Are Information Systems Sustainability Requirements SMART Enough?

Christophe Ponsard[1], Renaud De Landtsheer[1], Abiola Paterne Chokki[1] and Mounir Touzani[2]

[1]CETIC Research Centre, Gosselies, Belgium
[2]INRAE, Toulouse, France

Abstract

This paper examines the maturity of sustainability requirements in Information Systems (IS) through the lens of the SMART criteria (Specific, Measurable, Achievable, Relevant, Time-bound). While sustainability is increasingly recognized as a key concern in IS, its definition remains ambiguous and multifaceted, spanning technical, environmental, social, and economic dimensions. As a result, sustainability requirements are often inconsistently formulated and difficult to operationalise. We provide an overview of the main strengths and weaknesses of current approaches and frameworks, illustrated through case studies carried out with companies. Based on this analysis, we identify the main gaps and propose improvements regarding sustainability requirements classification, grounding, coupling with modernisation, and the impact of artificial intelligence tools.

Keywords

Sustainable Software Engineering, Requirements Engineering, Information Systems, Frugality, Metrics

1. Introduction and Context

Information Systems (IS) are "formal, sociotechnical, and organisational systems designed to collect, process, store, and distribute information" [1]. They form the backbone underpinning daily life and organisations across all sectors such as healthcare, transportation, finance, and public administration. Their influence is pervasive, shaping social, economic, and environmental outcomes through the decisions, processes, and behaviours they enable. By contrast, **Information Technology (IT)** refers to the tools and infrastructure for processing data. IS encompasses IT alongside people, processes, and organisational context to turn data into meaningful information and support decisions.

Given this extensive reach, there is an increasing need to consider sustainability in the design, deployment, and operation of IS. The **sustainability of software systems** covers multiple dimensions, but lacks a precise and widely adopted definition. The Karlskrona Manifesto defines it as the "capacity to endure" [2] and complements it with a framework covering social, environmental, technical, personal, and economic aspects. The GREENSOFT definition emphasises impacts across similar dimensions: "software whose development, deployment, and usage result in minimal direct and indirect negative impacts or even positive impacts on the economy, society, human beings, and the environment" [3]. It can also be defined in terms of software quality attributes such as maintainability and scalability, or broader socio-technical dimensions. The concept is often described as multi-faceted, encompassing both the software product itself and its development process. A key debate concerns whether sustainability should be treated as a non-functional requirement (NFR) [4] or as a broader system-level property [5, 6].

However, sustainability requirements for IS must ultimately be formulated, either directly from NFRs or with the support of sustainability frameworks such as SuSaF [7] or available catalogues [8, 9]. We illustrate this with selected requirements from a night-shift scheduler that we designed [10]:

- R1 - Workload Balance: Ensure that, over any rolling 3-month period, the difference in total assigned duty hours between any two doctors does not exceed 10% of the team average.

RCIS 2026: Companion Proceedings of the 20th Conference on Research Challenges in Information Science: RCIS Research Projects and Workshops, May 26-29, 2026, Toulouse, France

christophe.ponsard@cetic.be (C. Ponsard); renaud.delandtsheer@cetic.be (R. D. Landtsheer); abiola.chokki@cetic.be (A. P. Chokki); mounir.touzani@inrae.fr (M. Touzani)
0000-0002-5027-2114 (C. Ponsard); 0000-0002-5503-9442 (R. D. Landtsheer); 0000-0003-4500-2141 (A. P. Chokki)

- R2 - Computation Approach: the system shall produce solutions within acceptable computation time and resource usage.
- R3 - Efficient Update of Rules: the system shall allow scheduling constraints (i.e. legal rules, preferences, fairness criteria) to be updated with minimal impact on other components, thereby facilitating long-term evolution of the rostering logic.

These three requirements differ in nature: R1 is about fairness of the result, R2 concerns the algorithmic implementation approach, which affects both performance and energy consumption, and R3 is about the maintenance process. Their quality is also different: while R1 is precise and measurable, R2 is high-level and its fulfilment cannot be evaluated. R3 lacks precision about what minimal impact means (e.g. in architectural terms or development effort). The quality of requirements can be assessed using the SMART criteria, initially defined for management objectives [11], but also applied to software and systems requirements [12]. Among the variants, we consider that a Requirement should be:

- **S - Specific:** clearly defining what is to be achieved, leaving no ambiguity.
- **M - Measurable:** with concrete criteria or metrics to evaluate completion or to track progress.
- **A - Achievable:** realistic given available resources, technology, and constraints.
- **R - Relevant:** aligned with overall objectives, stakeholder needs, or strategic goals.
- **T - Time-bound:** with a defined deadline or timeframe for completion. Note that this letter is sometimes used for *Traceable*, but we cover this aspect under *Relevant*.

Our ongoing work and proposed workshop contribution are to look at the maturity of sustainability requirements for IS through the following research question: "Are Information Systems Sustainability Requirements SMART Enough?". Our approach first explores (Section 2) each SMART criterion by examining strengths and weaknesses identified in available sustainability approaches and catalogues, but also drawing on lessons learned from case studies we conducted over the past 10 years covering multiple domains (transportation, health, housing, etc.) [9]. Then, we identify and discuss interesting areas of improvement (Section 3) before presenting our conclusions and research agenda.

2. SMART Analysis of Sustainability Requirements for IS

2.1. Specific

A sustainability requirement must clearly and precisely state what is needed, avoiding vague terms such as "green" or "eco-friendly".

Strengths:

- Sustainability is organised around well-defined dimensions such as environmental, social, economic, technical, and personal. There is some variability but it is consistently present across reference frameworks (Karlskrona, GREENSOFT, SuSaF, SEER) and provides an initial classification structure.
- Frameworks also provide either traceable structuring mechanisms (e.g. through activities contributing to values anchored in a given dimension [13]) or more refined classifications such as technical longevity, social fairness, or environmental energy consumption [8, 9].

Weaknesses:

- Requirements are often expressed as high-level goals, lacking precise scope, stakeholder assignment, or actionable boundaries (e.g. R2 above, or "Reduce environmental impact").
- The difficulty in defining the sustainability of software systems generates ambiguity and confusion [5]. However, this is also inherent to the multi-dimensional and context-dependent nature of sustainability.
- Frameworks only provide partial guidance, and catalogues remain limited in scope.

In our experience, more specific requirements tend to emerge later in the design process. The available frameworks provide guidance, although richer taxonomies would be beneficial. This motivated our work towards defining a catalogue for the fairness dimension [9]. Such refinements arose naturally when implementing scheduling systems to support oncology care, night shifts, and carpooling [10].

2.2. Measurable

The level of achievement of a sustainability requirement must be verifiable through quantitative or qualitative key performance indicators (KPIs), metrics, and success criteria.

Strengths:

- KPIs and metrics have been defined for specific requirements (e.g. for the environmental dimension: energy consumption, carbon footprint). Notably, the GREENSOFT model embeds them as part of the DevOps process [14]. Dimension-level indicators are also available [15].
- Standard quality metrics defined for NFRs in the ISO 25010 standard are also applicable through their mapping to sustainability requirements [16].
- Indicators are also formalised in various frameworks through ontologies or meta-models [13].

Weaknesses:

- There are major gaps in available metrics, e.g. for the individual/social dimensions, and higher-level indicators reflecting code and architectural endurance beyond mere maintainability [6].
- Sustainability metrics are still considered to be at an early stage of maturity, although all technical quality attributes are in practice measurable [17].

Our experience once again draws on the fairness dimension and relies on domain-specific metrics, e.g. to measure how well a schedule is balanced at a given moment or over time, and for tracking adherence to an oncology process metric (Relative Dose Intensity), which is a standard published medical indicator [10]. In our fairness pattern modelling, such indicators are explicitly captured [9].

2.3. Achievable

The sustainability target must be within the limits of current knowledge and available technological resources.

Strengths:

- Existing frameworks demonstrate that sustainability can be operationalised through established engineering practices, such as modular architectures, resource-efficient algorithms, and DevOps optimisation, even if these practices remain immature [3, 6, 18].
- Sustainability has a long-term dimension, making incremental progress achievable within an enterprise architecture roadmap, e.g. progressively improving efficiency as the system scales [19].

Weaknesses:

- The sustainability of a software architecture remains an open research challenge, particularly regarding architectural erosion over time. Moreover, the sustainability of architectures themselves is still underexplored, with few long-lived reference architectures and limited evaluation [6].
- Social, regulatory, and some economic or personal requirements (e.g. fairness, long-term viability) are often acknowledged but not operationalised into the design, making realisation uncertain.
- Practical implementation guidelines are lacking, and many approaches have only been validated in academic settings using simplified examples.
- Sustainability often conflicts with other requirements, and such conflicts can go undetected [20].

Our experience: we defined sustainability requirements for 5 real-world systems, mainly in decision-making systems with a focus on the fairness social dimension. To validate our approach, we also reviewed 11 other real-world systems documented in the literature [9]. Our optimisation team is focusing on the algorithmic feasibility of vehicle routing combining a complete set of requirements about fuel consumption/CO_2 emission, human resource constraints, and required IT resources [21].

2.4. Relevant

The requirement must support the system's purpose, address stakeholder priorities, and contribute to organisational or project objectives.

Strengths:

- Sustainability is widely acknowledged as a key property for all systems and is strongly aligned with global challenges such as climate change and the UN Sustainable Development Goals [22].

- Available frameworks provide sound stakeholder elicitation and traceability from key sustainability goals down to requirements [7, 13].

Weaknesses:

- Practitioners may lack awareness or training across all sustainability dimensions, introducing irrelevant requirements or causing gaps and under-representation (e.g. in social dimension) [13].

Our experience: we focused on less explored dimensions such as social impact through a fairness lens. We identified specific requirement patterns linking IS to higher-level goals, such as transparency through the explanation of rule application and the anticipation of violations [9].

2.5. Time-bound

The requirement must state a clear temporal scope, including a defined deadline, time horizon, or evaluation period against which the expected outcome can be measured.

Strengths:

- Frameworks often capture specific timeframes or categories of effects such as immediate, enabling, and structural ones, to identify milestones and build chains of effects [2, 7].

Weaknesses:

- Sustainability is inherently long-term and emergent, making it difficult to define clear deadlines or evaluation horizons at the requirements stage [5, 7].
- In particular, emergent effects may also be influenced by usage and adoption patterns, such as rebound effects, although several approaches exist to address such effects [3, 7].

Our experience: we specified time constraints to allow a limited violation period for rules in scheduling systems, resulting in a temporary degraded mode (staff overload), or through a rolling time window (e.g. mean load over 3 months). We also documented specific patterns for such temporal constraints [9].

3. Discussion of Areas of Improvement

For space reasons, we only sketch a few key points to stimulate discussion during the workshop:

- Numerous classifications and catalogues exist, often with overlaps and complementarities [4, 8, 9, 16]. These could be better unified, interconnected, and extended with architecture-level requirements such as reconfigurability, resilience, and API openness/compatibility.
- The use of established requirements engineering techniques to reason about conflicts and obstacles is particularly relevant for deriving realistic sustainability requirements. These techniques can complement or enrich existing frameworks, as we have explored for fairness [9].
- For long-lived systems, the sustainability process can be viewed as evolutionary and should be considered in close relation to modernisation approaches, as well as within the broader perspective of enterprise architecture. This affects several phases, including architecture refactoring, service replacement, and infrastructure deployment models.
- Finally, Artificial Intelligence can support various activities: at the problem level, by improving the quality of sustainability requirements, and at the solution level, by assisting with refactoring, code rewriting, etc. However, its own sustainability impact must also be considered in order to determine when and where it provides real value.

4. Conclusion and Perspectives

To sum up, sustainability requirements are only partially SMART: their main strength lies in their relevance, while their main weakness concerns measurability. Analyzing these strengths and weaknesses helped identify key references and promising research directions, which can contribute to a research roadmap based on the topics discussed in Section 3.

Declaration on Generative AI. During the preparation of this work, the authors used ChatGPT to support translation, grammar/spelling check and sentence rewording. After using this tool/service, they reviewed and edited the content as needed and take full responsibility for the publication's content.

Acknowledgments. This publication was partially funded by the Walloon Region CyberExcellence Project (Grant Agreement No. 2110186).

References

[1] G. Piccoli, F. Pigni, Information Systems for Managers: With Cases, 4.0 ed., Prospect Press, 2018.
[2] C. Becker, et al., Sustainability Design and Software: The Karlskrona Manifesto, in: Proc. of the 37th IEEE/ACM Int. Conf. on Software Engineering (ICSE), IEEE/ACM, Florence, Italy, 2015.
[3] S. Naumann, et al., The GREENSOFT Model: A reference model for green and sustainable software and its engineering, Sustainable Computing: Informatics and Systems 1 (2011) 294–304.
[4] A. Raturi, et al., Developing a sustainability non-functional requirements framework, in: Proc. of the 3rd International Workshop on Green and Sustainable Software, GREENS 2014, 2014.
[5] C. C. Venters, L. Lau, M. K. Griffiths, V. Holmes, R. Ward, C. Jay, C. Dibsdale, J. Xu, Software sustainability: The modern tower of babel, ACM Computing Surveys 50 (2017) 1–35.
[6] C. C. Venters, et al., Sustainable software engineering: Reflections on advances in research and practice, Information and Software Technology 164 (2023) 107316.
[7] L. Duboc, et al., Requirements engineering for sustainability: an awareness framework for designing software systems for a better tomorrow, Requir. Eng. 25 (2020).
[8] A. Moreira, J. a. Araújo, C. Gralha, M. Goulão, I. S. Brito, D. Albuquerque, A social and technical sustainability requirements catalogue, Data Knowl. Eng. 143 (2023).
[9] C. Ponsard, B. Nihoul, M. Touzani, Modélisation et classification de patrons d'équité pour la conception de systèmes d'information durables, Revue ouverte d'ingénierie des systèmes d'information 4 (2024). doi:10.21494/iste.op.2024.1141.
[10] C. Ponsard, R. De Landtsheer, Dealing with scheduling fairness in local search: Lessons learned from case studies, in: ICORES (Selected Papers), 2019, pp. 220–243.
[11] G. T. Doran, There's a S.M.A.R.T. way to write management's goals and objectives, Management Review 70 (1981) 35–36.
[12] M. Mannion, B. Keepence, SMART Requirements, ACM SIGSOFT Soft. Eng. Notes 20 (1995) 42–47.
[13] B. Penzenstadler, H. Femmer, A generic model for sustainability with process- and product-specific instances, in: Proc. of the 2013 Workshop on Green in/by Software Engineering, 2013.
[14] M. T. Ailane, C. Rubner, A. Rausch, Green DevOps: A Strategic Framework for Sustainable Software Development, Computer Sciences & Mathematics Forum 10 (2025).
[15] D. H. Meadows, Indicators and Information Systems for Sustainable Development, The Sustainability Institute, 1998.
[16] N. Condori-Fernandez, P. Lago, Characterizing the contribution of quality requirements to software sustainability, Journal of Systems and Software 137 (2018) 289–305.
[17] N. Condori-Fernández, A. Bagnato, E. Kern, A focus group for operationalizing software sustainability with the measure platform, in: MeGSuS@ ESEM, volume 2286, 2018.
[18] C. C. Venters, et al., Software sustainability: Research and practice from a software architecture viewpoint, Journal of Systems and Software 138 (2018).
[19] N. Vandevenne, J. Van Riel, G. Poels, Green Enterprise Architecture (GREAN)—Leveraging EA for Environmentally Sustainable Digital Transformation, Sustainability 15 (2023) 14342.
[20] R. Chitchyan, et al., Sustainability design in requirements engineering: State of practice, in: Proc. 5th International Workshop on Green and Sustainable Software, GREENS, 2016.
[21] R. D. Landtsheer, Green IT for Green Logistics, Blog https://www.cetic.be/4115, 2023.
[22] N. Seyff, et al., Transforming our World through Software: Mapping SuSAF to the UN Sustainable Development Goals, in: Proc. of the 17th International Conference ENASE, 2022.