.
- Gill, S., & Buyya, R. (2018). Sustainable cloud computing: Foundations and future directions. *Journal of Cloud Computing, 7*(1), 1–29.
- Izurieta, C., et al. (2017). Organizing Technical Debt Prioritization: A Value-Based Approach. Journal of Systems and Software, 137, 188–205.
- Koziolek, H. (2011). Sustainability evaluation of software architectures. *Proceedings of the Joint ACM SIGSOFT Conference*.
- Larsen, K. G., Pettersson, P., & Yi, W. (1997). *UPPAAL in a nutshell. International Journal on Software Tools for Technology Transfer, 1*(1–2), 134–152.
- Lehman, M. (1996). Laws of software evolution revisited. *Proceedings of the European Workshop on Software Process Technology*.
- Mens, T., et al. (2014). The ecology of software evolution. *Science of Computer Programming, 98*(2), 70–88.
- Monperrus, M. (2018). Automatic software repair: A bibliography. ACM Computing Surveys, 51(1), 1–24.
- Murugesan, S. (2008). Harnessing green IT: Principles and practices. *IEEE IT Professional, 10*(1), 24–33.
- Pereira, R., et al. (2017). Software engineering for sustainability: Energy efficiency and green IT. *IEEE Software, 34*(6), 56–61.
- Gustavo Pinto, Francisco Soares-Neto, and Fernando Castor. “Refactoring for energy efficiency: A reflection on the state of the art”. In: *2015 IEEE/ACM 4th International Workshop on Green and Sustainable Software*.
- IEEE. 2015, pp. 29–35.
- Potdar, A., & Shihab, E. (2014). An exploratory study on self-admitted technical debt. *IEEE ICSME*, 91–100.

- Olivia Poy et al. “Impact on energy consumption of design patterns, code smells and refactoring techniques: A systematic mapping study”. In: *Journal of Systems and Software* (2024), p. 112303.
- Schmidt, R., et al. (2021). Continuous sustainability assessment in DevOps. *IEEE Software, 38*(6), 45–53.