3

Edge AI System Engineering

Edge AI system engineering is a holistic and interdisciplinary approach to designing, developing, and managing edge AI systems. It involves applying principles, methods, and tools to define, analyse, and validate system requirements and specifications.

3.1 Systems Engineering

Systems engineering is a systematic, interdisciplinary approach to a system's design, implementation, technical management, operation, and eventual retirement. A "system" is an integrated assembly of elements that collectively deliver the capability necessary to fulfil a specific need. These elements encompass all hardware, software, equipment, facilities, personnel, processes, and procedures required to achieve the desired system-level outcomes. The outcomes include system-level qualities, properties, characteristics, functions, behaviours, and performance metrics. The unique value of the system, beyond the sum of its parts and components, is primarily derived from the interrelationships and interactions among these parts, emphasising the importance of their integration. This holistic perspective is crucial when making technical decisions, ensuring that stakeholder requirements for functional, physical, and operational performance are met within the intended use environment throughout the system's lifecycle, all while adhering to constraints related to cost, schedule, and other factors. Systems engineering thus serves as a disciplined methodology that helps manage and contain the life cycle costs of a system, embodying a logical and structured way of thinking.

Systems engineering focuses on developing an operable system that meets specified requirements within often conflicting constraints. It is a holistic, integrative discipline where the contributions of various engineering specialities, such as electrical engineering, software engineering, power engineering, and human factors engineering, are evaluated and balanced against one another to create a cohesive whole. This approach ensures that no single discipline's perspective dominates the system design.

The system's engineering aims to achieve a safe and balanced design amidst competing interests and multiple, sometimes conflicting constraints. Throughout the process, continuous validation is essential to confirm that the system's operational goals are being met. This balanced and integrative approach ensures that the final system is robust, efficient, and capable of fulfilling its intended purpose.

A foundational principle of systems engineering is its lifecycle perspective as the development is viewed as a comprehensive process that encompasses concept development, engineering development, upgradability and post-development stages, including production, operations, training, support, and eventual disposal. By considering the entire lifecycle, systems engineering helps to mitigate risks and ensure the long-term viability and success of the system.

Traditional systems engineering lifecycle models, such as the sequential waterfall method or the more structured V-model, are predicated on a development process that flows from well-defined requirements to design, implementation, and validation. These models are applied in domains where the problem is well understood, and system behaviour is deterministic. Still, they are not well-suited for the development of edge AI systems, which are characterised by non-determinism, probabilistic reasoning, and emergent behaviours [22]. The edge AI model's performance is not guaranteed by its code but is a learned property derived from data, making its behaviour inherently uncertain.

As a result, the edge AI development lifecycle must be iterative, data-centric, and adaptive. The process is not a linear progression but a continuous cycle of data collection, model development, model training and re-training, testing, and deployment, where feedback from each stage interacts with the following. This development requires a co-design approach where the edge AI models, the target hardware, the system software, the data and data sets are not developed in sequence but are considered and optimised in parallel. This holistic perspective is captured by the data-model-HW/SW system optimisation quadruple, which asserts that these four interconnected elements must be engineered in concert to achieve a viable solution. The choice of a hardware accelerator (system) influences the type of software (system), which will determine which model architectures can be run efficiently (model), which in turn dictates the nature and volume of data required for training (data) [23].

The edge AI development lifecycle is iterative, exploratory and emergent. The system's final capabilities and performance characteristics are often discovered through empirical experimentation rather than being fully specified at the outset. A traditional iterative model assumes that while the system is constructed in increments, the overarching requirements are known and stable. In many AI-driven systems, the requirements themselves can be ambiguous (e.g., "accurately detect a threat"), and their feasibility is unknown until a model is trained and evaluated on representative real-world data [24].

The development workflow in edge AI systems can be represented by a series of experiments designed to answer questions about feasibility and performance. Each cycle or iteration of data collection, training, and testing is an experiment to determine whether a specific level of performance, such as a target accuracy, is achievable within the given resource constraints. In this development workflow, it is vital to consider the concept drift, where the statistical properties of the operational environment change over time, degrading model performance and necessitating a continuous lifecycle of monitoring, re-evaluation, and re-training to maintain the edge AI system's suitability.

The complexity and unique constraints of edge AI systems demand the rigorous application of core systems engineering principles. These principles provide the structure and discipline necessary to manage development, mitigate risks, and ensure the final system is robust, reliable, and fit for purpose. Principles such as modularity, decomposition, systematic trade-off analysis, and verification and validation can be applied to edge AI requirements.

Modularity is the practice of decomposing a complex system into smaller, independent, and interchangeable components or modules to meet the requirements of these components. In edge AI system design, this principle is critical for managing the inherent complexity. A monolithic design is challenging to develop, test, and maintain. By breaking the system into logical modules, such as a data acquisition module, a data pre-processing module, an inference engine, and a decision logic module, the requirements can be applied in parallel, and the system becomes more scalable and maintainable. This approach applies to both software, to structure the AI workflow, and hardware, where modular components enable flexible and resilient system configurations [29].

Trade-off analysis is essential for edge AI systems engineering. The constrained nature of edge devices imposes the need to make decisions that balance conflicting requirements. There is a continuous challenge between competing quality attributes: improving model accuracy often increases computational complexity, which in turn increases inference latency and power consumption [30]. Enhancing security through encryption may add computational overhead, again impacting performance. These are not choices that can be made informally; they require a systematic and evidence-based process. Formal methodologies, such as the weighted sum method or multi-attribute utility theory, provide structured frameworks for evaluating alternatives against a set of weighted criteria, enabling engineers to make proper decisions that align with project priorities [31]

Verification and validation (V&V) in the context of edge AI presents challenges that are different from those of traditional software testing. For conventional systems, V&V is the process of confirming that the system was built correctly according to its specifications. For edge AI-based systems, this is complicated by their non-deterministic nature and because the system's behaviour is probabilistic. AI models can be vulnerable to adversarial examples based on subtly perturbed inputs designed to cause misclassification, which exposes a unique failure mode that must be tested. The V&V process for edge AI must therefore encompass comprehensive data validation to check for quality and bias, rigorous model validation to assess performance on unseen data, and robustness testing to evaluate the system's resilience to adversarial attacks and unexpected environmental conditions.

In edge AI engineering, trade-off analysis and V&V are not discrete, sequential phases but are instead two components of a single, continuous feedback loop that drives the iterative lifecycle. While traditional engineering may conduct a trade study early in the design phase to select an architecture, in edge AI, the core trade-offs are influenced at every stage. The application of model compression techniques like quantisation or pruning is a direct implementation of the accuracy-versus-latency trade-off. The V&V process does not produce a simple pass/fail result but rather a set of statistical performance measures, such as "the AI model is 98% accurate and consumes 10 mW". This output is quantitative data that serves as the direct input for the next stage of trade-off analysis, prompting the designer to determine whether the performance change is an acceptable price for the resource savings. V&V is thus transformed from a quality assurance element into an ongoing system characterisation process that fuels continuous, informed decision-making.

3.2 Requirements Engineering

Requirement engineering (RE) is the process of formalising and defining needs at each stage of system design, including customer requirements, design requirements, and testing requirements. It focuses on identifying, tracing, and characterising these needs to ensure their implementation and impact are clearly understood and deals with developing and verifying the system requirements.

RE is an addition to systems engineering and focuses on an essential concept in implementing systems engineering: practices related to the process (e.g., quality, standards to be applied) or product development (e.g., non-functional, functional requirements).

RE is the discipline of establishing user requirements and specifying hardware, software, AI, and data systems. Defining requirements involves determining what users want from a system and understanding what the needs mean regarding design. It is closely related to hardware, software, AI, and data engineering and focuses on designing and developing the system that users want.

Following good requirements engineering practices helps achieve the primary objective of ensuring that the delivered system meets the customer's needs. Requirements engineering consists of establishing and documenting requirements. The various activities associated with requirements engineering are formulation, elicitation, specification, analysis, verification and validation, and management. Requirements formulation involves collecting, organising, communicating, and managing requirements. Recommended practices for software requirements specifications are provided in [54].

Several standards focus on requirements engineering. One example is ISO/IEC/ IEEE 29148-2018 [55], which addresses requirements engineering and specifies the required processes implemented in the engineering activities that result in requirements for systems and software products (including services) throughout the life cycle.

For edge AI systems, a disciplined, requirements-driven approach is essential, but its application demands a considerable paradigm shift away from traditional practices. The fundamental challenge lies in transitioning from a specification-driven development model, where system behaviour is explicitly defined, to a data-driven one, where system capabilities are learned and requirements are often emergent, probabilistic, and difficult to articulate with deterministic certainty [24].

This shift necessitates the introduction of new categories of requirements that are paramount in AI systems. While traditional RE focuses heavily on functional requirements, edge AI engineering must elevate the importance of data requirements, which concern the quality, quantity, format, and representativeness of the data used for training and testing the model [24]. Equally critical are model requirements, which define the desired qualitative attributes of the AI model itself, such as its explainability, transparency, and fairness. Finally, the autonomous nature of these systems brings ethical requirements to the forefront, demanding that the system be lawful, prevent harm, and respect human values [24].

The RE process itself must also adapt. Elicitation can no longer rely solely on stakeholder interviews; it must incorporate data-centric activities and may be augmented by AI-powered tools that can analyse large volumes of text or user feedback to identify latent needs [25]. The act of specifying requirements also changes. The precision and iterative refinement involved in prompt engineering for large language models offer a useful analogy for how requirements for AI systems must be crafted: with clarity, context, and a willingness to refine based on the system's response.

This evolution changes the very definition of a requirement in the context of systems engineering. A requirement for an edge AI system evolves from being a static, deterministic statement of need into a dynamic, negotiated contract of acceptable performance under conditions of uncertainty.

For a task like "recognise a stop sign," a complete, deterministic specification that covers every possible real-world variation is impossible. Therefore, the requirement cannot be "The system shall correctly identify all stop signs." Instead, the requirement specification must define the acceptable bounds of the system's probabilistic behaviour. It becomes a composite statement comprising three key elements: (1) a set of evaluation measures, such as recall and precision; (2) criteria for acceptable values of these measures, such as "recall shall be greater than 99.5% and precision shall be greater than 98%"; and (3) a description of the operational context and data used to validate these criteria [26, 27]. This transforms the requirement into a testable element that explicitly captures the agreed-upon trade-offs, for example: "The system shall achieve a minimum F1-score of 0.9 for object detection while maintaining an end-to-end latency below 33 ms and consuming less than 200 mW of power."

Distributed AI-intensive systems raise several challenges for requirements and systems engineering. To guarantee a desired AI system behaviour, it is essential to ensure system attributes such as safety, robustness, and

quality, and to establish process support, which involves addressing four key areas [28]:

- Defining contextual descriptions and requirements by properly describing and formulating requirements on the context in which an AI-intensive system is operated. For example, if a trained machine learning model is placed in another context, desired system behaviour and especially safety attributes can no longer be guaranteed without retraining the model on the new context.
- Setting data attributes and requirements considering the need for proper definitions of quality attributes of, and requirements on data that are used in the AI-intensive system. It is argued that data is the most critical element of an AI system and that system quality attributes, such as safety and robustness, cannot be provided without the ability to ensure such properties on the data used in the system.
- Establishing performance definitions in the context and data of an AI-intensive system raises the question of how the fulfilment of these requirements and the performance of the AI-intensive system can be monitored, considering that it is vital to understand first what needs to be monitored, before asking how it can be measured and monitored.
- Considering human factors and how to integrate human factor requirements in the AI-based system development process to increase the safety, acceptance and trust.

It is important to prioritise RE in the lifecycle development of AI and edge AI before the technical implementation, which extends to explainability (XAI) and its holistic incorporation as well. Explainability is an emerging non-functional requirement that has been highlighted as a critical quality aspect. When developing and deploying AI technologies, RE is essential to understand and document the needs, expectations, and constraints of different stakeholders. To ensure transparency, interpretability, and accountability of the AI system, it is crucial to involve end-users, domain experts, regulatory bodies, and ethicists in identifying and articulating the requirements related to explainability [32].

Requirements engineering principles can support the definition of FRs and NFRs, which serve different, unique purposes. FRs outline the provided functionality by the system, focusing on user needs and specific tasks [35] and are defined in such a way that they are testable with input-output pairs, even though this is not always feasible. NFRs do not detail what the system does, but rather how it does it, i.e., they highlight quality attributes and

user experience, often requiring qualitative evaluations due to the difficulty of measuring or testing them directly [34]. FRs shape the system's core functionalities, whereas NFRs specify quality goals [33].

An overview of the landscape of techniques and practices in RE for ML is provided in [36].

As generative AI models evolve, they will be utilised in systems and requirements engineering to improve the analysis and classification of engineering requirements for edge AI systems, which underpin product design and development in these areas.

Classifying requirements in software systems is an established practice that involves addressing specific, categorised tasks in software or quality requirements, for standardised classification as defined by INCOSE's (International Council on Systems Engineering) standards [37] using categories such as FRs and NFRs. The advancements in generative AI can offer rapid and accurate analyses that increase systems engineers' efficiency. Challenges remain due to AI's disposition to generate misinterpretations, as seen in diverse applications [38, 39].

The study presented in [40] explores how generative AI can help automate and improve key steps in systems engineering. It examines AI's ability to analyse system requirements based on INCOSE's good requirement criteria, identifying well-formed and poorly written requirements. The AI models used classify requirements and explain why some do not meet the standards. The study evaluates the accuracy and reliability of AI in identifying quality issues by comparing AI assessments with those of experienced human engineers and explores AI's ability to classify FRs and NFRs and generate test specifications based on these classifications. The study aims to assess AI's potential to streamline engineering processes and improve learning outcomes through both quantitative and qualitative analysis, highlighting the challenges and limitations of AI, ensuring its safe and ethical use in professional and academic settings [40].

3.2.1 Requirements Engineering Evolution

The process of defining requirements is evolving as edge AI systems incorporate autonomous functions and new technologies to address what users want from the system and require understanding the implications for the design of these systems.

One of the primary objectives of edge AI system architecture is to determine the optimal partitioning of the system. This process involves

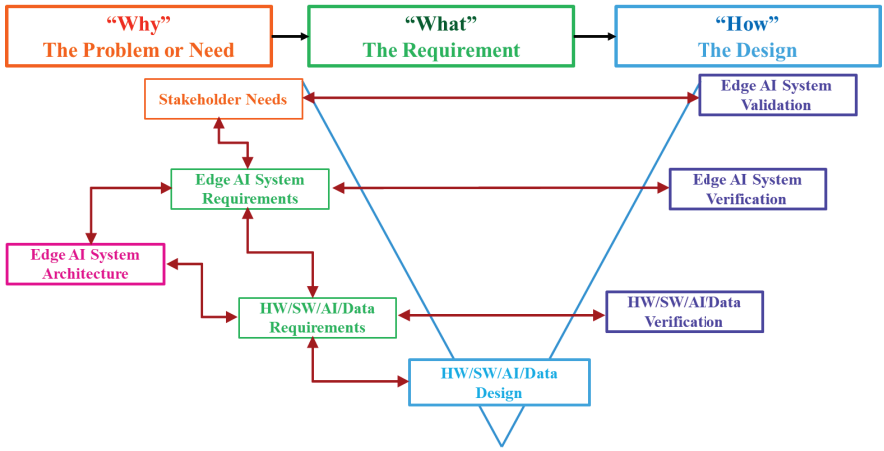

Figure 3.1 The process of writing requirements using the V-Model development life-cycle.

identifying how requirements should be allocated to specific system elements. The edge AI system elements can include hardware, software, AI, edge AI, GenAI, agentic AI and data. As these system elements are defined, additional requirement statements, known as derived requirements, must be created. These derived requirements specify relationships among architectural elements, provide clarity at lower levels of abstraction, and define specific design constraints or performance levels. This allocation and derivation are accomplished through the recursive application of requirements definition processes. The process of writing requirements when using the V-Model development life-cycle is illustrated in Figure 3.1.

In this context, the principles of requirements engineering, as outlined in ISO/IEC/IEEE 29148:2018 and the accompanying text, can provide a structured framework for managing the complexity of edge AI systems.

Certain requirements cannot be derived until specific portions of the architecture or design have evolved. Many requirements depend heavily on the interoperation and interoperability of multiple edge AI system elements. The information throughput of edge systems depends on the complex interactions among system hardware, software, AI algorithms, frameworks, data, datasets and the operational environment. To capture the elements effectively, the requirements, architecture, and design processes must be applied recursively and iteratively. The flow for transferring edge AI system requirements into subsystem requirements, system architecture, and design is illustrated in Figure 3.2.

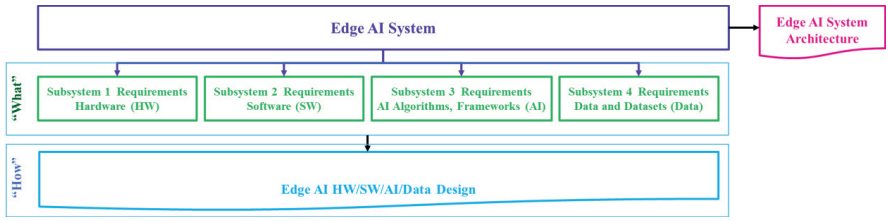

Figure 3.2 Edge AI system requirements flow into subsystem requirements, architecture and design.

Even in scenarios where requirements engineering is well-resourced, the level of analysis is seldom applied uniformly across the entire edge AI system. Experienced engineers can often identify early in the process where existing or off-the-shelf solutions can be adapted for specific edge AI system elements. As a result, requirements allocated to the standard elements may require less analysis. In contrast, elements where the solution is less obvious, or novel, require further, more detailed analysis. Critical requirements, those involving high risk or impacting public safety, the environment, or health, must always be subjected to the most rigorous analysis.

Requirements processes, and their resulting specifications depend on the scope of the system being defined. Requirements for a system or element are always subject to higher-level requirements regarding business operations. These requirements are allocated progressively to lower-level edge AI systems. A representation of the requirements scope in a business context is illustrated in Figure 3.3.

The hierarchy of specifications generally includes several key documents (ISO/IEC/IEEE 29148:2018). The business requirements specification represents business management levels, while the stakeholder requirements specification represents business operational levels. These flow down into the system requirements specification. At the technical implementation level, specific documents are generated, including the edge AI system requirements specification, hardware requirements specification software requirements specification, AI algorithms, frameworks requirements specification, and data and datasets requirements specification. The information items can be applied to multiple specifications iteratively or recursively.

The system requirements and the edge AI system requirements specifications are critical documents that identifies the technical requirements for the system-of-interest and defines the usability of system interaction. It outlines high-level requirements from a domain perspective, including objectives,

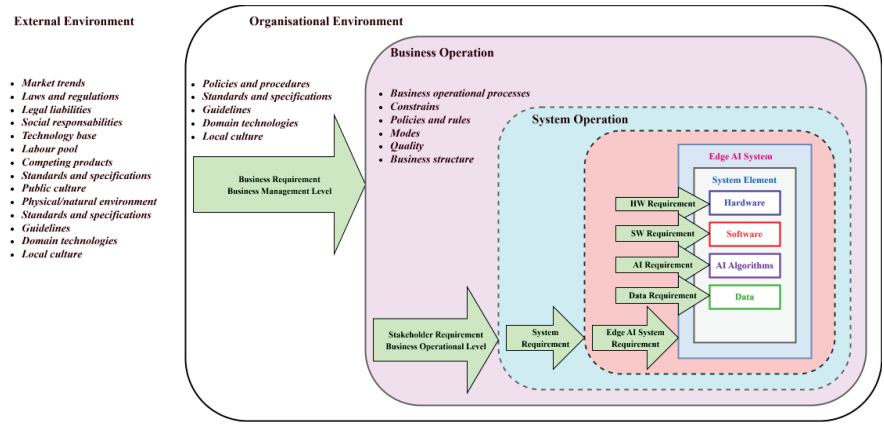

Figure 3.3 Requirements scope in a business context (Adapted from ISO/IEC/IEEE 29148:2018).

the target environment, constraints, assumptions, and non-functional requirements. It may also utilise conceptual models to illustrate system context, usage scenarios, entities, data, and workflows.

The primary purpose of the system requirements specification is to provide a description of how the system should interact with its external environment. It must completely describe all inputs, outputs, and their relationships. The system requirements specification serves as a bridge between the user and the technical community. The requirements collection in the specification must be understandable to both groups.

A distinction should be made between the structured collection of requirement information and its presentation to various stakeholders (e.g., users, data scientists, embedded engineers, etc.). The presentation of the system requirements specification should take a form appropriate for its intended use, whether as a paper document, models, prototypes, or a combination thereof. All presentations must be traceable to a common source of information to avoid ambiguity.

Generally, process requirements for developing or constructing the system should be documented in the contract, such as a statement of work, rather than in the requirements specification. If they are included in the specification, they must be clearly identified as process requirements. The system requirements specification ultimately documents the results of the need definition, operational concept, system architecture, and requirements analysis tasks, describing the acquirer's expectations for performance, quality, and verification.

The life cycle of an edge AI system demands rigorous monitoring of requirements throughout every stage of development. This continuous oversight spans from the initial analysis and specification phases through to design, verification, validation, and final testing. By maintaining this monitoring process, engineers ensure that the system evolves consistently with its original goals and adapts to any necessary changes during its development.

Defining the requirements for an edge AI system necessitates a comprehensive scope that goes beyond simple functionality. These specifications must detail every required capability, including the underlying hardware, the software stack, specific edge AI algorithms, chosen edge AI frameworks, and data and datasets handling processes. The requirements must document the exact environmental conditions and constraints under which the edge AI system is expected to operate, ensuring the technology is robust enough for its real-world application.

A critical component of this process is requirements verification, which focuses on the quality and structure of the specifications themselves. This involves a systematic examination to confirm that each requirement, both individually and as part of a collective set, is well-formed. The verification process reviews these specifications to ensure they exhibit the characteristics of high-quality requirements, such as clarity and consistency, that the requirements are necessary, verifiable, achievable and that the entire set is logically organised.

Complementing verification is requirements validation, which ensures alignment with user needs. Validation confirms that the requirements, whether viewed in isolation or as a complete package, accurately define the "right" edge AI system. This step bridges the gap between technical specifications and stakeholder or user expectations, ensuring the intended design accurately reflects what users and stakeholders envisioned for the final solution.

The requirements documents must explicitly outline the intended approaches for verification, validation, and testing. By establishing testing protocols early in the requirements phase, developers create a clear roadmap for assessing the edge AI system's performance. This approach ensures that the hardware, software, edge AI algorithms, frameworks and data meet technical standards and function effectively within the specific constraints of the edge AI environment.