

Energy Processing Diagrams

Energy Cost Information in Data-Centred Projects

Victor Zudilin
Radboud University
Nijmegen, Netherlands
victor.zudilin@ru.nl

Jordy Aaldering
Radboud University
Nijmegen, Netherlands
jordy.aaldering@ru.nl

Bernard van Gastel
Radboud University
Nijmegen, Netherlands
bernard.vangastel@ru.nl

Mart Lubbers
Radboud University
Nijmegen, Netherlands
mart@cs.ru.nl

Roel Bouman
Radboud University
Nijmegen, Netherlands
roel.bouman@ru.nl

Abstract

Deploying data-centred systems responsibly requires balancing different priorities, such as effectiveness, data privacy and environmental sustainability. Understanding and tracking energy consumption throughout the system architecture is essential for making informed decisions about resource allocation and environmental impact. While Data Processing Diagrams have proven effective for modelling data flows and privacy safeguards, they do not visualise energy consumption. This paper introduces Energy Processing Diagrams, an extension of Data Processing Diagrams, that integrates energy information. These diagrams provide a shared visual and analytical language that helps software engineers design more sustainable software projects in collaboration with stakeholders. They enable teams to track energy flows, compare architectures, identify energy-intensive components and reason about trade-offs. Through a case study, we show how sustainability considerations can be integrated into architectural thinking from the earliest design phases by making energy data visible and actionable.

CCS Concepts

• **Hardware** → *Communication hardware, interfaces and storage*; • **Power and energy**; • **Computer systems organization** → *Architectures*; • **Human-centered computing** → *Visualization systems and tools*; • **Applied computing** → *Education*; • **Social and professional topics** → *Sustainability*.

Keywords

Energy Processing Diagrams, Data Processing Diagrams, Green Software, Sustainability, AI Architecture

ACM Reference Format:

Victor Zudilin, Jordy Aaldering, Bernard van Gastel, Mart Lubbers, and Roel Bouman. 2026. Energy Processing Diagrams: Energy Cost Information in Data-Centred Projects. In *International Conference on Information Technology for Social Good (GoodIT '26)*, September 02–04, 2026, Pisa, Italy. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3794786.3830774>



This work is licensed under a Creative Commons Attribution 4.0 International License. GoodIT '26, Pisa, Italy

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2483-1/26/09
<https://doi.org/10.1145/3794786.3830774>

1 Introduction

As software systems grow in scale and complexity, their environmental footprint shifts from a marginal operational concern to a primary design constraint. However, integrating sustainability into the software development life cycle remains a significant challenge. Recent studies highlight that software developers are often hindered by a lack of empirical knowledge and a deficit of actionable tools to measure or mitigate software energy consumption effectively [18, 23].

While traditional hardware-centric energy metrics exist, they rarely provide software developers with the information needed to understand how specific architectural choices or data flows contribute to the overall footprint of their system. To address these impacts, practitioners often look towards Life Cycle Assessment (LCA) methodologies. Yet, applying LCA to the rapidly evolving domain of software is complicated by the methodological diversity within the environmental sciences. The landscape of assessment has become increasingly fragmented, leading to what has been described as an *alphabet soup* [9] of LCA modes, including attributional (ALCA), consequential (CLCA) and various explorative varieties, such as prospective or scenario-based LCA. This methodological complexity makes it challenging for software engineers to decide between modelling a system as a static snapshot of environmental impacts or as a dynamic system that changes with architectural modifications.

The challenge has become particularly pronounced with the proliferation of Artificial Intelligence (AI) systems, which often rely on energy-intensive computational pipelines involving cloud services and resource-intensive algorithms [2, 15]. Current estimates suggest that the energy use of the Information and Communications Technology (ICT) sector already accounts for approximately 2–3% of global greenhouse gas emissions, which is a figure that is expected to rise significantly as AI integration accelerates [10]. This trend is particularly concerning, given the growing tension between privacy preservation and energy efficiency, where techniques like encryption or decentralised processing, while crucial for data protection, can significantly increase computational overhead and energy consumption [6].

Recent research highlights the need for standardised energy disclosure for AI systems, drawing parallels with energy ratings commonly found on household appliances [17]. Furthermore, studies

show that even seemingly small digital activities, such as email usage, can accumulate significant environmental impacts when scaled across millions of users [20]. This has led to growing concerns about the intended and unintended consequences of energy consumption disclosure on user behaviour and decision-making [13].

1.1 Bridging the Architectural Gap

Traditionally, privacy and environmental sustainability are treated as separate concerns. Data Processing Diagrams (DPDs) were developed as a suitable modelling technique that extends Data Flow Diagrams (DFDs) to provide a standardised, shared language for multidisciplinary teams [6, 25]. By maintaining a high abstraction level, DPDs allow engineers, legal experts and educators to reason about privacy tactics, such as encryption and pseudonymisation, from the earliest design phases. However, a system's energy consumption remains out of scope for DPDs, making it difficult for stakeholders to identify how specific design choices affect a system's energy footprint.

The Green Software Measurement Model (GSMM) provides a comprehensive reference framework for assessing software-induced energy consumption through standardised metrics and technical measurement setups [10]. While the GSMM is effective for detailed technical evaluation, it operates primarily at the component level, lacking integration with high-level architectural models. This creates a critical gap between early-stage design decisions and their environmental implications. Since software decisions directly influence hardware energy use, energy efficiency must be recognised as a design problem, not just an operational one [11].

1.2 Research Contributions and Approach

This paper introduces Energy Processing Diagrams (EnPDs) to bridge the gap between architectural modelling and energy measurement. EnPDs extend the structural modelling of DPDs by integrating energy-related data, as suggested by the GSMM framework. These diagrams function as analytical models, where architectural components are annotated with actual or estimated energy profiles, enabling multidisciplinary teams to visualise energy costs and identify energy *hotspots* within proposed systems.

The development of EnPDs was motivated by the lack of existing tools that effectively bridge the gap between sustainability goals and architectural decision-making. While various approaches address either energy measurement or system modelling in isolation, we found no comprehensive method that integrates both perspectives in a way that supports sustainable design from the earliest stages. EnPDs fill this gap by providing a practical framework for *green by design* approaches, enabling developers to consider environmental impact alongside other system requirements.

The framework is validated through a case study of the mAIInd-map educational platform, where we demonstrate how EnPDs help identify energy hotspots and highlight architectural trade-offs. The rest of the paper thereafter discusses related work, limitations and future directions.

2 Foundational Principles

EnPDs represent a synthesis of two distinct domains: the structural modelling of software architectures and technical assessment of

software-induced energy consumption. By combining the visual clarity of DPDs with the standardised metrics of the GSMM, EnPDs provide a framework for reasoning about sustainability during the architectural design phase. While DPDs excel at modelling privacy safeguards, they lack the vocabulary to express the energy implications of these measures. For instance, end-to-end encryption ensures data confidentiality but increases processing overhead. Similarly, data minimisation techniques reduce privacy risks but may require additional computational steps to implement. EnPDs bridge this gap by allowing software developers and stakeholders to quantify and visualise the energy cost of software system components, enabling informed trade-offs between security and sustainability.

2.1 Architectural Modelling

DPDs provide a foundation for modelling modern, complex data processing systems through several core structural elements. Figure 1 shows how these elements are visualised in EnPDs:

- *processes* (the computational operations that transform or handle data),
- *hotspots* (energy-intensive components, visually distinguished with a dashed red pattern),
- *data stores* (where data is held at rest) and
- *a group of entities* (a logical collection of external actors).

This abstraction is critical for multidisciplinary teams, as it provides a shared language that allows software developers and non-technical stakeholders to identify energy-intensive components and evaluate optimisation strategies, such as computational offloading or model compression, consistently across the system.

A central innovation in DPDs is the use of *context boundaries*, which delineate organisational, legal and physical domains. In the context of EnPDs, these boundaries help visualise where energy consumption occurs within the system architecture. By placing processes and data stores within specific boundaries, software developers and stakeholders can better understand the energy implications of architectural decisions, such as whether to process data locally or in the cloud. This is particularly valuable in distributed AI systems, where computational load and its associated energy costs might be distributed across different parts of the system, helping stakeholders identify which energy-saving opportunities are most actionable within their sphere of influence.

2.2 Measurement Integration

In an EnPD, every process or data store is treated as a potentially measured object with a specific goal. To move beyond high-level overviews, EnPDs adopt some approaches from the GSMM, which provides a structured reference for assessing the resource efficiency of software products. While the GSMM is designed for detailed energy assessments, EnPDs are intended for early-stage architectural design where estimated values are often more practical and sufficient for comparative analysis. The framework remains compatible with empirical data when they become available, allowing for increasing precision as the design progresses.

This model designates energy (measured in Watt-hours [Wh] or Joules [J]) as the primary metric, potentially supplemented by secondary indicators, like runtime (s), hardware utilisation (CPU/GPU,

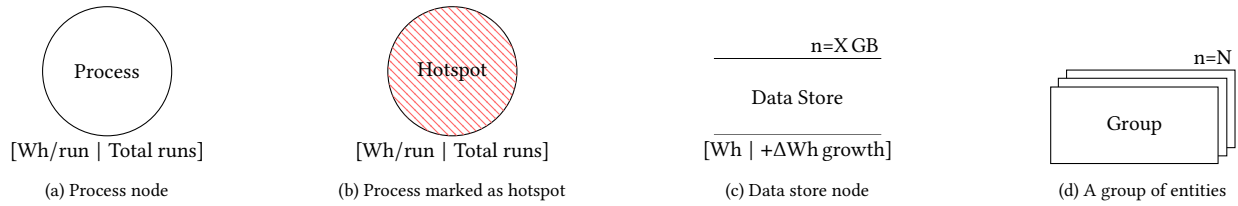


Figure 1: Core elements of the Energy Processing Diagram notation.

in %) and network throughput (MB). By standardising these metrics and units, EnPDs ensure that the annotations within a diagram are comparable across different architectural iterations.

The transition from DPDs to EnPDs requires careful consideration of both architectural design and energy impact assessment. While DPDs provide the structural foundation for displaying data flows between actors, processes and storage units, EnPDs extend this by incorporating energy-related annotations that help reason about the environmental implications of design choices. This integration enables stakeholders to make informed decisions about energy efficiency during the early design phases, when architectural changes are most cost-effective to implement.

3 The Energy Processing Diagram Framework

The EnPD framework extends DPDs by integrating energy data into architectural models. While maintaining DPDs' focus on information flow, EnPDs provide a structured method for evaluating the energy implications of data processing, storage and transfer. This extension allows stakeholders to consider energy efficiency as a first-class constraint during the earliest stages of system design.

A critical scoping decision in the EnPD framework is the exclusion of production and deployment costs, also known as *embodied energy*. Although the manufacturing and transport of hardware represent a significant portion of a system's total environmental footprint, EnPDs are specifically intended to be a tool for those involved in system architecture design. By focusing exclusively on *operational energy* (the consumption driven by software components and data transit) the EnPD framework isolates the variables that system architecture designers can directly influence. This ensures that the model remains a clear reflection of architectural efficiency rather than a hardware life cycle assessment.

To maintain clarity in multi-stakeholder settings and ensure that diagrams remain actionable, EnPDs adhere to the following conventions:

- **Context Boundaries:** In the EnPD framework, the concept of boundaries is retained to define the structural and administrative environments of system components, though they do not explicitly imply energy responsibility or accountability. This allows teams to distinguish between local, cloud and third-party environments without complicating the model with attributional logic.
- **Data Flow Representation:** Real-world network interactions often involve asymmetrical energy costs for uploading and downloading data. However, at the architectural design level, capturing this granularity often introduces additional complexity. EnPDs utilise an aggregated bidirectional metric

for data flows, which can be expressed in two ways. The first notation, [Wh/transfer | Total transfers], is intended for analytical modelling, where identifying the frequency of high-cost transfers is necessary to evaluate optimisation strategies, like batching. Alternatively, the [Wh] notation may be used for high-level summaries, providing a simplified, comparable value representing the total cost of a communication link without detailing the underlying frequency.

- **Process Annotation:** Computational operations are treated as measured objects, with the annotation drawn directly beneath the component. Similar to data flows, designers can choose between two modes. The [Wh/run | Total runs] notation is the preferred style for optimisation tasks, as it allows designers to distinguish between an inefficient algorithm and a high-demand service. For instances where a process is a constant background task or where the frequency is irrelevant to the specific architectural comparison, the simplified [Wh] notation may be utilised.
- **Data Storage Metrics:** Storage units are annotated with their energy consumption in Wh, shown below the component. The physical size of the data store is indicated in the top-right corner using the notation 'n=X GB'. This separation of concerns allows designers to focus on both the storage requirements and their associated energy costs. The energy annotation can represent either the total annual cost of maintaining the stored data or, when relevant, the energy cost of data growth over time (e.g., [Wh | +ΔWh growth]). This approach enables stakeholders to evaluate both the immediate and long-term energy use implications of data storage decisions.
- **Energy Hotspots:** To support rapid decision-making, EnPDs make use of a structured pipeline. First, the total energy consumption for each individual component and data flow is calculated by multiplying its unit energy cost by its expected execution frequency. Second, a pool of hotspot candidates is established by filtering these results to include only *actionable* nodes and data flows (the ones that the stakeholders have architectural control over). For instance, highly intensive cloud components, such as AI model training machines, are deliberately excluded from this candidate pool if they cannot be directly altered by the system design team. Third, the hotspots are determined by isolating the candidates that exceed a predefined accepted threshold value that depends on the task associated with that node or data flow. The hotspots are then visually highlighted using dashed red patterns: processes and data stores are marked with a red

dashed background, while data flows are represented with dashed red arrows. This systematic filtering provides an immediate cognitive map of energy-intensive, addressable components, allowing system designers to quickly identify and prioritise them for refinement.

By standardising these conventions, the EnPD framework ensures that energy annotations are not merely decorative but serve as a basis for comparing different architectures. This alignment between measurement and design allows for green software principles to be systematically applied during the earliest stages of the software development life cycle.

4 Case Study: mAIndmap

Using a case study, we show how EnPDs can support architectural decision-making even before a concrete implementation exists. We start with a skeleton Data Flow Diagram derived from the project requirements document, annotate it with yearly energy labels based on estimates and then refine the architecture incrementally. The goal is to make trade-offs visible already during the design phase, while design choices are still open.

NOLAI is an organisation focused on developing AI-driven solutions for educational purposes. One of their projects, mAIndmap, centres on an AI-assisted tool for vocabulary learning in early primary education (grades 1–3, ages 4–6, in the Dutch education system). Teachers in these grades use mindmaps to expand children’s vocabulary: a central concept is shown with an image in the middle, surrounded by related words and images. In practice, creating such mindmaps is time-consuming and disrupts the flow of lessons, especially when teachers have to search for age-appropriate illustrations during classes. The mAIndmap project addresses this by generating the mindmap structure and accompanying images automatically, with the help of several AI components.

4.1 Baseline: Skeleton DFD

The proposed architecture relies on two main AI components. First, a word-embedding model trained on the *BasiLex* corpus [22] (~11.5 million words from Dutch primary school reading material) computes semantic similarity and determines word placement in the mindmap. Second, a text-to-image model generates age-appropriate illustrations in a consistent visual style. To keep latency low during lessons, images for words likely to arise in grades 1–3 are pre-generated, with missing images generated on demand.

The architecture spans two contexts relevant to this case study: the school environment, which consists of classrooms and infrastructure, and a cloud-based back-end. The project’s design proposal contains a DPD that documents data flows and privacy classifications across these contexts, which serves as the starting point for the design of the skeleton Data Flow Diagram. After omitting details from the project proposal that are not relevant to the case study, we arrive at the DFD shown in Figure 2.

4.2 Constructing an EnPD from the DFD

Then, to construct an EnPD from the DFD, we maintain the same structure and add yearly energy labels to components and inter-context data flows. The model represents a single school, with all energy terms scaled accordingly. The labels account for baseline

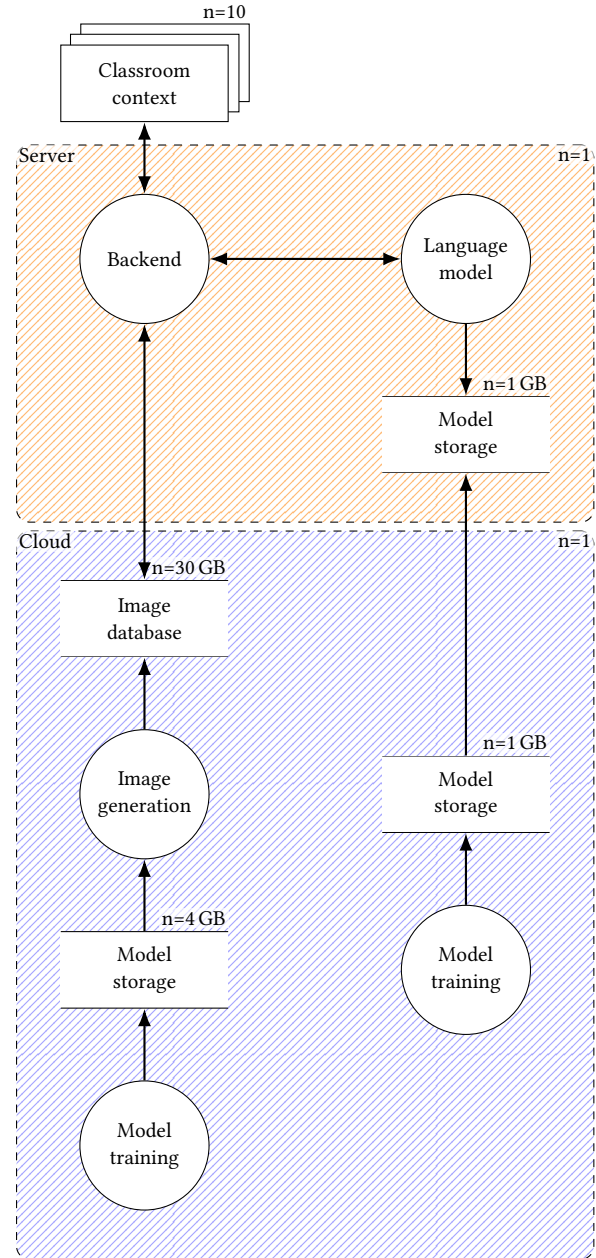


Figure 2: Skeleton Data Flow Diagram, extracted from the described architecture in the mAIndmap project proposal.

infrastructure power, persistent storage, inference workloads and network transfer, using the assumptions summarised in Table 1.

In the school context, three elements contribute energy: local infrastructure, local language model storage and local language model inference. We assume an always-on local server with a baseline draw of 10 W, corresponding to 88 kWh per year. Across grades 1–3, we estimate 10 classes per school and 20 lessons per class per year, giving 200 mindmap requests annually. Language model inference

Table 1: Baseline assumptions and energy estimates used to calculate energy costs for the mAlndmap components.

Component	Cost
Mindmap structure size	10 kB/mindmap
Generated image size	300 kB/image
Network transfer	0.1 Wh/GB
Local storage	15 W/TB
Cloud storage	5 W/TB

executes locally at 0.1 Wh per mindmap structure. Its 1 GB local model storage contributes 131 Wh per year.

The cloud environment covers image generation, shared image storage and delivery to schools. The language model is synchronised once per year, while image retrieval requires a network transfer for every requested image. If an image is missing from the shared database, it is generated on demand and then retained in storage. We estimate 15 images per mindmap, yielding 3,000 image retrievals per year across 200 lessons, of which 300 require generation.

The cloud context also contains training and model storage. Model training is assumed to consume 10 kWh for the language model and 500 kWh for the image model. We deliberately do not consider these nodes as hotspots, as they are not directly actionable for the stakeholders in this case study. Cloud storage is set to 5 W/TB and cloud image inference to 15 Wh per image, which is lower than the school context, reflecting economies of scale relative to local execution. The shared image database is assumed to start at 100,000 images and grow by 1,000 images per year. With these assumptions, we obtain the baseline EnPD in Figure 3.

4.3 Moving Image Generation to the School

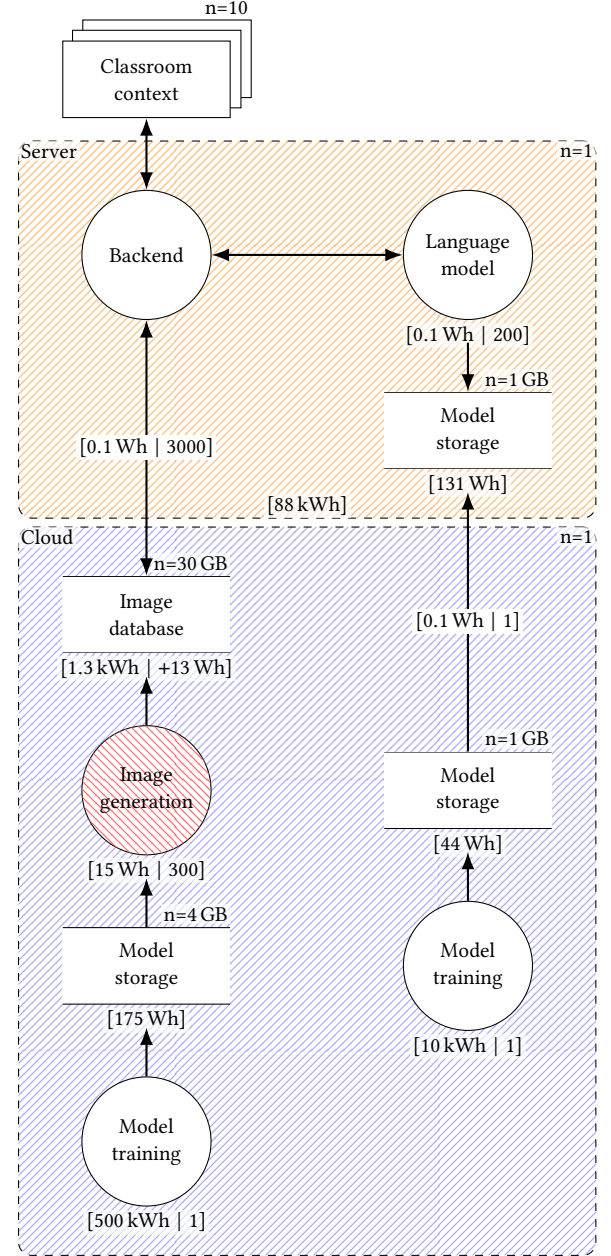
The design proposal explicitly mentions robust operation for schools with unstable internet connectivity. In practice, this means moving image storage and generation from cloud to school. This is feasible without introducing a separate machine: the existing local server that hosts the language model can host the image model as well.

Moving image generation and storage to the school substantially reduces network transfer. Instead of fetching all images from the cloud, only model weights are downloaded periodically. The local image database can also be smaller than the shared cloud database, assuming higher vocabulary overlap within one school. We therefore assume a local image database that starts at 10,000 images and grows by 300 images per year.

However, moving image generation to the school increases local energy demand. Without cloud-scale efficiency, school-local image generation is more expensive per image (25 Wh local versus 15 Wh cloud). In addition, image generation may require dedicated hardware, which we model as an increase in the server’s baseline power draw from 10 W to 20 W. Total school-side energy therefore rises to 175 kWh per year. The resulting architecture is shown in Figure 4.

4.4 Addressing the Energy Hotspot

In the local inference variant, the EnPD makes one hotspot explicit: local image generation. The hotspot highlights the local image inference as a key target for energy reduction. One approach to

**Figure 3: EnPD with a one-year timescale, created by extending the skeleton Data Flow Diagram with energy information.**

address this is by compressing the model before distributing it to schools. Compression reduces model size and compute demand, typically at some quality cost.

This change keeps the connectivity and autonomy benefits of local deployment while reducing both main energy values. First, image inference energy decreases from 25 Wh to 5 Wh. Second, lower compute demand allows lighter hardware, which is reflected by a reduction in baseline server power draw to 15 W. This results in the

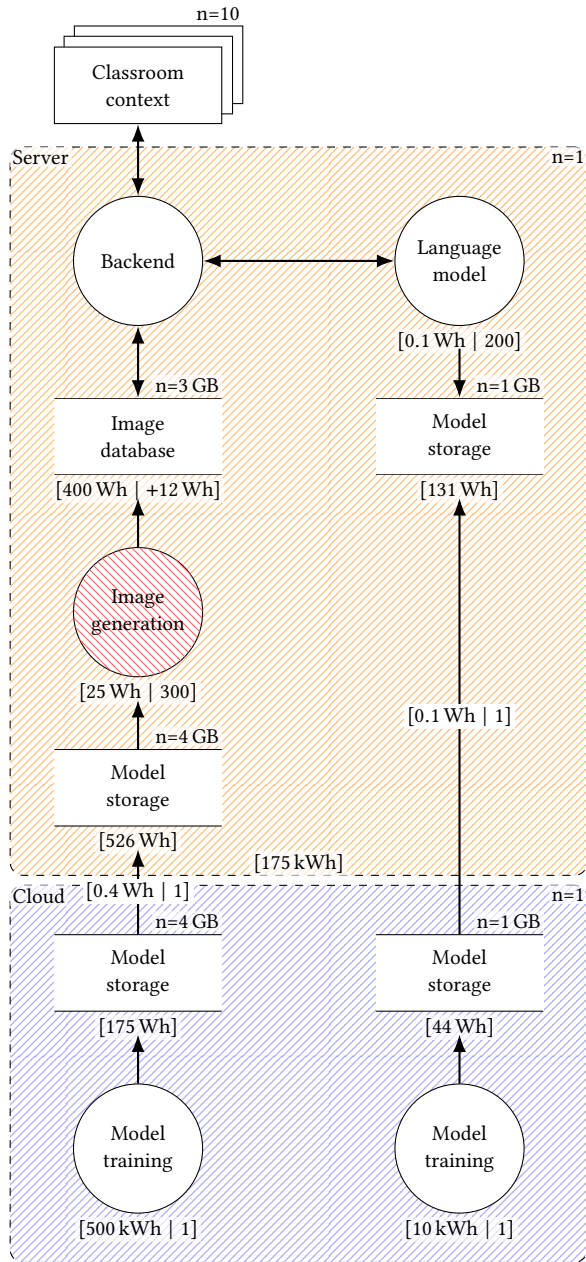


Figure 4: EnPD with a one-year timescale, using school-local image inference and storage.

architecture shown in Figure 5. In this variant, compression reduces both image inference cost and baseline infrastructure demand while preserving local operation.

4.5 Reflection

Across the three EnPD variants, the method supports design in two concrete ways. First, annotating the DFD with energy labels highlights that local image generation dominates yearly energy, which is not obvious from the data flow alone. Second, comparing

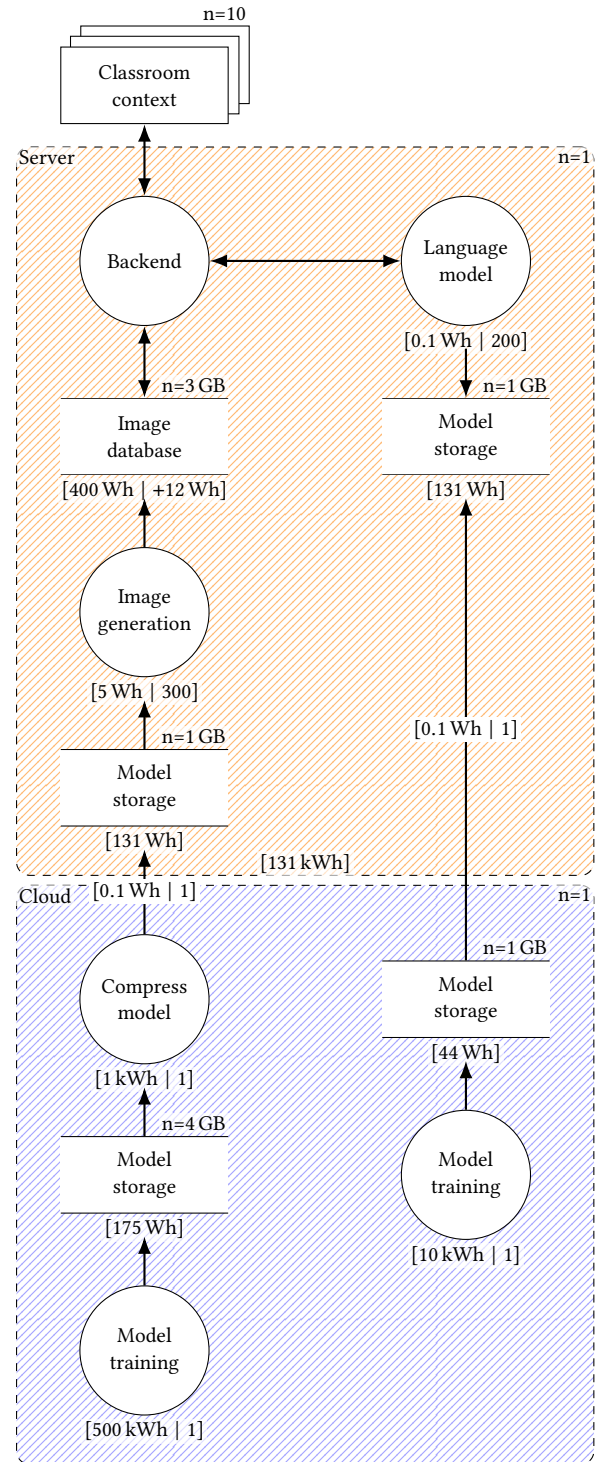


Figure 5: EnPD with a one-year timescale, using local image inference and storage with a compressed model.

variants makes trade-offs explicit: moving image generation to the school decreases network reliance, but introduces a local hotspot and causes image duplication across schools. Model compression addresses this hotspot, at the cost of potential quality loss.

The case study also shows why the stakeholder perspective should be explicit. From a school's perspective, the most actionable changes are local deployment and hardware choices. From a platform provider perspective, the most actionable changes are cloud architecture and shared-service design. Stating this perspective clearly helps ensure that identified hotspots translate into concrete design decisions.

These lessons generalise beyond mAIndmap. Any system that balances centralised and distributed processing faces similar tensions. EnPDs offer a way to expose those tensions early, before the architecture is fixed. They can be used at different levels, from deployment level choices to targeted design decisions. Precise measurements are not required: rough estimates are sufficient to identify hotspots, motivate alternatives and reason about stakeholder-specific trade-offs.

5 Discussion

This section reflects on the key findings, limitations and broader implications of our approach.

5.1 Insights from the Case Study

The mAIndmap case study demonstrates how EnPDs can reveal critical energy trade-offs in architectural decisions. Our analysis of three distinct configurations highlights how different approaches to image processing and storage can significantly impact a system's overall energy footprint. The case study reveals that the choice between centralised and distributed processing involves complex trade-offs between computational load, network transfer costs and energy consumption.

The EnPD framework proves particularly valuable in visualising these trade-offs, enabling stakeholders to understand how architectural decisions affect energy consumption at both component and system levels. For instance, while local processing of images eliminates network transfer costs, it introduces substantial computational overhead. Conversely, centralised processing reduces per-school computational requirements but increases network-induced energy consumption. These insights suggest that EnPDs can help software developers and stakeholders make more informed decisions about energy efficiency during early design phases.

5.2 Scope and Practical Considerations

The framework's practical application involves both opportunities and considerations. While EnPDs can potentially incorporate both estimated and measured energy values, our current case study demonstrates their utility with estimated data. This approach is particularly valuable during early architectural design when precise measurements may not yet be available. Future work could validate the framework using real-time energy monitoring tools, such as CarbonTracker [1], to explore how measurement data affects decision-making.

Several limitations of the current EnPD framework should be acknowledged. The framework focuses primarily on operational energy consumption, excluding environmental aspects of hardware manufacturing and disposal. While this scope is appropriate for software developers and stakeholders making design decisions, a more comprehensive environmental assessment would require integration with life cycle assessment (LCA) methodologies [9].

The EnPD framework operates at the architectural level, where understanding relative energy impacts between components takes precedence over precise temporal dynamics. While the timing of operations can influence energy consumption (e.g., using renewable energy or running components during off-peak hours), such considerations are highly dependent on specific deployment contexts and energy grids. Rather than attempting to model these complex, location-specific factors, EnPDs provide the foundation for such optimisations by identifying energy-intensive components that could be scheduled or relocated based on environmental considerations.

To summarise, the current EnPD framework deliberately excludes several factors that influence a system's total environmental impact. These include:

- **Embodied carbon:** The emissions from hardware manufacturing, transportation and disposal are excluded, as they depend heavily on supply chain factors and local manufacturing practices. For mobile devices, embodied carbon can represent up to 75% of total life cycle emissions, while for servers operational emissions typically dominate ($\approx 75\%$ of total).
- **Idle energy consumption:** The energy consumed by hardware components when idle is mostly excluded, as it can be easily incorporated at the context level when needed. This design choice maintains focus on software-controlled energy consumption while allowing for flexibility in specific use cases.
- **Network infrastructure:** The energy consumed by network routers, switches and other infrastructure components between system boundaries is excluded. While this could be modelled, it typically falls outside the control of software developers and exhibits high variability across different network configurations and geographic regions, making it less relevant for architectural decision-making.
- **Peripheral devices:** Energy consumption of user devices like monitors, keyboards and other peripherals is typically not included, as it falls outside the core system architecture being modelled.

These exclusions reflect our focus on software architecture decisions within the control of development teams, while acknowledging that a complete environmental assessment would require complementary tools and methodologies.

6 Related Work and Broader Implications

The integration of sustainability metrics into system design necessitates both measurement methodologies and visualisation tools. This section contextualises the EnPD framework within the broader landscape of green software research.

6.1 Data Flows

The EnPD framework builds upon established diagramming techniques, such as Data Processing Diagrams [6], to model energy flows within software systems. DPDs provide a shared visual language for multidisciplinary teams to map context boundaries and implement mitigating measures like pseudonymisation or encryption [5]. However, while traditional modelling approaches focus on functional requirements and data flow, they lack the native notation required to represent energy as a first-class architectural constraint. The EnPD framework addresses this limitation by treating energy data as essential architectural attributes, moving beyond legacy design principles that focused primarily on coupling and cohesion without regard for hardware-induced resource use [25].

6.2 Standardised Energy Measurement

For architectural models to serve as valid analytical tools, they must be supported by measurement protocols. The Green Software Measurement Model (GSMM) provides a reference framework for assessing resource efficiency in software products, delineating components such as measurement setup, data evaluation and life cycle boundaries [10]. This standardisation is critical for enabling reproducible evaluations and meaningful comparisons across different architectural implementations.

Quantifying environmental impact requires precise accounting of both operational and embodied carbon. Frameworks like Green Algorithms [15] provide standardised methodologies to compute carbon dioxide equivalent (CO_2eq) emissions based on runtime, hardware power draw and the carbon intensity of local energy grids. Complementary tools, such as CarbonTracker, enable real-time tracking and prediction of emissions during AI model training, integrating power metrics directly into the development workflow [1].

Recent research emphasises that such standardisation is essential to overcome the *carbon tunnel vision* that often ignores critical impact categories like metal scarcity [16]. However, while these tools excel at precise measurement of existing systems, they often require detailed runtime information that may not be available during early architectural design. EnPDs complement these approaches by providing a framework for energy-aware design decisions before implementation, using estimates that can be refined with empirical data as the system develops. This forward-looking perspective is particularly valuable for identifying potential energy hotspots and evaluating architectural alternatives when changes are still relatively inexpensive to implement.

6.3 Green AI and Architectural Optimisation

The motivation for EnPDs is particularly acute in AI systems, where the exponential growth in computational demand has led to significant environmental concerns [19]. Studies indicate that training large language models can emit up to 284 tonnes of CO_2eq , highlighting the urgent need for sustainable design practices [21]. Empirical evaluations of design patterns demonstrate that architectural choices can significantly impact energy consumption. For instance, certain object-oriented design patterns may increase energy usage by up to 25% compared to optimised alternatives [8]. Similarly, research on mobile applications identifies specific energy

patterns that can be applied to reduce energy consumption [3]. By integrating these insights with LCA methodologies, EnPDs enable software developers to identify energy *hotspots* and optimise systems for minimal environmental impact before deployment [16]. Recent work also highlights the importance of considering both operational and embodied carbon in AI systems. While operational emissions often dominate current discourse, embodied carbon from hardware manufacturing can surpass operational footprints as efficiency improves [24].

7 Conclusion

The Energy Processing Diagram framework offers a structured way to integrate energy awareness into software systems' design, providing a common language that bridges the gap between technical and non-technical stakeholders. By combining the transparency of DPDs with some of the methods proposed in GSMM, EnPDs enable meaningful, interdisciplinary discussions about sustainability without sacrificing technical depth. This shared visual vocabulary facilitates communication between software developers, system architects, product managers and environmental specialists, allowing them to collaboratively identify and address energy concerns from the earliest design phases.

The mAIIndmap case study illustrates how careful architectural choices (like performing heavy computation offline or clarifying energy use across contexts) can help reduce environmental impact. By using visual aids, EnPDs make energy efficiency considerations more accessible and actionable, transforming abstract energy metrics into concrete design decisions. This approach helps teams balance competing priorities, such as sustainability, performance and privacy, while making the environmental implications of architectural choices more transparent and understandable to all stakeholders involved in the software development life cycle.

8 Future Directions

The EnPD framework presents several promising avenues for future development and application. First, the integration of EnPDs with emerging standards like the Software Carbon Intensity (SCI) specification [4] could enhance their practical utility in specialised domains, such as federated learning and edge computing.

The development of automated tools to generate EnPDs from existing codebases or architectural descriptions would significantly lower the barrier to adoption, making energy-aware design more accessible to development teams.

A particularly promising extension involves the Origami Plot [7], a visualisation technique that could serve as an energy label within the EnPD framework. Similar to energy efficiency labels on household appliances, this visual approach would move beyond single-metric representations by simultaneously displaying performance across multiple sustainability criteria. The Weighted Origami Plot variant would further enhance this capability by allowing stakeholders to assign relative importance to different sustainability metrics, ensuring the visual area directly reflects strategic priorities and trade-offs.

Other technical developments could include formalising EnPDs as a complete language grammar, to provide a stronger foundation for energy-aware architectural modelling [12, 14].

Additionally, further case studies demonstrating the practical benefits of EnPDs in real-world projects, such as the mAIIndmap educational platform, could help build consensus around EnPDs' utility and refine the framework through practical application. An interesting direction for future research would be a comprehensive case study involving software developers and other stakeholders, where the practical impact of EnPDs on design decisions could be evaluated through interviews and observational studies. This would help validate the framework's effectiveness in real-world development environments.

The intersection of diagramming and energy measurement represents a promising avenue for advancing sustainable software development. By providing an accessible framework for visualising energy consumption, EnPDs can help bridge the gap between architectural design and environmental sustainability, ultimately contributing to the development of more energy-efficient and responsible software systems.

Acknowledgments

This research is performed in context of the Dutch National Education Lab AI (NOLAI), funded by the Dutch National Growth Fund.

References

- [1] L.F.W. Anthony, B. Kanding, and R. Selvan. 2020. Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models. [doi:10.48550/ARXIV.2007.03051](https://arxiv.org/abs/2007.03051) Version Number: 1.
- [2] A. Berthelot, E. Caron, M. Jay, and L. Lefèvre. 2024. Estimating the Environmental Impact of Generative-AI Services Using an LCA-Based Methodology. *Procedia CIRP* 122 (2024), 707–712. [doi:10.1016/j.procir.2024.01.098](https://doi.org/10.1016/j.procir.2024.01.098)
- [3] L. Cruz and R. Abreu. 2019. Catalog of Energy Patterns for Mobile Applications. *Empirical Software Engineering* 24, 4 (Aug. 2019), 2209–2235. [doi:10.1007/s10664-019-09682-0](https://doi.org/10.1007/s10664-019-09682-0)
- [4] J. Dodge, T. Prewitt, R. Tachet Des Combes, E. Odmark, R. Schwartz, E. Strubell, A.S. Luccioni, N.A. Smith, N. DeCario, and W. Buchanan. 2022. Measuring the Carbon Intensity of AI in Cloud Instances. In *2022 ACM Conference on Fairness, Accountability and Transparency*. ACM, Seoul Republic of Korea, 1877–1894. [doi:10.1145/3531146.3533234](https://doi.org/10.1145/3531146.3533234)
- [5] J. Doesburg, B. van Gastel, and E. Poll. 2025. Pseudonymization as a Service: Compartmentalizing and Controlling Data Processing in Evolving Systems with Micropseudonymization. (2025).
- [6] J. Doesburg, P. van Gastel, B. van Gastel, and E. Poll. 2025. Data Processing Diagrams: A Modeling Technique for Privacy in Complex Data Processing Systems. In *Privacy and Identity Management. Generating Futures*, F. Bieker, S. De Conca, J.M. Del Alamo, and Y.S. Martín (Eds.). Vol. 705. Springer Nature Switzerland, Cham, 115–131. [doi:10.1007/978-3-031-91054-8_6](https://doi.org/10.1007/978-3-031-91054-8_6) Series Title: IFIP Advances in Information and Communication Technology.
- [7] R. Duan, J. Tong, A.J. Sutton, D.A. Asch, H. Chu, C.H. Schmid, and Y. Chen. 2023. Origami Plot: A Novel Multivariate Data Visualization Tool That Improves Radar Chart. *Journal of Clinical Epidemiology* 156 (April 2023), 85–94. [doi:10.1016/j.jclinepi.2023.02.020](https://doi.org/10.1016/j.jclinepi.2023.02.020)
- [8] D. Feitosa, R. Alders, A. Ampatzoglou, P. Avgeriou, and E.Y. Nakagawa. 2017. Investigating the Effect of Design Patterns on Energy Consumption. *Journal of Software: Evolution and Process* 29, 2 (Feb. 2017), e1851. [doi:10.1002/smr.1851](https://doi.org/10.1002/smr.1851)
- [9] J.B. Guinée, S. Cucurachi, P.J.G. Henriksson, and R. Heijungs. 2018. Digesting the Alphabet Soup of LCA. *The International Journal of Life Cycle Assessment* 23, 7 (July 2018), 1507–1511. [doi:10.1007/s11367-018-1478-0](https://doi.org/10.1007/s11367-018-1478-0)
- [10] A. Guldner, R. Bender, C. Calero, G.S. Fernando, M. Funke, J. Gröger, L.M. Hilty, J. Hörschemeyer, G.-D. Hoffmann, et al. 2024. Development and Evaluation of a Reference Measurement Model for Assessing the Resource and Energy Efficiency of Software Products and Components – Green Software Measurement Model (GSMM). *Future Generation Computer Systems* 155 (June 2024), 402–418. [doi:10.1016/j.future.2024.01.033](https://doi.org/10.1016/j.future.2024.01.033)
- [11] A. Hindle. 2012. Green Mining: A Methodology of Relating Software Change to Power Consumption. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, Zurich, 78–87. [doi:10.1109/MSR.2012.6224303](https://doi.org/10.1109/MSR.2012.6224303)
- [12] R. Kersten, P.P. Toldin, B. van Gastel, and M. van Eekelen. 2014. A Hoare Logic for Energy Consumption Analysis. In *Foundational and Practical Aspects of Resource Analysis*, Ugo Dal Lago and Ricardo Peña (Eds.). Vol. 8552. Springer International Publishing, Cham, 93–109. [doi:10.1007/978-3-319-12466-7_6](https://doi.org/10.1007/978-3-319-12466-7_6) Series Title: Lecture Notes in Computer Science.
- [13] M. Klesel and U. Messer. 2026. Good for the Planet, Bad for Me? Intended and Unintended Consequences of AI Energy Consumption Disclosure. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. ACM, Barcelona Spain, 1–14. [doi:10.1145/3772318.3791015](https://doi.org/10.1145/3772318.3791015)
- [14] M. Klinik, B. van Gastel, C. Kop, and M. van Eekelen. 2020. Skylines for Symbolic Energy Consumption Analysis. In *Formal Methods for Industrial Critical Systems*, Maurice H. Ter Beek and Dejan Ničković (Eds.). Vol. 12327. Springer International Publishing, Cham, 93–112. [doi:10.1007/978-3-030-58298-2_3](https://doi.org/10.1007/978-3-030-58298-2_3) Series Title: Lecture Notes in Computer Science.
- [15] L. Lannelongue, J. Grealey, and M. Inouye. 2021. Green Algorithms: Quantifying the Carbon Footprint of Computation. *Advanced Science* 8, 12 (June 2021), 2100707. [doi:10.1002/advs.202100707](https://doi.org/10.1002/advs.202100707)
- [16] A.-L. Ligozat, J. Lefevre, A. Bugeau, and J. Combaz. 2022. Unraveling the Hidden Environmental Impacts of AI Solutions for Environment Life Cycle Assessment of AI Solutions. *Sustainability* 14, 9 (April 2022), 5172. [doi:10.3390/su14095172](https://doi.org/10.3390/su14095172)
- [17] S. Luccioni, B. Gamazaychikov, S. Hooker, R. Pierrard, E. Strubell, Y. Jernite, and C.-J. Wu. 2024. Light Bulbs Have Energy Ratings – So Why Can't AI Chatbots? *Nature* 632, 8026 (Aug. 2024), 736–738. [doi:10.1038/d41586-024-02680-3](https://doi.org/10.1038/d41586-024-02680-3)
- [18] G. Pinto and F. Castor. 2017. Energy Efficiency: A New Concern for Application Software Developers. *Commun. ACM* 60, 12 (Nov. 2017), 68–75. [doi:10.1145/3154384](https://doi.org/10.1145/3154384)
- [19] R. Schwartz, J. Dodge, N.A. Smith, and O. Etzioni. 2020. Green AI. *Communications of The ACM* (2020). [doi:10.1145/3381831](https://doi.org/10.1145/3381831)
- [20] C. Siy, G. Do, K. Jung, K.H. Lee, and J. Kim. 2025. E-litter: Nudging Email Usage Behavior One Byte at a Time MHC1007. *Proceedings of the ACM on Human-Computer Interaction* 9, 5 (Sept. 2025), 1–22. [doi:10.1145/3744198](https://doi.org/10.1145/3744198)
- [21] E. Strubell, A. Ganesh, and A. McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. *Annual Meeting of the Association for Computational Linguistics* (2019). [doi:10.18653/v1/P19-1355](https://doi.org/10.18653/v1/P19-1355)
- [22] A. Tellings, M. Hulsbosch, A. Vermeer, and A. van den Bosch. 2014. BasiLex: An 11.5 Million Words Corpus of Dutch Texts Written for Children. *Computational Linguistics in the Netherlands Journal* 4 (2014), 191–208.
- [23] R. van der Steen and B. van Gastel. 2023. The organizational hurdles of structurally reducing the energy consumption of software. In *Proceedings of the 22nd Belgium-Netherlands Software Evolution Workshop, Nijmegen, The Netherlands, November 27-28, 2023 (CEUR Workshop Proceedings, Vol. 3567)*, Fernanda Madeiral and Ayushi Rastogi (Eds.). CEUR-WS.org, 25–32. <https://ceur-ws.org/Vol-3567/paper4.pdf>
- [24] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. Aga, J. Huang, et al. 2022. Sustainable AI: Environmental Implications, Challenges and Opportunities. In *Proceedings of Machine Learning and Systems*, D. Marculescu, Y. Chi, and C. Wu (Eds.), Vol. 4. 795–813. https://proceedings.mlsys.org/paper_files/paper/2022/file/462211f67c7d858f663355eff93b745e-Paper.pdf
- [25] E. Yourdon and L.L. Constantine. 1979. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design* (1st ed.). Prentice-Hall, Inc., USA.