VEDLIoT

Very Efficient Deep Learning in IoT

ICT-56-2020 — Next Generation Internet of Things

D2.5
Final report on the methods for requirement, specification, performance metrics and verification of context limited AI processing systems

Version 1.0

Document Information	
Contract Number	957197
Project Website	https://vedliot.eu/
Dissemination Level	PU (Public)
Nature	R (Report)
Contractual Deadline	30 April 2022
Author	UGOT & VEONEER
Contributors	Stefan Andersson (VEONEER), Oliver Bruneegard (VEONEER), Olof Eriksson (VEONEER), Hans-Martin Heyn (UGOT), Eric Knauss (UGOT)
Reviewers	Gunnar Billung-Meyer (CHR), Mario Porrmann (UOS)

The VEDLIoT project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 957197.

Table of Contents

1 Introduction **6**
1.1 VEDLIoT Project Introduction 6
1.2 VEDLIoT WP2 Introduction 8
1.3 VEDLIoT T2.1 & T2.2 outlining 9
1.3.1 Task 2.1: Requirement methods & performance metrics (M1-M18) 9
1.3.2 Task 2.2: Specification & verification methods (M1-M18) 10
1.4 The VEDLIoT Thing 10

2 Background **14**
2.1 Current best practices 14
2.1.1 System Architecture 14
2.2 Identified problems with current best practices 21
2.2.1 **PA 1**: Contextual definitions and requirements 22
2.2.2 **PA 2**: Data requirements and quality attributes of data 23
2.2.3 **PA 3**: Performance definition and monitoring 24
2.2.4 **PA 4**: Human Factors 25
2.2.5 Cross-Cutting research problem: Integration in modern system development 26

3 Architectural Framework **28**
3.1 Architecture descriptions for distributed systems 28
3.2 Clusters of concern 28
3.2.1 Behaviour and Context 30
3.2.2 Means and Resources 31
3.2.3 Quality Concerns 31
3.2.4 Architectural views for AI systems 32
3.2.5 Example of correspondences between a quality concern and the remaining architecture concerns 32
3.2.6 Example of correspondences between context, learning, and AI model 34
3.3 Levels of abstraction 36
3.3.1 Knowledge and analytical level 36
3.3.2 Conceptual level 36
3.3.3 Design level 36
3.3.4 Run time level 36
3.4 Compositional architecture framework for VEDLIoT 37
3.4.1 How to apply a compositional architectural framework in practice 38
3.5 Monitoring and controlling concepts for the run-time behaviour of advanced AI systems 40

4 Requirement Methods **41**

4.1 Define Scope, High-level Goals about Functionality and Quality, Prioritise Concerns . . . 41
4.1.1 Define Scope . . . 44
4.1.2 Functional goals . . . 45
4.1.3 Quality goals . . . 45
4.2 Define System and Context Description . . . 47
4.2.1 Define System Context . . . 48
4.2.2 Define System Requirements . . . 49
4.2.3 Define Quality Requirements . . . 50
4.2.4 Derive Quality Requirements on Data and Learning . . . 51
4.3 Quality aspects of data used for systems with deep learning . . . 53
4.3.1 A data quality assessment and maintenance framework . . . 54
4.3.2 Data Quality Workflow . . . 54
4.3.3 List of Challenges . . . 54
4.3.4 List of Data Quality Attributes . . . 55

5 The influence of distributed AI on methods for specification, performance criteria and verification . . . 57
5.1 Specification content . . . 57
5.2 Requirement and Specification processes . . . 59
5.3 Technology neutral sub system definitions . . . 60
5.4 Synchronisation requirements . . . 61
5.5 System specification method . . . 61
5.6 AI performance criteria . . . 62
5.6.1 Functional Performance . . . 62
5.6.2 Qualitative Performance . . . 62
5.6.3 Relation to intended use-case context . . . 63
5.6.4 Training Data Quality . . . 63
5.6.5 Safeguarding automation . . . 63
5.7 Additional considerations for distributed AI . . . 63
5.8 Verification Methods . . . 64
5.8.1 Simulations . . . 64
5.8.2 System Integration testing . . . 65
5.8.3 Quality verification . . . 65
5.8.4 Validation data selection . . . 65
5.8.5 Field tests . . . 65
5.9 Runtime Monitoring . . . 65
5.9.1 Real-Time Monitoring functionalities . . . 66
5.9.2 RTM and functional Safety . . . 69
5.9.3 RTM and AI/ML . . . 69
5.9.4 RTM and continuous learning . . . 70
5.9.5 RTM and post-event data correlation . . . 70
5.9.6 Summary . . . 70

6 Conclusion . . . 71

7 References . . . 73

List of Figures

1.1 Global picture of VEDLIoT. This deliverable will discuss WP2: Requirements. . . . 7
1.2 A VEDLIoT Thing . . . 11
1.3 Characteristics of "a Thing" in the context of Very Efficient Deep Learning in the IoT. . . . 13

2.1 Concerns relevant for systems of the IoT with AI components . . . 17
2.2 Taxonomy for AI based on [65] . . . 21
2.3 The development of complex, AI systems implies the need of certain abilities (blue boxes) that depend on solutions for challenges in four areas (red/yellow boxes). We argue that one has to find solutions for the red challenge areas before approaching the yellow challenge area. 22

3.1 Conceptual model of an architecture description [39] . . . 29
3.2 Conceptual system architecture for an automotive automatic emergency brake system . . . 32
3.3 Conceptual system architecture for an automotive automatic emergency brake system after safety decomposition . . . 34
3.4 Co-Design of context and constraints, learning concept, and AI model. 35
3.5 Conceptual model of a compositional architecture framework . . . 37
3.6 Steps taken for defining a compositional architectural framework for VEDLIoT . . . 38
3.7 Compositional architecture framework for VEDLIoT, categorising views in different clusters of concerns on different levels of abstraction. . . . 39

4.1 Basic flow of building VEDLIoT-based systems . . . 43
4.2 Context diagram, describing scope and high-level functionality of a potential smart mirror based system . . . 44
4.3 High-level requirements information model that connects use cases to operational context (Operational Design Domain, ODD), Data Requirements, and Quality Requirements. . . . 46
4.4 *Data Quality Workflow* Framework Component . . . 55

5.1 Visualisation of Veoneer requirement process . . . 59
5.2 An architectural framework merging clusters of concerns for an AI enabled system . . . 60
5.3 An architectural formal specification as part of requirements specification [85] . . . 61
5.4 V-model with Functional Safety and SotiF . . . 66
5.5 RTM system concept . . . 67
5.6 Trends and Performance monitoring . . . 68
5.7 Proposed real-time monitoring for a single core Arm processor . . . 69

Executive Summary

VEDLIoT supports building complex systems based on components supporting very efficient deep learning in IoT. From a system's point of view, these components behave differently than other components, especially with respect to security, safety, and robustness. This is due to the fact that deep learning algorithms solve complex tasks by returning probabilities with statistical error margins. Given a set of assumptions, this statistical error can be quantified and trade-offs between accuracy, response-time, energy-consumption and other quality attributes can be optimised. These assumptions include:

- The context of operation.
- The quality of input as well as training environment.

Based on Task 1.1, this deliverable first details the challenges that typically occur when applying current best practices in systems engineering to distributed deep learning systems. Then, an architectural framework is proposed that takes these challenges explicitly into account in order to solve them. This is achieved by introducing the concept of *Clusters of Concern*, which allow to "divide and conquer" the challenges of developing deep learning systems in IoT. Specifically, the architecture framework for VEDLIoT contains *Clusters of Concern* dealing with the *Context Description* of the system, *Learning Environment* of the deep learning components, *Communication* concerns, and a set of *Quality Concerns*, such as *Ethical Aspects*, *Safety*, *Power*, *Security*, and *Privacy* aspects. Each cluster contains a set of architectural views, which are sorted into different *Levels of Abstraction*. The result is a matrix of architectural views on the deep learning system in the IoT, which explicitly contains all relevant concerns for the development of the system. This explicit treatment of concerns allows for true "safety-by-design", "security-by-design", "ethical-by-design", or any other recognised "quality-by-design".

Based on the architectural framework for VEDLIoT, a requirement engineering method is developed which supports the architectural framework, and which

- guarantees system behaviour, given that a suitable context of operation is maintained;
- guarantees quality of decisions (=output data) of VEDLIoT components, given that a defined level of input data quality is met;
- describes quality of input, training, and output data in relation to system purpose and context of operation; and
- specifies monitoring concepts for VEDLIoT systems that continuously track performance and the expected behaviour of the system.

The preliminary work of Deliverable 2.1 has been extended by providing more detailed descriptions of the levels of abstractions and the identified clusters of concern,

by providing examples taken from the VEDLIoT use cases in WP7, and by detailing how the proposed compositional architectural framework can be applied in practice. Based on Task 2.2, the report details how system specifications, performance criteria and verification methods are derived for distributed deep learning system. As a major extension to the previous deliverable, this deliverable also includes runtime monitoring aspects of the IoT systems, because many quality guarantees, such as safety, can only hold with the help of runtime monitors.

The output of the two tasks are used to define and specify the use cases, especially through Work Package 7. The compositional architectural framework for VEDLIoT is flexible enough to serve as a way to specify more clearer the needs and requirements of all the different current and future use cases in VEDLIoT, such as automotive, industrial, and smart home use case. The explicit handling of quality aspects, such as safety, security, ethics and privacy serve as input and tool to Work Package 5. It supports specifying and implementing quality requirements on the VEDLIoT systems. Especially the new aspects about runtime monitoring discussed throughout this report are relevant for safe, robust and secure deep learning systems in the IoT. Also, in addition to the previous deliverable, methods for defining and controlling data quality requirements and context definitions are part of this deliverable. Being able to manage data quality aspects, and defining the desired context of the deep learning system in the IoT is critical for being able to state guarantees on the robustness, safety, or other quality aspects of the system.

1 Introduction

1.1 VEDLIoT Project Introduction

Computer systems have advanced at a very high rate for the last years and as a result we are currently able to hold in our hands mobile devices that have orders of magnitude larger computational power than what was available in the systems used to place a man on the moon some decades ago[1]. The enormous functional expansion of the individual devices goes hand in hand with the availability of cheap communication technology, which facilitates the ubiquitous and spontaneous networking of objects of all kinds. The result is a heterogeneous infrastructure that enables an unforeseen variety of new applications, services, and products for a smart, prosperous society. Key sectors of high relevance for improving our quality of life are transportation, industry, and our homes. For these and other sectors, applications that seemed to belong to the "science-fiction" are now starting to become implementable.

Regarding transportation, the increasing population and in particular the increasing population density in urban spaces urges for more efficient and effective transportation solutions that are at the same time flexible to fit each one's needs as well as safer than the existing ones. Self-driving cars would be a potential application towards addressing these goals. In terms of industry, (Industry 4.0), significant advances have been observed in the manufacturing of goods so essential for the development of our society. Automation has been introduced at a high degree in the industrial process by means of robotisation, for example. There is a need to push even further the efficiency, effectiveness, and productiveness. This will require to push automation to a new level. Self-optimizing and self-maintaining production lines could be one application that would target these goals. Finally, our homes are designed to provide us with the necessary space for our needs of a safe and controlled environment. At present, they are mostly a passive element in our life. User-centric automation could significantly improve the situation by enabling better self-adaptation to our changing needs as well as variations of external conditions. In addition, it should evolve to become more interactive as to further improve our quality of life. Smart-home with smart-devices could provide a solution towards the above set goals.

The increasing functionality of future technical systems is accompanied by a higher design and management complexity. A purely manual configuration of the upcoming complex devices and networks will become more and more difficult. In order to overcome these emerging problems as well as the presumably continuing heterogeneity of existing technologies, future products and services must be easily scalable and equipped with features that facilitate the automatic adaptation to each other and, especially, to the user. Furthermore, current system engineering approaches for building such systems start to fail and cannot be applied to find resource-efficient solutions to these applications. The amounts of data collected to be processed are extremely large, the computational power required is extreme and thus requires large amounts of energy and the algorithms are too complex and cannot deliver solutions within the tight time constraints. **Even describing requirements and constraints**

[1] https://www.independent.co.uk/news/science/apollo-11-moon-landing-mobile-phones-smartphone-iphone-a8988351.html

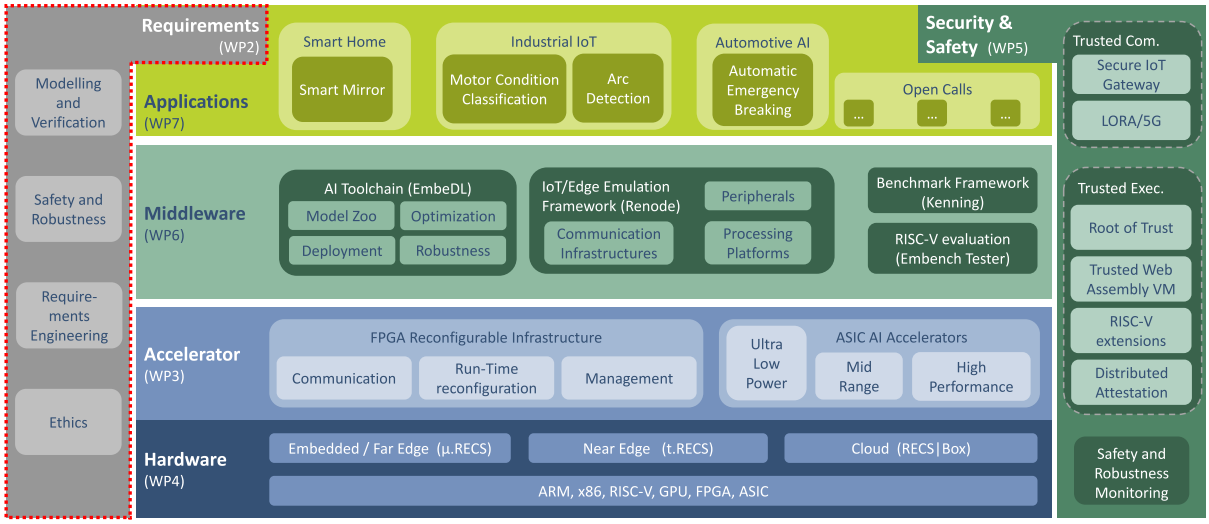

Figure 1.1. Global picture of VEDLIoT. This deliverable will discuss WP2: Requirements.

such as security, privacy, or robustness for such systems, and guaranteeing that these are fulfilled, becomes a critical challenge and threatens the deployment of such futuristic applications to society.

The solution is to introduce a disruption in traditional systems and design approaches. On the one hand, instead of traditional algorithms to solve these complex problems, algorithms based on artificial intelligence and deep learning can be effectively used to handle the large complexity in a graceful way. On the other hand, computer systems need to be broken into different components in order to be placed where they are most needed and can be more efficient. At the same time all components should work together as a large collaborative system. In such a system computation exists in devices of different "shapes" and "formats" or configurations that are connected with each other, often via a high-bandwidth connection. Such systems are known as Internet of Things (IoT). IoT devices can spread all the way from the sensor nodes collecting the physical data, which can then interact with a system at the edge that can eventually interact with other systems up to the cloud, where larger common resources are available, forming what we call the compute continuum from the edge to the cloud.

An effective enabler that aims at delivering the framework to provide the solution for advancement in the automation for different key sectors is VEDLIoT, a Very Efficient Deep Learning IoT system. A representation of the different components of VEDLIoT is shown in Figure 1.1.

In terms of hardware, VEDLIoT offers a platform, the Cognitive IoT Platform, which is composed of different systems, each one adapted better to its level of operation in the compute continuum. In addition, these systems include devices to support the latest high-bandwidth and low-latency wireless interconnection technology (e.g. 5G or LoRa). In order to deliver the necessary required computational power at the different levels, the platform may include dedicated Hardware Accelerators. Different instances of these accelerators are available so that it is possible to cover the different needs and requirements of the IoT devices.

The VEDLIoT toolchain will address the task of interfacing AI software to hardware in a comprehensive fashion leveraging and building on existing open source com-

ponents. Uniquely, the toolchain development will be closely coordinated with the Edge-to-Cloud IoT Distributed System and the emerging use cases from industry with their specific needs and with a human-centric focus aimed at delivering trusted solutions to end users. The optimisation toolchain sits between well-known and widely used DL frameworks (e.g. TensorFlow) and back-end technologies involving compilers and hardware interfaces. The intermediate representation and optimisation will be agnostic to underlying hardware platform offering support and interoperability across a range of hardware platforms. Robustness, privacy and security measures will be incorporated into the toolchain. It will enable processor vendors, device makers, and deep learning developers to rapidly bring new and independent innovations in machine learning to a wide variety of hardware platforms and applications.

VEDLIoT is driven by the challenging use cases in the key sectors of automotive, industry, and smart home. In addition, within the context of the VEDLIoT project, there will be an open call to explore new opportunities by extending the application of the VEDLIoT infrastructure to a larger set of relevant and new use cases.

Finally, the whole framework is supported by cross-cutting aspects that guarantee at all levels that the systems satisfy qualities of security, privacy and trust as well as robustness and safety. In order to build systems that incorporate such qualities by design, these aspects include specialised methods for defining requirements and architectures for systems built based on VEDLIoT, for specifying components. In addition, novel mechanisms to analyse such qualities during development and continuous integration of VEDLIoT-based systems will be developed. Finally, the use of runtime monitors will allow to assess the ability to guarantee such qualities at runtime and to gracefully bring a VEDLIoT-enabled system into a safe state in case of problems.

Overall, VEDLIoT offers a framework for the Next Generation Internet (NGI) based on IoT devices capable of solving the complex deep learning applications in a collaborative manner across a distributed system that is secure and satisfies robustness guarantees. Altogether, the VEDLIoT infrastructure provides the means for advancement leading to solutions for the challenging problems in emerging use cases such as autonomous driving.

1.2 VEDLIoT WP2 Introduction

The ability to apply disruptive systems and methods such as those developed in VEDLIoT in real world applications relies on advances in development methodology. New methods for effectively describing requirements for AI-based algorithms that are distributed over IoT devices from edge to the cloud and how they relate to end-user concerns and needs are a crucial part of the solution. These methods build the foundation for specifying components of such systems in a way that enables to reason about robustness and safety as well as to enable security, privacy and trust by design.

In VEDLIoT, these cross-cutting aspects are prepared for in WP2. A high-level method to break down use cases into context description and requirements towards VEDLIoT components directly relates to two of VEDLIoTs key objectives:

- Objective 6: Security, privacy, and trust by design
- Objective 7: Robustness and (functional) safety

WP2 aims to find methods to do verifiable requirements and specifications for all levels of the system architecture for all individual use cases. This includes ways to describe sensor behaviour and data space structure in a generic way. It also includes the sensor output information format and validity as well as the variation of information quality based on sensor usage and environment. Sensors can be any type of information source both based on physical measurements and data mining.

VEDLIoT systems contain both traditional software and hardware components and AI components running on specialised AI acceleration hardware. The challenge is not only to specify and design the AI components, but also to integrate them together with the traditional components into an overall AI enabled system. In WP2, a compositional architectural framework has been developed that solves the challenge of co-design of traditional software, hardware, and AI components, while concurrently ensuring safety, security, privacy, and ethical aspects of the overall system.

The architecture definition shall describe the flow of information and description of the required refinement at the different nodes of the system. Furthermore, it shall account for possible latency and loss of data over the entire data flow network, and it shall include all data sources, the required data processing and the data output interface. This WP will also create the specifications and requirements for the selected pilots and use cases. This includes both hardware and software as well as testing and benchmarking requirements and specifications. The specific objectives for this WP are methods for:

- extracting AI architecture specifications from the requirements;
- creating sub-block design specifications, software and hardware, from architecture and requirements;
- developing performance metrics for the AI processing design;
- developing verification processes for the AI processing system;
- developing specifications, performance metrics and verification processes for defined use cases.

1.3 VEDLIoT T2.1 & T2.2 outlining

In this report, we present final results with respect to Task T2.1 and T2.2.

1.3.1 Task 2.1: Requirement methods & performance metrics (M1-M18)

- Context definition based on use case, will enable different processing requirements. It is necessary to define a context in which the system is supposed to operate and in which the system specifications will be valid;
- Functional requirements;
- Data quality requirements. The data to be processed at each node has to fulfil a defined level of quality with respect to precision, dynamic range and noise, as well as other higher-level qualities such as sensitivity, timeliness, and trustworthiness; An architectural framework that supports co-design of systems consisting of traditional software components and VEDLIoT components.

1.3.2 Task 2.2: Specification & verification methods (M1-M18)

- System and sub-block specification methods. Based on the proposed architecture specifications for hardware and software, operational feature blocks shall be identified and defined.
- Real-time performance, redundancy, security and safety levels. The maximum allowed processing output latency with respect to actual event timing needs to be defined. Also the processing robustness as defined by, among others, ISO26262 and SOTIF documents needs to be considered in the analysis methods developed in this task. The security requirements both for sensor and processing platforms as well as the transfer of data over different channels must be defined.
- Operating environment description. The final operational specification requires an operating environment description, for example the definition of an operational design domain (ODD). All the expected use cases and the corresponding environment shall be described in an ODD;
- Update and maintenance methods. This task will try to propose safe and secure ways of updating the complete system. It will also evaluate the way of maintaining the system in real time to evaluate if and when updates will be necessary;
- Verification data selection. Based on the specifications and methods developed in this WP it is necessary to define a method to select the data set that can be used as verification of the set requirements.

1.4 The VEDLIoT Thing

The IEEE 2413:2019 Standard provides an Architectural Framework for the IoT [34]. It defines *a Thing* as "an IoT component or IoT system that has functions, properties, and ways of information exchange". In addition, the standard emphasises the need for appropriate security of the information that is stored on *a Thing* or being exchanged between things.

The standard defines an *IoT environment* as "the set of IoT components available to be composed into IoT systems, the network connecting the components, and any associated services for discovery, composition, and orchestration". Combining IoT components that interact with each other through an IoT environment allows to form *IoT systems*. An example of an IoT component in a connected, typically distributed IoT system, is a *cyber-physical device*. A cyber-physical device is characterised by being able to interact with the physical world through sensors and/or actuators [27]. Other examples of IoT components are data storage devices, networking equipment, or processing nodes.

Especially the availability of efficient hardware for massively parallel processing gave a significant boost in using deep neural networks in many IoT applications, such as Smart Healthcare (Health Monitoring, Disease Analysis), Smart Home (Home Robotics, Speech Recognition), Smart Transport (Traffic Prediction, Autonomous Driving), and Smart Industry (Fault Assessment), see [60] for a comprehensive survey on Deep Learning empowered IoT applications. Compared to manual feature specification, Deep

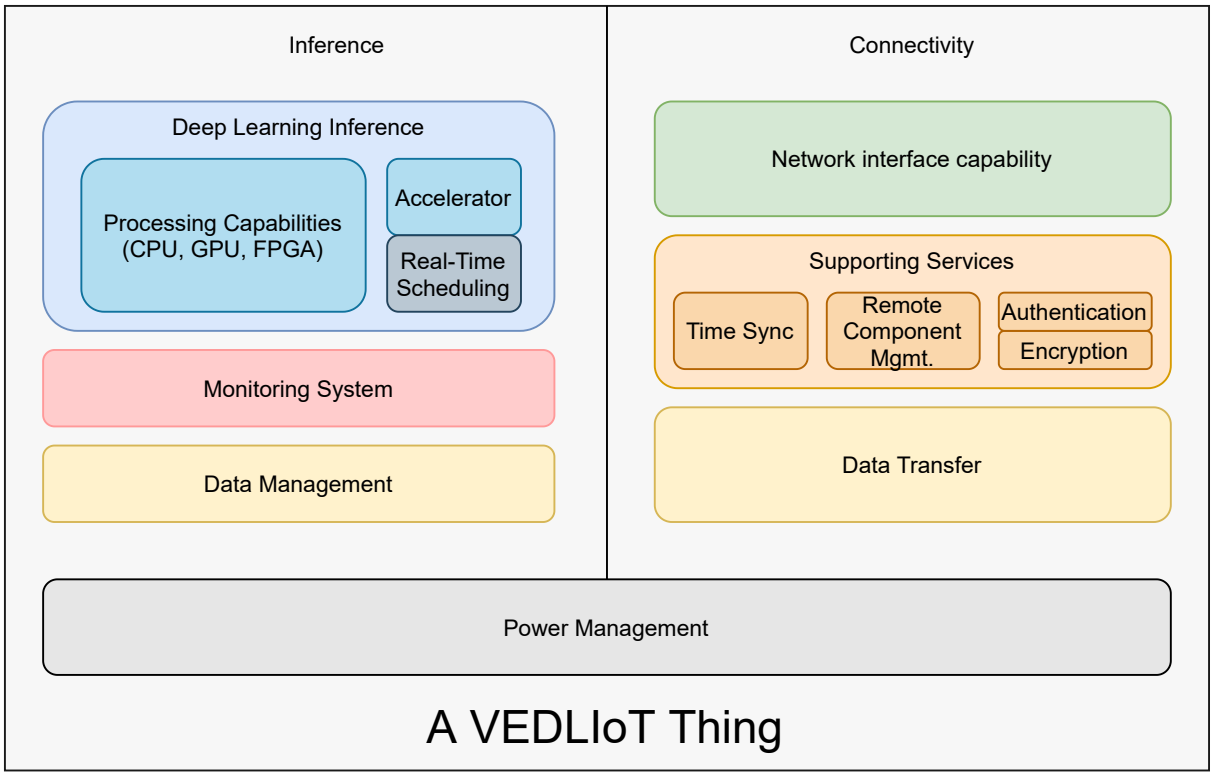

Figure 1.2. A VEDLIoT Thing

Learning allows for a significantly easier extraction of complex features from the raw data provided by sensors or other sources of data in the IoT system.

To understand the peculiarity of a VEDLIoT component in comparison to a regular IoT component, we asked the participants of a workshop meeting[2] to give free text answers (with multiple answers possible) on the question *What is a VEDLIoT Thing*? After the meeting, the answers were categorised into themes. The themes were chosen to fit best the answers given from the participants, but also to represent typical IoT characteristics. The final map is shown in Figure 1.3. While sensing and actuating can still be typical characteristics, the main characteristics are in the field of AI/Deep Learning and Data.

Answers such as *"An inference machine"*, *"A device that efficiently uses ML (Machine Learning)"*, and *"Deep Learning light-weight inference engine"* indicate that one characteristic of a VEDLIoT Thing is the ability to perform *efficient inference using deep learning*.

Efficient inference with deep neural networks can be supported with *Accelerator*, as mentioned by another participant. A VEDLIoT component is also described by a participant as *"Something that enables AI-based applications in IoT"*, which indicates that a VEDLIoT component is an extension providing AI capabilities to the IoT.

Another answer was *A VEDLIoT Thing is "[s]omething that trains a model"*. In order to be able to train a deep learning model, data must be made available. Some participants described a VEDLIoT Thing as *"Something that curates data"*, *"A Thing that collects data"*, and *"A data complexity reduction unit"*. Being able to handle potentially

[2]Work Package 2 Architecture Workshop Meeting on 21st February 2021. 20 participants represented the industry partners and academic partners in equal parts.

Table 1.1. Survey on required capabilities for a "VEDLIoT Thing". 16 participants answered the survey.
**: Score of 1 is a strong disagreement that a VEDLIoT component needs the capability, a score of 5 indicates strong agreement.*
***: Supporting capabilities include, among others, time synchronisation, data encryption, authentication, or remote component management)*
****: Latent capabilities are capabilities that lie dormant for future use. An example is a USB port, to which nothing is connected yet.*

Capabilities	Score*
Network Interface Capability	4.9
Data transferring	4.4
Data processing	4.4
Sensing	4.1
Supporting**	4.1
Data storing	2.7
Latent capabilities***	2.7
Actuating	2.3
Application Interface capability	2.3
Human UI	2

large data amounts seems to be another relevant characteristic of a VEDLIoT component.

The participants of the workshop were further asked to rank a set of typical IoT component capabilities, taken from [33], where a score of 1 means strong disagreement that a "VEDLIoT Thing" requires that capability, and 5 indicates strong agreement that a "VEDLIoT Thing" needs that particular capability. The result of the survey is given in Table 1.1. It shows that communication, data processing and data transferring capabilities are highly important to VEDLIoT, while controlling actuators and providing a human-machine-interface are less important capabilities.

Based on the results from the common workshop that took place on 21st February 2021, a common view on what a "VEDLIoT Thing" could comprise is illustrated in Figure 1.2. The figure shows two major groups: Inference and Connectivity. Inference is enabled through Deep Learning, which, based on the use case, needs certain processing capabilities, accelerators and real-time scheduling. A monitoring system ensures continuous supervision of the AI, and other components, and, because data is the core of AI, data management organises data collection, filtering, and, if required, data storage. On the connectivity part of the "VEDLIoT Thing", network interfaces ensure connectivity, data transfer controls the data flow between devices and supporting services allow for security protocols, remote management and additional services such as time synchronisation. Finally, because many VEDLIoT components will run as embedded devices, power management regulates the energy and power requirements for the device.

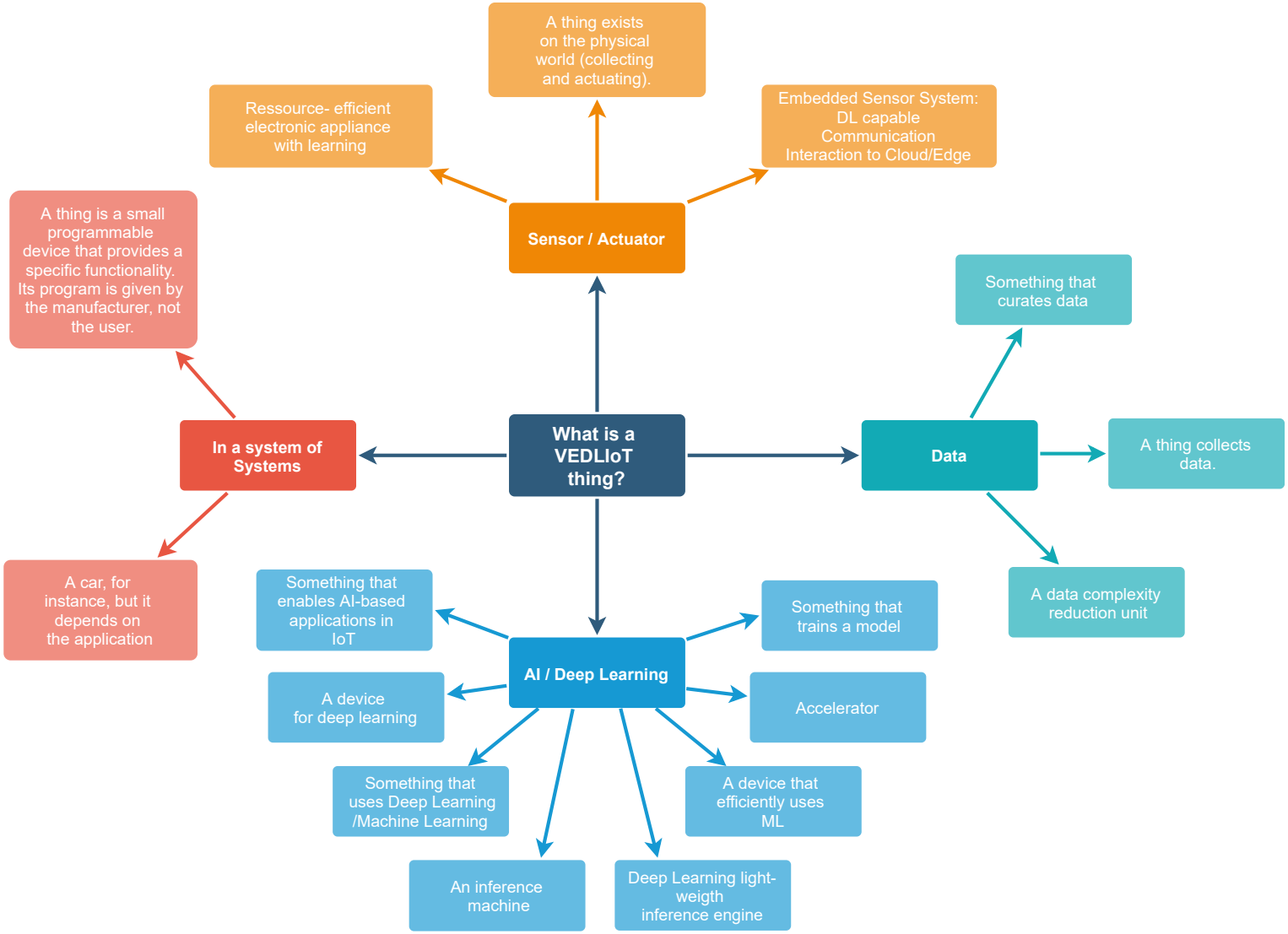

Figure 1.3. Characteristics of "a Thing" in the context of Very Efficient Deep Learning in the IoT.

2 Background

The VEDLIoT toolchain addresses the task of interfacing AI software to hardware in a comprehensive fashion leveraging and building on existing open source components. Robustness, privacy, safety, and security measures will be incorporated into the toolchain. VEDLIoT has been driven by use cases taken from industry, automotive and smart home. In addition, an open call has been initiated to apply the VEDLIoT toolchain to a larger set of relevant and new use cases. In order to build AI systems that incorporate the right robustness, privacy, safety, and security measures, specialised methods for defining requirements and architectures need to be developed for VEDLIoT. In addition, novel mechanism to analyse these qualities during development and continuous integration of VEDLIoT-based systems must be made available. This includes the use of runtime monitors which allow to assess the ability to measure and therewith guarantee such qualities at runtime. Overall requirements and the system architecture of a VEDLIoT-based system must be put in relation with specification and design decisions of its IoT components and the distributed AI algorithms.

2.1 Current best practices

2.1.1 System Architecture

2.1.1.1 Challenges with architectural frameworks for AI systems in the IoT

We applied three steps in order to derive and understand the challenges with architectural frameworks for AI systems in the IoT: In a first step, we compiled a list of relevant standards, and standard-like documents that provide a starting point for understanding architectural frameworks for the IoT and AI systems. While standards are a good representation of the state of practice for a proven technology, a literature survey helped us understand the current state of the art in research, especially in regards to systems engineering for AI. As a third step, we conducted a workshop and focus groups with practitioners in order to truly understand and validate the challenges we need to account for in an architectural framework for distributed AI systems.

2.1.1.2 Standardisation of architectural frameworks

A natural starting point when discussing system architecture is the ISO 42010 standard on architecture description [39] and ISO 15288 [41] for the life cycle management of an architectural framework. Most terms and definitions in this paper will be taken from these standards.

2.1.1.3 Standardisation of architectural frameworks for the IoT

Based on the architecture ontology and methodology of ISO 42010, the IEEE published a standard for an architectural framework for the IoT [33]. The purpose of this standard is to ”provide a framework for system designers to accelerate design, implementation, and deployment processes”. It therefore is as a key reference for the proposed architectural framework. Besides major standardisation institutions such as ISO and IEEE, there are many organisations and interest groups which provide standardisation attempts for architectural frameworks for the IoT. Good overviews of standardisation attempts for IoT architectures can be found in [92, 78], and in the architecture section of [69].

The IoT is a network of cyber-physical devices and systems, and, although it does not directly address IoT, NATOs architecture framework [14] provides an architectural framework for large distributed systems of intelligent agents. Ideas from the NATO Architecture Framework serve as input to the proposed architectural framework for distributed AI systems.

2.1.1.4 Standardisation of architectural frameworks for AI

In 2021 international standardisation for architectural frameworks of AI systems is still ongoing. The only published international standard relevant for AI systems is currently ISO/IEC TR 20547 which describes a standardisation of big data reference architectures [44]. Table 2.1 provides an overview of ongoing international standardisation efforts.

Table 2.1. List of ongoing international standardisation related to architecture frameworks for AI system

Number	Title	Status
ISO/IEC WD 5338	AI system life cycle processes	Preparatory
ISO/IEC WD 5392	Reference architecture of knowledge engineering	Preparatory
ISO/IEC AWI TR 5469	Functional safety and AI systems	Proposal
ISO/IEC DIS 23053	Framework for AI systems using machine learning	Under Approval
ISO/IEC TR 24030	Artificial intelligence - Use cases	Under publication
ISO/IEC DTR 24372	Overview of computational approaches for AI systems	In draft

2.1.1.5 State of the art for AI systems architecture

In a research agenda for engineering AI systems, the authors provide a list of challenges when developing architectures for systems with AI components [8]: Providing the right (quality of) data used for training, establishing the right learning infrastructure, building a sufficient storage and computing infrastructure and creating a suitable deployment infrastructure. The latter includes monitoring of the behaviour of the AI systems under operation, because it might only be possible to detect and correct flaws in an AI systems after deployment [5]. Furthermore, AI systems does not only consist of AI components, but relies also on conventional software and hardware components. The development of AI components and traditional system components must therefore be aligned to avoid unwanted technical debt [83]. However, as Woods emphasises, traditional architecture frameworks, such as the 4+1 architectural view model by Kruchten [55], does not account for data and algorithm concerns connected to AI component development [98]. Generally, new stakeholders (e.g.

data engineers, or governmental agency overseeing the use of AI in society [19]) and new concerns connected to AI like data quality aspects, ethical considerations such as fairness or explainability and eventually many more, need to represented through new architectural viewpoint. An example of such an additional viewpoint is a learning viewpoint governing the view on the machine learning flow [70].

Developing AI components is a hierarchical, yet also iterative task: Prepare training data / environment, create a suitable model, train and evaluate the model, tune and repeat training, and eventually finally deploy and monitor the run-time behaviour of the trained model [8, 90]. To fulfil a stakeholder's goal with a system, its design needs to be decomposed into different levels of system design, and consistency needs to be ensured in order to satisfy high-level requirements [22]. In addition, the system design must also allow for "middle-out development", where existing components need to be integrated in the overall system design (e.g. transfer-learning from existing AI models or integration of off-the-shelf components) [71]. Murugesan et al. propose a hierarchical reference model which supports the appropriate decomposition of requirements to the composition of the system's components. In their model they define how components can be decomposed into sub-components. To ensure consistency between the system architecture and the requirements, they define the terms *consistency*, *satisfaction*, and *acceptability*. One major advantage of their model is that, if decomposition of system components is done correctly, these components can be independently specified and developed.

2.1.1.6 Focus on challenges of system design for AI systems in the IoT

When combining AI with the properties of the Internet of Things, new concerns might arise that are not yet foreseen by standards and literature. To understand these concerns, we conducted in February 2021 a workshop with academic and industrial partners in the VEDLIoT project.

The aim of the workshop was to identify concerns relevant for a reference architecture concept for VEDLIoT. The workshop was conducted remotely using interactive survey elements such as Mentimeter and Microsoft Forms.

After presenting the aim of the workshop, the participants were presented with fundamental concepts of Architectural Framework for the IoT as described in IEEE 2413-2019 [33]. The participants were given Table 1 from the standard showing common stakeholders of IoT systems. They were further asked, if, when considering IoT systems with AI components, they think additional stakeholders need to be considered. The participants agreed that the list of common stakeholders from [33] contains all relevant stakeholders, and that the only additional stakeholder in regards to the AI components are legislator / policy makers who might impose additional rules, e.g. for transparency or explainability of the AI's decisions.

In a second step, the workshop participants were asked to list relevant concerns for systems that are part of the IoT and contain AI components. The list of concerns for IoT systems given in Table 2 of [33] was provided to the participants in advance of the workshop. During the workshop, the participants were asked to list all relevant concerns for IoT systems with AI components that are either on the standard's list of concerns, or are not mentioned by the standard.

The result was collected in a mind-map and, together with the participants, clustered

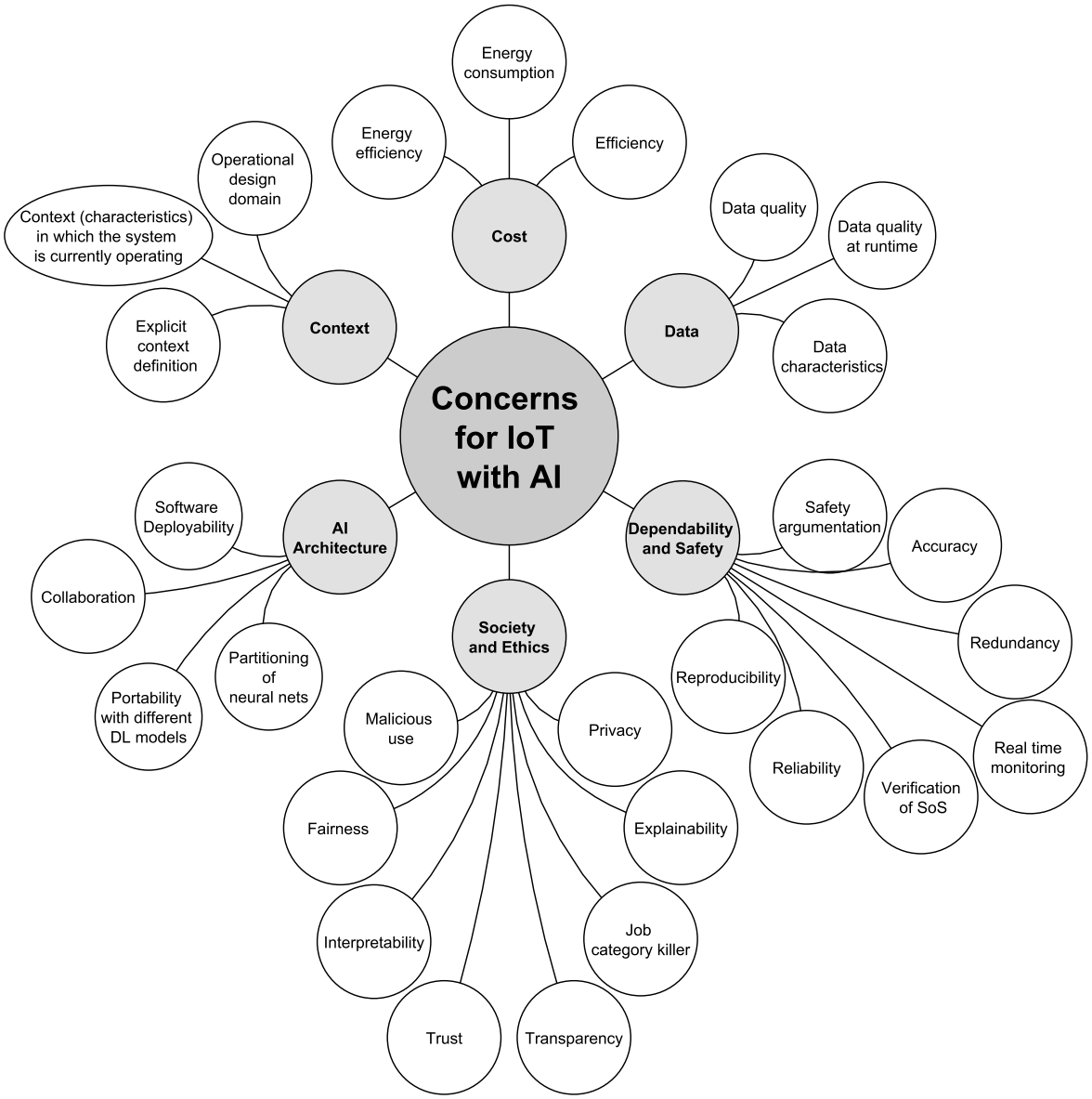

Figure 2.1. Concerns relevant for systems of the IoT with AI components

into what we will call ”clusters of concerns”. The resulting mind-map is presented in Figure 2.1.

To summarise the findings from literature and the workshop, we identified that when combining architectural aspects for the IoT and for AI systems, many new concerns arise beyond traditional software engineering, such as data quality aspects, heuristic AI modelling, AI learning, or even ethical considerations. New stakeholder such as data engineers enter the stage, and common languages or interfaces need to be found between the different stakeholders. Architectural views, governed through viewpoints, help to capture the different concerns from different stakeholders, but typical architectural frameworks, such as the ISO 42010 [39] or the IEEE 2413 [33] standard cannot cope with the large set of architectural views necessary to satisfy all stakeholders’ concerns. One major challenge we identified in the workshop is the difficulty to keep track of dependencies, e.g. through correspondence rules, between the different architectural views. Another problem of current architectural frame-

Table 2.2. Participating partners in workshop on an architectural framework for VEDLIoT

No	Name	Industry Partner	Academic Partner
1	Industrial IoT	✓	
2	Smart Home		✓
3	Automotive Systems	✓	
4	DL Optimisation	✓	
5	AI Hardware	✓	
6	Requirement Engineering		✓
7	IoT and AI research		✓
8	AI systems development	✓	
9	Security concepts for IoT and AI		✓
10	AI Hardware Research		✓
11	Systems Safety Concepts		✓

works is the lack of a clear system development hierarchy, which would support the early identification and mapping of dependencies between different architectural views [72].

2.1.1.7 Terms and definitions

For this report, we will refer to a terminology which follows the ISO standard on architecture description for system and software engineering [46] and the IEEE standard for an architectural framework for the internet of Things (IoT) [34]:

Architecture specific

architecture: ”Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution” [39];

architecture description: ”Work product used to express an architecture” [39];

architecture framework: ”Conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders” [39];

architecture view: ”Work product expressing the architecture of a system from the perspective of specific system concerns” [39];

architecture viewpoint: ”Work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns” [39];

concern: ”Interest in a system relevant to one or more of its stakeholders” [39];

interface: ”Shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate” [35];

model kind: ”Conventions for a type of modelling” [39];

performance: ”Quantitative or qualitative level of a property at any point in time considered” [33];

privacy: ”Right of individuals to control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed” [33];

Reference model: ”A reference model is an abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details. A reference model may be used as a basis for education and explaining standards to non-specialists” [33];

reliability: ”The ability of an item to perform a required function under given conditions for a given time interval” [37];

resilience: ”Ability of a system or component to maintain an acceptable level of service in the face of disruption” [33];

safety: ”The condition of the system operating without causing unacceptable risk of physical injury or damage to the health of people, either directly, or indirectly as a result of damage to property or to the environment” [43];

security: ”A condition that results from the establishment and maintenance of protective measures that enable an enterprise to perform its mission or critical functions despite risks posed by threats to its use of information systems. Protective measures may involve a combination of deterrence, avoidance, prevention, detection, recovery, and correction that should form part of the enterprise’s risk management approach” [40];

service: ”The means by which the needs of a consumer are brought together with the capabilities of a provider” [33];

stakeholder: ”An individual, team, or organisation (or classes thereof) with interests in, or concerns relative to, a system” [39];

technical artifact: ”Entity that is designed by humans and that has a composition. The composition of the technical artifact enables functions that are accessible to humans and other technical artifacts” [17].

IoT specific

application: "Functional unit that is specific to the solution of a problem. Note that an application may be distributed among resources, and may communicate with other applications" [38];

coexistence: "Ability of two or more devices to operate independently of one another in the same network respecting the common rules for sharing the same medium" [38];

confidentiality: "Property that information is not made available or disclosed to unauthorised individuals, entities, or processes." [42]

data: "Representation of facts, concepts, or instructions in a formalised manner suitable for communication, interpretation, or processing by human beings or by automatic means" [33];

data type: "A set of values together with a set of permitted operations" [33];

device: "Independent physical entity capable of performing one or more specified functions in a particular context and delimited by its interfaces" [36];

entity: "A particular thing, such as a person, place, process, object, concept, association, or event" [36];

physical entity: "A physical entity is a discrete, identifiable part of the physical environment that is of interest to the user for the completion of her goal. Physical entities can be almost any physical object or environment; from humans or animals to cars; from store or logistics chain items to computers; from electronic appliances to closed or open environments" [33];

system: "Set of interrelated elements considered in a defined context as a whole and separated from its environment" [17]

thing: "Any physical entity in combination with its digital representation" [33]

virtual entity: "Computational or data element representing a physical entity" [33]

AI specific

AI-based system: A system consisting of various software and potentially other components out of which at least one is an AI component [65];

AI component: "A part of a system. The component uses primarily AI" [65];

AI library: "A special AI component that provides a concrete implementation of AI algorithms" [65].

2.1.1.8 AI taxonomy

This reports follows the taxonomy proposed in [65] for AI as shown in Figure 2.2

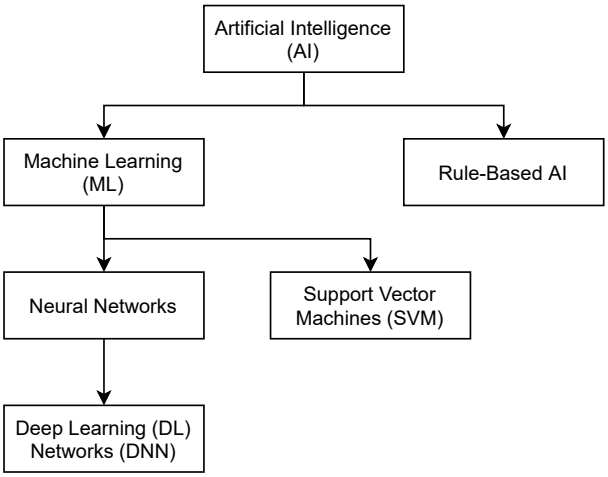

Figure 2.2. Taxonomy for AI based on [65]

2.2 Identified problems with current best practices

From the use cases, we derive four problem areas that demand for further research. Figure 2.3 illustrates that all four problems are interconnected with each other.

PA 1: Contextual definition and requirements; The ability of ensuring a desired system behaviour requires that the context, in which the machine learning model is deployed, is clearly defined.

PA 2: Data attributes and requirements; The context, and especially the ability to guarantee certain system quality attributes such as safety and robustness, on the other hand will impose non-functional requirements which will lead to requirements of the data in use.

PA 3: Performance metrics, reproducibility, comparability and real-time monitoring of trained machine learning models; To achieve continuous improvement of the system and to enable the system to react on situations where functionality and quality attributes can no longer be satisfied, performance monitoring and reporting needs to be established in the given context.

PA 4: Human Factors; For the success of the system, human factors must be considered - will humans accept decisions of the automated system? Will they react accordingly? Will this affect the performance of the system in use?

Figure 2.3 illustrates that all four problems are interconnected with each other. In addition, a cross-cutting aspect relates to process support for modern system development approaches and the success of solutions within each problem area depend on good integration into engineering practice.

Given the dependencies between the problem areas, we focus on the first two problem areas, because good conceptual approaches for managing context and data requirements will support identifying solutions in the other problem areas. The following research question will guide us through our work on requirement methods for the VEDLIoT project. It is an initial collection of relevant research questions, and not necessarily all of them will be answered in this project.

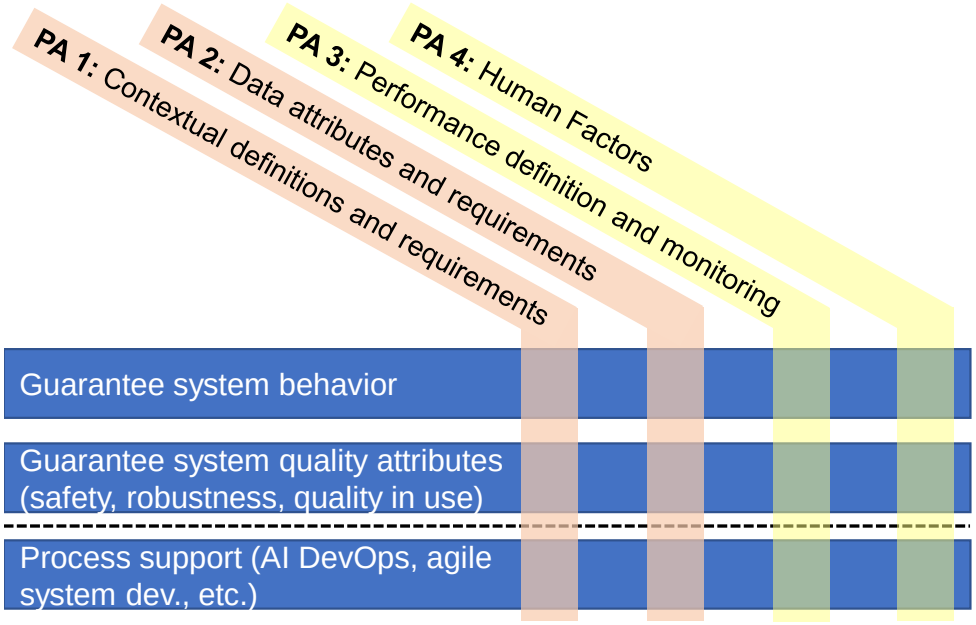

Figure 2.3. The development of complex, AI systems implies the need of certain abilities (blue boxes) that depend on solutions for challenges in four areas (red/yellow boxes). We argue that one has to find solutions for the red challenge areas before approaching the yellow challenge area.

2.2.1 PA 1: Contextual definitions and requirements

In the automotive use case, the problem of contextual definitions is probably most obvious: Today, an ADAS cannot operate in any given driving situation. The system is designed and tested to perform safely only in a priori defined conditions. According to SAE [81], these operating conditions define the operational design domain (ODD). A significant problem for the development of more automated vehicles is a lack of a common definition for the ODD of a vehicle [54, 28]. Still, the original equipment manufacturer (OEM) must be able to guarantee a certain system behaviour and especially safety attributes.

If deep learning is used to enable complex object (people, obstacles) and pattern (road markings) detection, the ODD, as a way to describe the context in which the vehicle will operate, will govern the required performance of the trained machine learning model. The problem of contextual definition can even be abstracted beyond the automotive use case to other applications of machine learning:

A trained machine learning model cannot be placed into another context without appropriate new training and testing. The context can also change slowly over time. To preserve the ability of ensuring the system's behaviour, significant more transparency about the entire life-cycle of a machine learning model will be required. It must be shown that the context, in which a machine learning model operates, is suitable. A starting point towards more transparency for machine learning models are model cards [67].

The importance of proper context definition for machine learning becomes apparent in the *no-free-lunch-theorems* [97]. In brief they state that over *all* data-generating distributions, every machine learning algorithm will perform equally poor when confronted with previously unobserved data. It is necessary to make assumptions on the

probability distributions that the trained model is expected to encounter in the application. Those assumptions, or beliefs, can be explicit by directly stating assumed probability distributions over parameters of the model. They can also be implicit by choosing learning algorithm that are biased towards choosing some class of function over another [25]. A link obviously exists between the context of an application and the expected data attributes.

Research Questions: Contextual definitions

RQ 1-1: What challenges arise when deriving contextual definitions and requirements from use cases?

RQ 1-2: Which practices would be appropriate for deriving contextual definitions and requirements?

RQ 1-3: How to express and document explicit or implicit beliefs based on the derived contextual definitions?

Research roadmap: A first step to answering the research questions is a qualitative exploratory study. A deeper understanding of the problem in a realistic environment can be gained with data collected in interviews and focus groups. The main goal of the interviews is to get a better understanding of the challenges in defining the context for applying machine learning. The interview partners will be function developers and experts from the OEM domain, a Tier 1, or a Tier 2 automotive supplier.

A thematic analysis of the collected data will be performed to create a suitable model by interconnection [15]. In a focus group the findings of the study shall be validated and evaluated by the interview participants and other relevant stakeholders.

2.2.2 PA 2: Data requirements and quality attributes of data

Data, and especially their representation in the form of probability distributions are the core of machine learning. Different types of data (input data, training data, test data, etc.) play a role when deploying and using machine learning or deep learning. Each type can even further be categorised: For an autonomous driving system there could be driver data, vehicle data, surround data, and global data for example. The other use cases have similar data categories such as user data, sensor platform data, or cloud data.

The contexts in which machine learning algorithms are used govern properties of the data used in design time (e.g. during the training and testing) and during runtime. Furthermore, data often originate from many different sources with varying degree of quality, which can be a challenge for the ability of ensuring the system's behaviour in a given context. Especially if system quality attributes, such as safety or robustness, must be ensured, it is crucial that the used data are trustworthy, timely, and of the required quality. An AI will be trained with data representing a context in which the system is expected to operate. However, if the context changes over time, properties of the input data to the AI change as well [93]. Based on the context, there will be requirements on the input data in order to allow the AI to arrive at the right decision. An example for a data requirement could be, that the data shall represent a given probability distribution for which the AI has been trained. Only then can a machine learning model arrive at the right decision. During operation, the system might also record data that allows developers to continuously receive feedback and to implement improvements in the overall system, e.g. through retraining to correct

for a (slowly) changing context.

Although data are probably the most important aspect of a machine learning application, there is no proper system to determine and manage the required quality and quantity of the data. Not only since the introduction of more rigid data privacy rules, such as GDPR, there is a growing pushback to the idea to ”collect as much data as possible” for a machine learning application in the hope that the right data might be among them.

Research Questions: Data attributes and requirements

RQ 2-1: What are relevant challenges of managing data quality requirements when developing large distributed systems based on deep learning?

RQ 2-2: What constitutes a data quality framework for developing large systems based on distributed deep learning?

RQ 2-3: To what extend can relevant challenges of managing data quality requirements be mitigated by a data quality framework for developing systems based on deep learning?

Research roadmap: To find answers to the proposed research questions, we utilised a design science research methodology. From the automotive use case, typical data parameters such as precision, timeliness, dynamic range, and noise were collected from sensor specifications and data examples. Combined with interviews and workshops with ADAS and deep learning experts, and following the methodology for design science studies [74, 53], the principal goal of the research was to devise a solution for understanding data quality requirements and dependencies between data types in distributed systems using machine learning models.

Literature references

Previous research on data quality in software engineering and data quality frameworks served as a starting point:

- The significance of data quality in design, validation and implementation of software [6].
- A proposed data quality framework for distributed computing environment [20].
- The effects of data quality on machine learning algorithms [84].
- A data quality assessment and monitoring framework [3].
- Characteristics and challenges of big data environments [9].
- Reporting mechanisms of data quality in distributed networks [49].
- Requirements for data quality metrics [32].

2.2.3 PA 3: Performance definition and monitoring

The next step after having established a framework for defining the context of the machine learning application, and required data attributes, was the definition of performance metrics, or key performance indicators (KPI), and subsequently the setup of monitoring regimes.

Performance definitions and monitoring of machine learning enabled systems allow to check that the system stays within its guaranteed system behaviour. In addition, it supports development processes by providing developers with feedback on how the deployed machine learning model performs ”in the field”. Only a continuous mon-

itoring of the system allows for continuous integration and control of the machine learning model.

While it is meanwhile common practise to use machine learning models for fault detection, there are only very few efforts to use fault detection in machine learning models. How can we ensure that a machine learning model is fault-free during its operation? The questions on how faults e.g. in a deep neural network can be classified and detected are not answered yet. Beyond fault detection, fault isolation and fault handling can help ensuring safety goals for systems with AI.

A concrete research roadmap will be derived based on the finding of the research on contextual definition and data attributes, but some general research questions about performance monitoring of machine learning models are:

Research Questions: Performance definition and monitoring

RQ 3-1: What are suitable performance metrics / KPIs for trained machine learning models in a given context?

RQ 3-2: What approaches are possible to compare / compete different trained machine learning models for a given context?

RQ 3-3: What faults can occur in a machine learning model? Can faults in machine learning models be classified in different categories?

The performance requirements set during the requirement engineering define which metrics need to be used. Machine Learning requires probabilistic thinking, such that requirements such as ”The system shall always detect an obstacle ahead” cannot be validated. The term ”always” suggests that the system shall detect an obstacle in 100% of all cases, which is unfeasible with a probabilistic system. Instead, proper metrics, such as uncertainty measures suggested in [91] need to be introduced and the requirements need to be adopted accordingly. Section 4.2.3 provides an overview of how requirements can be adopted to probabilistic systems.

2.2.4 PA 4: Human Factors

Literature references

Previous research on human factors related to automatic vehicles and AI systems:

- Lee et al. highlight the danger if human factors are not sufficiently considered during AV design, related to achieving sufficient safety, trust and acceptance as well to avoid the miss-use and disuse of the automated technology [57].
- Hancock’s warning that human factors must be integrated in automation design [31].
- A list of socio-technical challenges [30].
- A methodology based on multidisciplinary cognitive engineering (CE+) [88].
- The User Centered Ecological Interface Design (UCEID) that enables including HF considerations in the early stages of the overall system design processes [79].

When building complex AI systems and products, it is important to complement a focus on internal, technical aspects of the system (e.g. the conditions and capabilities of the system in a given context) with a focus on how the intended users will interact with it in a realistic context. For this focus, ergonomics and human factors must be taken into account. Understanding human factors is particularly important for build-

ing a system that users accept and trust in. To achieve the desired results, it is critical to consider human factors right when the concepts are developed, i.e. as part of requirement engineering.

However, it is challenging to integrate human factors into the development process of complex AI systems, such as automated vehicles. One reason for this is the need to shorten the time-to-market when developing new features. Hence development teams focus more on technical parts and may not possess enough human factors competence to design them according to the users' needs.

Since many developing organisations are transitioning to agile or continuous system development and reject the idea of comprehensive upfront requirements, development teams cannot fall back to requirements specifications for the purpose of including human factor constraints in their design decisions.

New advanced and AI-enabled safety features such as for example automated emergency braking (AEB) have changed the interaction of human drivers with the their vehicle significantly. This frees up mental resources and improves the quality of driving but also affects other traffic participants and their behaviour. While AI-enabled support for driving tasks have dramatically changed in just a few years, humans have not changed in the past millennium. So, while, designing such functionalities, we need to keep in mind some key elements (limitations and capabilities) from the perspective of humans and specifically of users. For example, the fact that humans override or deactivate AEB functionality has become a major limitation in its ability to make traffic safer [62]. Such scenarios must be analysed from a human factor perspective. Thus, we suggest to investigate the extent to which human factors must be considered when analysing required functionality and quality of the system and its components, in particular in relation to modern system development approaches.

VEDLIoT is concerned with a technological platform for AI-intense systems and allows us to investigate appropriate decomposition of requirements and architecture. Human factors are not a key concern in VEDLIoT, but our ambition is to link VEDLIoT to other EU research projects that complement the aspects of human factors, such as SHAPE-IT[1]. SHAPE-IT focuses on the transportation domain, but specifically focuses on the interaction of AI-intense automatic vehicles (AV) with users and other traffic participants.

Research Questions: Human Factors

RQ 4-1: In what way must human factors be considered for understanding and ensuring system behaviour of AI systems?

RQ 4-2: In what way must human factors be considered for understanding and ensuring system quality attributes of AI systems?

RQ 4-3: How can Human Factors knowledge be effectively used in modern system development approaches?

2.2.5 Cross-Cutting research problem: Integration in modern system development

In systems development, there is a general trend towards agile, DevOps, and continuous deployment, since such approaches promise shorter time-to-market and in-

[1] https://www.shape-it.eu

creased responsiveness to change [26]. To achieve these goals, organisations rely on empowered, self-organised teams that take responsibility for features from inception, over design, implementation, and test, to deployment [52]. Ideally, such empowered teams allow for fast responses to change, since teams can make decisions directly. In order to facilitate such quick decisions for AI systems, and to prepare for scalability, the responsibility for any activities related to our problem areas should lie with the teams to the largest possible extent. In order to achieve this, the organisation must provide sufficient support:

- Requirements information model for context, data requirements, performance metrics, and human factors.
- Traceability information model that allows to identify and resolve inter-team dependencies
- Methodological support to generate and manage knowledge about context, data requirements, performance metrics, and human factors

Literature references

Previous research on integrating requirements management into modern system development approaches as a starting point:

- The state of the art in these areas with respect to machine learning has recently found unsatisfactory, especially in automotive use cases where ISO26262 does not well match the nature of deep neural networks [7].
- An overview of RE-related challenges in scaled-agile system development [50].
- A discussion of requirements information models that support collaboration across organisational boundaries [94].
- Considerations of challenges related to defining traceability strategies [61] and traceability information models for modern system development approaches [64, 95, 11, 13].
- Discussions of RE practices at scale [94] and boundary objects for requirements-based coordination [51, 96].

Research Questions: Integration in modern system development

RQ X-1: What are characteristics of suitable requirements information models?

RQ X-2: What are characteristics of suitable traceability strategies?

RQ X-3: What are characteristics of suitable methodological support?

3 Architectural Framework

The main goal is to introduce an architecture framework based on compositional thinking that is suitable for the development of distributed AI-based systems in the VEDLIoT project.

A major challenge in AI system design is the lack of design patterns, standards, and reference architectures that support the co-design of traditional software components and AI components [65]. When designing a system, a range of quality aspects, such as safety, security, and privacy needs to be taken into account. For AI systems, ethical aspects such as explainability of decisions, fairness, and participation play an important role during the system design process. Therefore, the architectural framework for VEDLIoT shall not only support the seamless design and integration of traditional software components and AI components, but also allow for all necessary quality concerns to be taken into account as early as possible in the design process.

3.1 Architecture descriptions for distributed systems

ISO 42010 provides a conceptual model of an architecture description as depicted in Figure 3.1. The idea of an architectural framework is to provide a knowledge structure that allows the division of an architectural description into different architectural views [73]. An architectural view expresses "the architecture of a system from the perspective of specific system concern" [39]. The conventions of how an architectural view is constructed and interpreted is given through an corresponding architectural viewpoint. The design of a system-of-interest needs to account for different concerns of different stakeholders. Therefore, the architecture of the system-of-interest must be expressed through many different architectural views. Several views on the architecture of the system-of-interest allow for factoring the design task into smaller and specialised tasks.

Designing a large, distributed system is a hierarchical process [71]. Therefore, for a given concern, there exist several views at different levels of abstraction. The hierarchical design process allows for the co-evolution of requirements and architecture, known as the "twin peaks of requirements and architecture" [72, 12].

Zardini et al. show how applied category theory supports the co-design of hardware and software for an autonomous driving system [99], and Bakirtzis et al. apply compositional thinking to engineering of cyber-physical systems [2]. It seems natural to apply compositional thinking to the evolution of system architectures by establishing suitable descriptions of the abstractions levels for the architectural views, their classification into clusters of concern, and the relation between the views.

3.2 Clusters of concern

In bi-weekly meetings throughout the first half of the year 2021, both industrial and academic partners of the VEDLIoT project analysed the use cases and applied the compositional architectural framework idea. The entire version history of the VEDLIoT architectural framework can be found in the supplement material[1].

[1] Available on the VEDLIoT cloud server `https://cloud.vedliot.eu/f/14483`

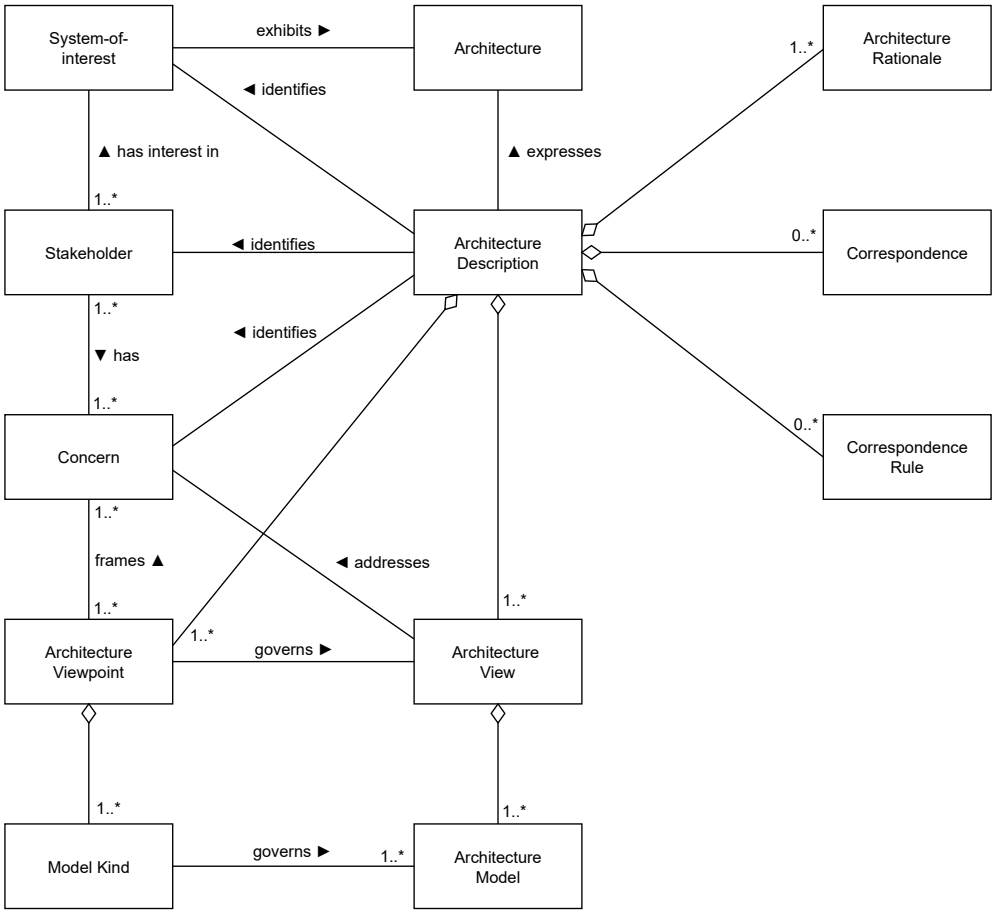

Figure 3.1. Conceptual model of an architecture description [39]

Clusters of concerns are determined through the identified use cases based on the operational context and high-level goals (in Figure 4.1 referred to as functional goals and quality goals) for the desired AI system. For example, privacy might not be of concern for an AI based diagnostic system detecting faults of a welding robot, but safety could be of paramount concern.

The results, illustrated in Figure 2.1, from a project workshop were used as a starting point for discussions on the necessary clusters of concern during bi-weekly meetings of the VEDLIoT partners.

In a first step, the group identified four major groups of concerns to emerge for the architecture framework:

Behaviour and Context contains aspects that concern the static and dynamic behaviour of the system, as well as the context and constraints for the desired behaviour.

Means and Resources contains aspects of the system that enable the desired behaviour. Obviously, this includes the required hardware that executes the desired behaviour. For AI systems, two more concerns play a significant role as ”means to execute a desired behaviour”: Choosing the right machine learning model, or deep learning model, is one of them. Classification of objects in visual images would require a different deep neural network setup than identify-

ing spoken words for natural language processing. The second concern which is characteristic to AI systems is the learning setting: Through a learning process, the AI model is trained to imitate a desired behaviour. Planning and preparing the learning of the AI model therefore becomes a ”mean to execute a desired behaviour” within an AI system. Learning can be conducted through preparing training datasets, or, in the case of reinforcement learning, could be done in a simulated environment.

Communication deals with aspects of data, connectivity and communication between nodes or components of the desired system, which is one major concern when developing *distributed* systems, such as automotive systems, or systems in the IoT.

Quality concerns basically encompasses all quality aspects described through quality attributes which can affect the architecture of the system. Examples are safety, security, privacy, robustness and ethical concerns. The latter can include aspects such as fairness and explainability. Recent legislation shows that the ethical aspects become a central concern when developing AI systems [19].

Then, in a second step, for each of the groups of concerns, the participants of the bi-weekly meetings analysed the use cases of VEDLIoT and determined the required clusters of concerns.

3.2.1 Behaviour and Context

To describe an architecture reflecting the desired behaviour of the system, two clusters of concern are introduced: **Logical Behaviour** covers views that are concerned with the static behaviour of the system, and **Process Behaviour** covers views concerned with the dynamic behaviour of the system. The **Context and Constraints** cluster of concern covers views on the system that define the context and limits the design domain. The latter cluster of concern is typically not explicitly mentioned in architectural frameworks, instead implicitly included in e.g. views of the behaviour of the system. However, for AI systems it is beneficial, sometimes even required, to explicitly state the desired context and to define views on the constraints and the design domain of the system. An example is the Operational Design Domain of automated vehicles as discussed in [28]. A challenge for the application of Deep Learning are changing contexts. It is common to collect the necessary training and testing data for deep learning networks through some form of big data pipeline. However, Jameel et al. highlight that one cannot assume stationary of that data, and over time properties of the data assemble might change, which requires regular retraining of the ML models based on that data [47]. A usual assumption for the training and testing of ML models is that the data are drawn from stationary probability distributions [58]. This assumption, however, might not hold in reality, i.e. the context might slowly change over time, and the probability distribution of the input data changes therefore slowly with time as well. Without retraining or adaptation, this change of context can render the ML model inadequate over time. There is initial efforts to detected drift in deep learning performance due to context changes, for example [18], or run-time uncertainty detection [91].

3.2.2 Means and Resources

The concerns in this group include views that allow to describe the resources and means available for the system to execute the desired behaviour in a given context. Typically, it includes views on the hardware architecture and component design of the system under the cluster of concern **Hardware**. Additionally, two AI related clusters of concern have been identified: First, the concern **AI models** contains views that describe the setup and configuration of the required AI model. For example, classification of objects in an optical videostream requires a different deep neural network configuration then recognising commands in a voice recording or predicting trajectories of other vehicles in the vicinity. Choosing the right AI model setup is a system design decision which requires suitable views on the AI model in relation to the overall system. Furthermore, the learning strategy of the AI model has paramount impact on the final behaviour of the AI system. Trained with the flawed datasets (e.g. bias present in the data), the behaviour of the AI system will exhibit the flaws learned during the learning process (e.g. it will show a bias in the decisions). The learning process for the AI model is part of the system design process, and therefore the cluster of concern titled **Learning** covers views on the system that allow to define and setup the learning environment for the AI model. The concerns of AI model and Learning have many dependencies between each other, which will be expressed through correspondences.

3.2.2.1 Communication

Communication is what drives the IoT. Two clusters of concerns have been identified: First, **Information** accumulates views on the system that model the information and data exchanged in and through the system-of-interest. Second, the cluster of concern **Connectivity** contains views on the means of communication available to the system and its resources.

3.2.3 Quality Concerns

This group contains concerns that influence the desired quality of the system. The cluster of concern **Safety** provides an example here: Assume one is to follow the workflow of ISO 26262 [43]. The starting point to designing a safe system is to identify, safety goals that the architecture, as part of the functionality providing item, needs to fulfil. This is often done through a Hazard Identification and Risk Assessment (HARA), which provides abstract information applicable on the entire system. On the next lower level of abstraction, the functional safety concept provides a view on a more detailed system architecture that introduces functional safety requirements and redundancies (through safety decomposition in hardware and software components) with the aim to assure the fulfilment of the earlier specified safety goals. On the next more detailed level, the technical safety concept provides information on the technical realisation of the functional safety concept. In addition, and not explicitly mentioned in ISO 26262, we propose that the run-time behaviour and monitoring is part in the system design process. For safety concerns, this could mean the introduction of safety degradation concepts and safety monitoring.

Further identified relevant clusters of concerns for quality aspects of an AI system in the IoT are security, privacy and ethical aspects such as Fairness and Transparency. For embedded system, energy efficiency can be taken up as explicit quality aspect covered by an own cluster of concern. Unlike previous architectural frameworks for

the IoT, such as [33], the compositional thinking in the architectural framework allows for co-designing the system to fulfil the explicitly identified quality concerns. It means that already early in the system development, correspondences between the views regarding the quality concerns and other views in the architecture description are established. The final system can then be said to be ”Safe by design”, ”Secure by design”, ”Efficient by design”, or ”Fair by design”.

3.2.4 Architectural views for AI systems

Many of the concerns that form clusters of concern in the architectural framework of VEDLIoT are concerns commonly found in systems engineering. They have been mentioned in other architectural standards before, such as the IEEE Standard for an Architectural Framework for the Internet of Things [33]. Therefore, the architectural viewpoints corresponding to these concerns will not be detailed in this article. The argument behind that is, that systems using the VEDLIoT toolchain will contain a significant amount of “traditional” system components around the AI components in order to facilitate the desired behaviour. The architectural viewpoints used to describe traditional system views are well covered in literature and regarded state of the art.

However, several architectural viewpoints are novel and aim to facilitate the design of AI components for the system. Table 3.1 provides a list of viewpoints, which govern architectural views in the architecture framework for VEDLIoT, that we assume to be novel and relevant specifically towards the AI components of the system.

3.2.5 Example of correspondences between a quality concern and the remaining architecture concerns

Assume a system that shall trigger the brakes when an object is detected in front of the vehicle. Assume further, that, through the HARA, the following safety goal has been identified: *”The system shall not trigger the emergency brake unintentionally (ASIL[2] B)”*.

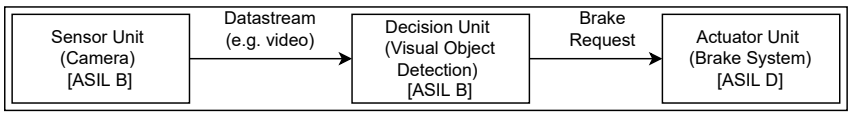

Figure 3.2. Conceptual system architecture for an automotive automatic emergency brake system

A preliminary high-level system architecture, as illustrated in Figure 3.2 consists of a sensing element (camera), a decision unit (object detection algorithm on an embedded platform), and an actuator (brakes). While the brakes are typically designed to a high safety integrity level (typically up to ASIL D), the camera and object detection algorithm might not be able to achieve even ASIL B. Therefore, we introduce a safety decomposition in the functional safety concept through redundancy in the sensing system: With an additional lidar sensor, together with a second object detection algorithm specifically designed for detecting objects in lidar point clouds and independent from the first object detection algorithm allows to reduce the required safety integrity level of all redundant components to ASIL A(B), which might be easier feasible to implement. The final high-level system architecture after safety decomposition is illustrated in Figure 3.3. By introducing the safety decomposition in the functional

[2] Automotive Safety Integrity Level

Table 3.1. Description of AI specific architectural views.

Cluster of Concern	Viewpoint Name	Description	Example
Learning	Learning objectives	High-level architecture that outlines which functions / models shall be trained through learning.	One function is depicted that shall be able to recognise obstacles on the road from camera images.
	Learning concept	Describes how the learning objective shall be achieved. This can include a description on how a simulation environment is set up, which datasets are required, if transfer learning should be used, properties of the datasets, etc.	A dataset containing a variety of obstacles typically found on Swedish roads (note that limiting the dataset to Swedish roads will introduce a morphism / correspondence to context definition), specifying the data attributes (e.g. resolution, input format, etc.), and desired quality attributes.
	Learning settings	Detailed description of the learning regime's configuration and setup.	Settings for learning, e.g. metric to use, stop condition, loss function, optimiser, batch size, data split for training and testing.
	Runtime data collection & continuous learning	Concept for collection new training data at run time. Concept for continuous re-training of AI.	Unknown obstacles shall be stored and transmitted to manufacturer to be included in re-training of AI.
AI Model	High-level AI model	High-level architecture showing which software components contain an AI model. Also outlines, which kind of AI is intended to be used.	A deep neural network is assigned to the function "Object detection".
	AI model concept	Setup of the AI model outlining the architecture of the AI.	For object detection using a sequential deep neural network with a number of 2D-convolution layers, a 2D-pooling layer, dropout, flatten, and a dense layer. Input, and output definition.
	AI model configuration	Detailed description of the AI model configuration.	For a deep neural network this can be kernel size of convolution layers, dropout rate, activation function, etc.
	AI model performance monitoring	Concept for monitoring the performance of the AI.	For example recording of inference time and uncertainty monitoring.
Context and Constraints	Context assumptions	Assumption on the context (this can be the environment, involved users, etc.).	Object detection shall operate at all times on Swedish roads.
	Context definition	Detailed description of the context in which the desired behaviour will be executed.	Swedish highways and motorways. All possible weather conditions in Sweden. Behaviour should be stable at all times of the day and night.
	Constraints / Design Domain	Lists constraints, and design domain, in which the desired behaviour will be expected.	Speed is above 30 km/h, but less than 130 km/h. Object to detect is not more than 1 metre above the road. At night, headlights illuminate road at least 20 metres ahead.
	Context monitoring	Concept for monitoring that the AI operates in the desired context of operation.	System monitors the geographical location of the vehicle (e.g. through GPS) and prevents activation of system if outside of desired context (e.g. Swedish highways and motorways).

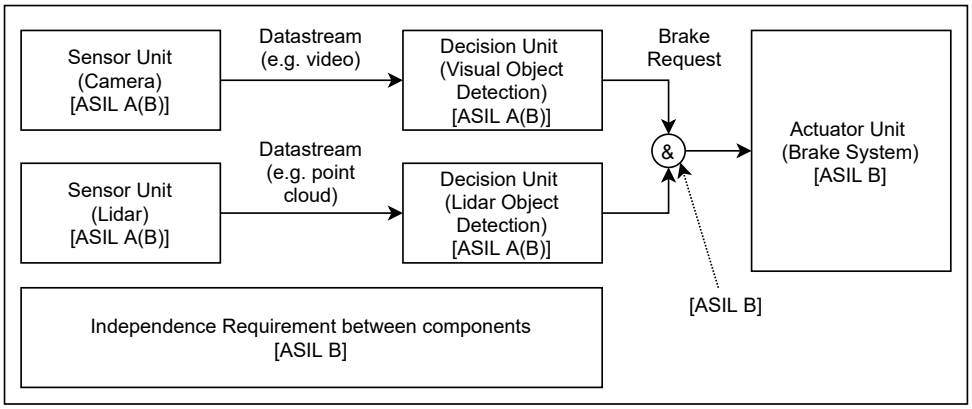

Figure 3.3. Conceptual system architecture for an automotive automatic emergency brake system after safety decomposition

safety concept, correspondences to the system hardware architecture view and the logical components view were established in order to fulfil the required safety concerns. On the next level of abstraction, the technical safety concept establishes a view on the overall system's architecture that allows the fulfilment of the functional safety concept. For example, the technical safety concept provides a view on the system architecture that requires the logical component "Visual object detection" to be deployed on safety certified hardware components, which creates a correspondence to the computing resource allocation view; or that the object detection algorithm only works at daylight, which creates a correspondence to the constraints / design domain view.

3.2.6 Example of correspondences between context, learning, and AI model

The second example provided in Figure 3.4 illustrates the parallel evolution of the context and constraints of the system, the learning concept, and the AI model. This example is especially interesting in that it demonstrates how the development of an AI component can be seen as a hierarchical process which needs to stay in synchronisation with the remaining concerns of the system development.

On the highest level of abstraction the context assumptions take direct influence on the required learning objectives (i.e. there exists a morphism from the assumption ("Pedestrians can either be in lane, or on the road but not in the lane, or the road is empty" to the learning object of classification of objects into three classes). One should only proceed to the next level of abstraction in the architectural framework, if the context assumption, the learning objective, the high-level AI model, and the remaining system views on the analytical level are consistent, i.e. no morphisms are broken between the views.

On the conceptual level, the context definition clarifies the earlier context assumptions, which provides input into the required data for training and testing of the AI model. By establishing morphisms from the system hardware architecture view, the context definition can take limitation of the hardware into account, e.g. mounting position of the camera requires a human to be at least 1.00 metre tall. Furthermore, the AI model can be conceptualised, because input data, required output data and objective of the AI model are known.

On the design level, the design domain provides clear constraints for the system's

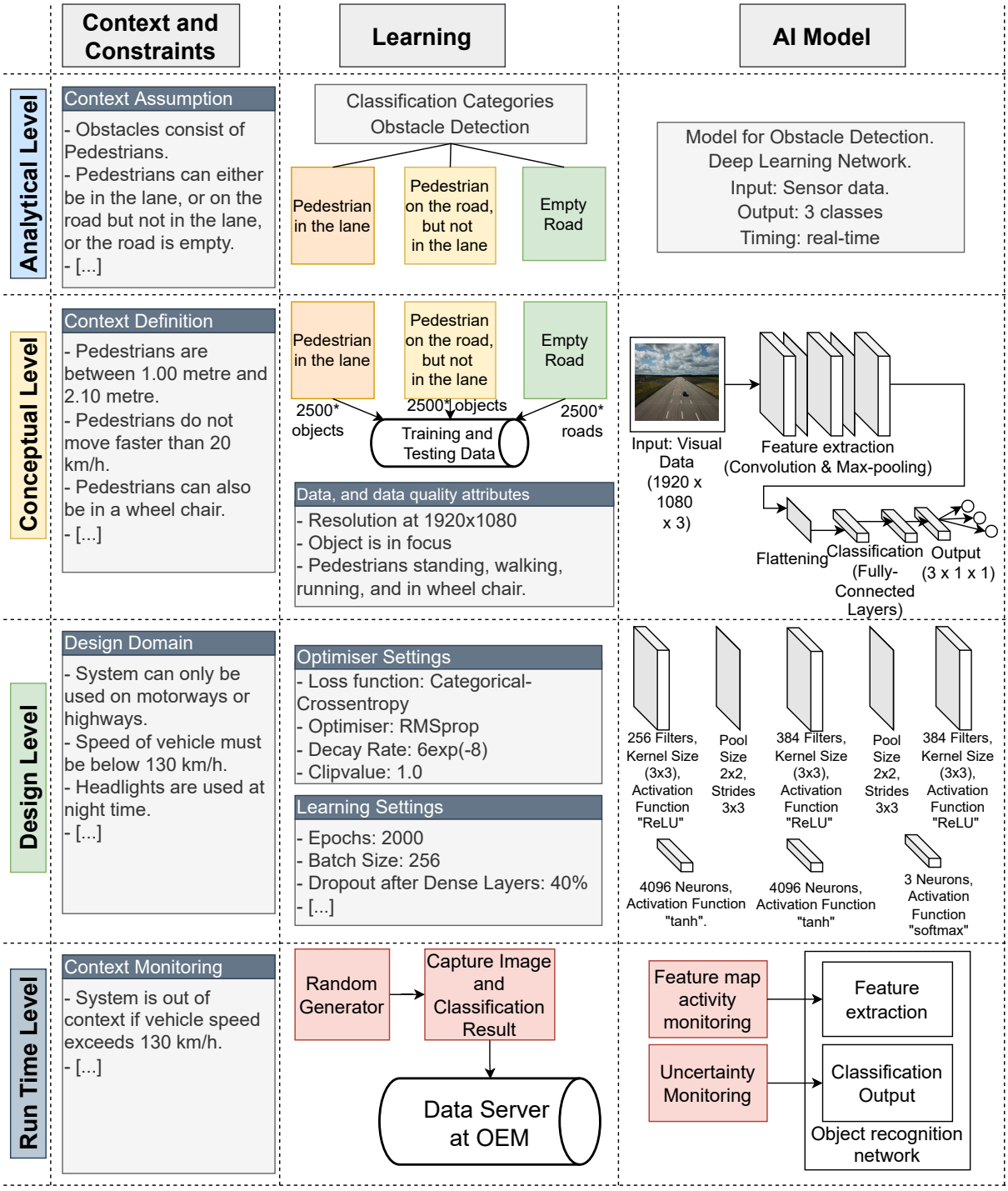

Figure 3.4. Co-Design of context and constraints, learning concept, and AI model.

operability, and the learning procedure's and AI model's configuration are set. Finally, the run time level provides views that explain which monitoring concepts and run time reconfiguration might be required during operation of the system. One example is the AI model run time view: Two monitors can check the feature map activity in the feature extraction section of the deep neural network, and uncertainty monitoring can be applied to the classification output of the network.

3.3 Levels of abstraction

The architectural views are not only sorted by clusters of concerns as discussed previously, but also by their represented level of abstraction. For VEDLIoT it was decided to follow four levels of abstraction, specifically *Knowledge and Analytical level*, *Conceptual level*,*Design level*, and *Run time level*:

3.3.1 Knowledge and analytical level

The first level of abstraction includes architectural views that provide an abstract and high-level view on the system-of-interest. On that level, all views provide a way to describe the system and context on a knowledge level, which provides information for further, more concrete system development. For example, the high-level AI model view could elaborate on which functions should be fulfilled through an AI.

3.3.2 Conceptual level

On the next level of abstraction, the views provide a more concrete description of the overall system-of-interest. Components are not detailed yet, but the overall system composition becomes clear and the context of operation is clearly defined. For example, the AI model could be concretely shaped as a Deep Learning Network with a required amount of layers. All views on this level combined provide a system specification that sets the system-of-interest in context and elaborates on how the desired functionality is fulfilled.

3.3.3 Design level

The most concrete level at design time of the system is the design level, which includes views that concretely shape the final system-of-interest. Resources are allocated to components, the AI model is configured to work most efficiently in the given environment, and the concrete component hardware architecture is defined. The solution specification describes the final embodiment of the system-of-interest.

3.3.4 Run time level

Complex systems, both AI driven and conventional, often require forms of monitoring and operations control. The purpose of the run time monitoring can be manifold: On one hand, monitoring of a deployed system at run time provides valuable feedback about its performance and reliability to developers and product owners. DevOps is an essential component of an agile development framework, and early detection of issues in a deployed system allows for a swift response from the developers. Furthermore, some requirements of the system might not be exhaustively testable before deployment of the final system. This is especially the case for AI systems: Russel describes in his book *Human Compatible: AI and the Problem of Control* the example of an AI algorithm commonly found in social media that maximises *click-through*, i.e. "the probability that the user clicks on the presented items" [80]. Russel highlights

that the problem with such an algorithm is, that it not necessarily “presents items that the user likes to click on”, but instead could (inadvertently) change the user’s behaviour in a manner to make him or her more predictable in his preferences e.g. by favouring extreme political views [80]. It was probably not the intended behaviour to change the user’s preferences. Most probably, we will not be able to design the perfect ”beneficial” AI in the near future, and therefore we have to anticipate undesired behaviours of deployed AI algorithms. By constantly monitoring the decisions of the AI algorithm, such deviations from the intended behaviour can be detected and mitigated, e.g. through retraining or by “pulling the plug”. Most AI systems are not “adaptive”. They are trained and tested with a dataset representing the desired context in which the AI system is intended to operate in under the assumption of stationarity in the probability distribution of the data. In reality, the assumption of stationarity of the probability distributions does not hold in most case, for example when the context, in which the AI operates in, can change over time. Concepts like *continual learning* allow the AI to handle drifts in data distributions [58]. However, continual learning requires run time monitoring concepts to detect deviations from the currently learned context, and automatic data collection (and labelling) for autonomous retraining of the AI model. These aspects of changes in run time behaviour are described on the run time level of abstraction in the compositional architectural framework.

3.4 Compositional architecture framework for VEDLIoT

The final conceptual model of a compositional architecture framework based on the stated propositions is illustrated in Figure 3.5. Figure 3.7 presents a compositional architectural framework that includes all earlier identified concerns for distributed AI systems and all levels of abstractions for VEDLIoT. The architectural framework is a composition of a number of clusters of concern, highlighting that it is *”the combining of distinct parts or elements to form a whole”* and *”the manner in which such parts are combined or related”*[3]. Therefore, we referred to it as ”compositional architectural framework for VEDLIoT”.

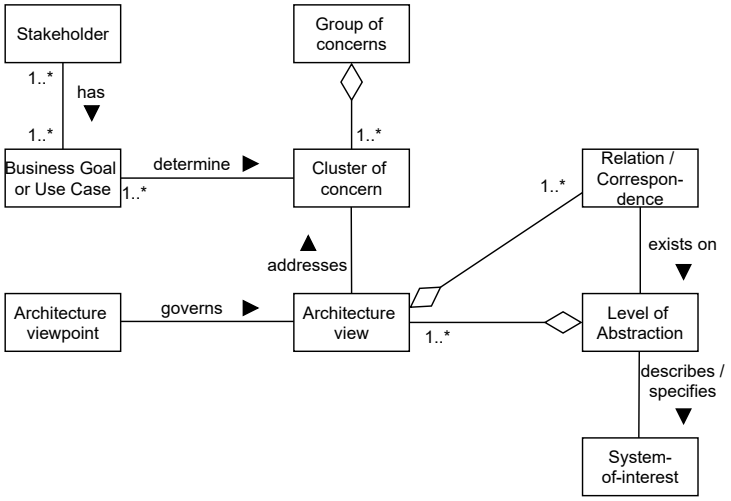

Figure 3.5. Conceptual model of a compositional architecture framework

[3]https://www.thefreedictionary.com/compositional

3.4.1 How to apply a compositional architectural framework in practice

Based on the experience of applying a compositional architectural framework to VEDLIoT, the following guideline can be provided:

Step 1: Identify clusters of concern.

Clusters of concerns are identified. Initially, larger groups of concerns (such as functionality, hardware, communication, quality) can be defined, which are then refined into atomic clusters of concerns.

Step 2: Identify levels of abstraction.

Levels of abstractions are identified. The number of required levels depend on the size and complexity of system-of-interest and the development settings of the company. Three to four different levels of abstraction seem a good default.

Step 3: Add existing architectural decisions.

Known architectural decisions are entered into the matrix. Most development projects do not start from scratch, but instead have to reuse or integrate into existing architectures. Prior knowledge, such as an existing component architecture, can be entered into the appropriate clusters of concerns and level of abstraction in the architecture matrix.

Step 4: Add missing architectural views.

Architectural views are added. Relations (morphisms) are created between the architectural views at each level of abstractions such that no inconsistencies occur when looking at the system-of-interest from different architectural views.

Step 5: Add missing relations.

All relations between architectural views must be mapped onto corresponding views of the next lower level of abstraction. If a relation between two architectural views on a higher level of abstraction does not have a correspondence on the next lower level of abstraction, the relation might be unnecessary and can be removed, or a corresponding relation needs to be created.

Step 6: Iterate if needed.

During the system development, additional clusters of concern might be discovered that iteratively are added.

Steps 1-5 are illustrated in Figure 3.6.

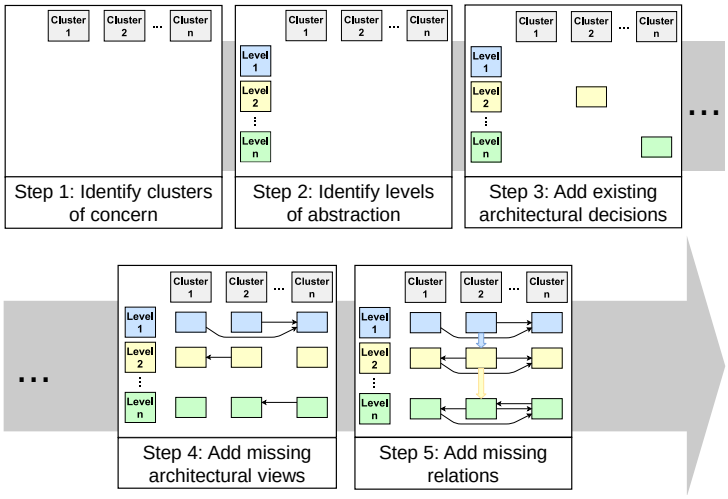

Figure 3.6. Steps taken for defining a compositional architectural framework for VEDLIoT

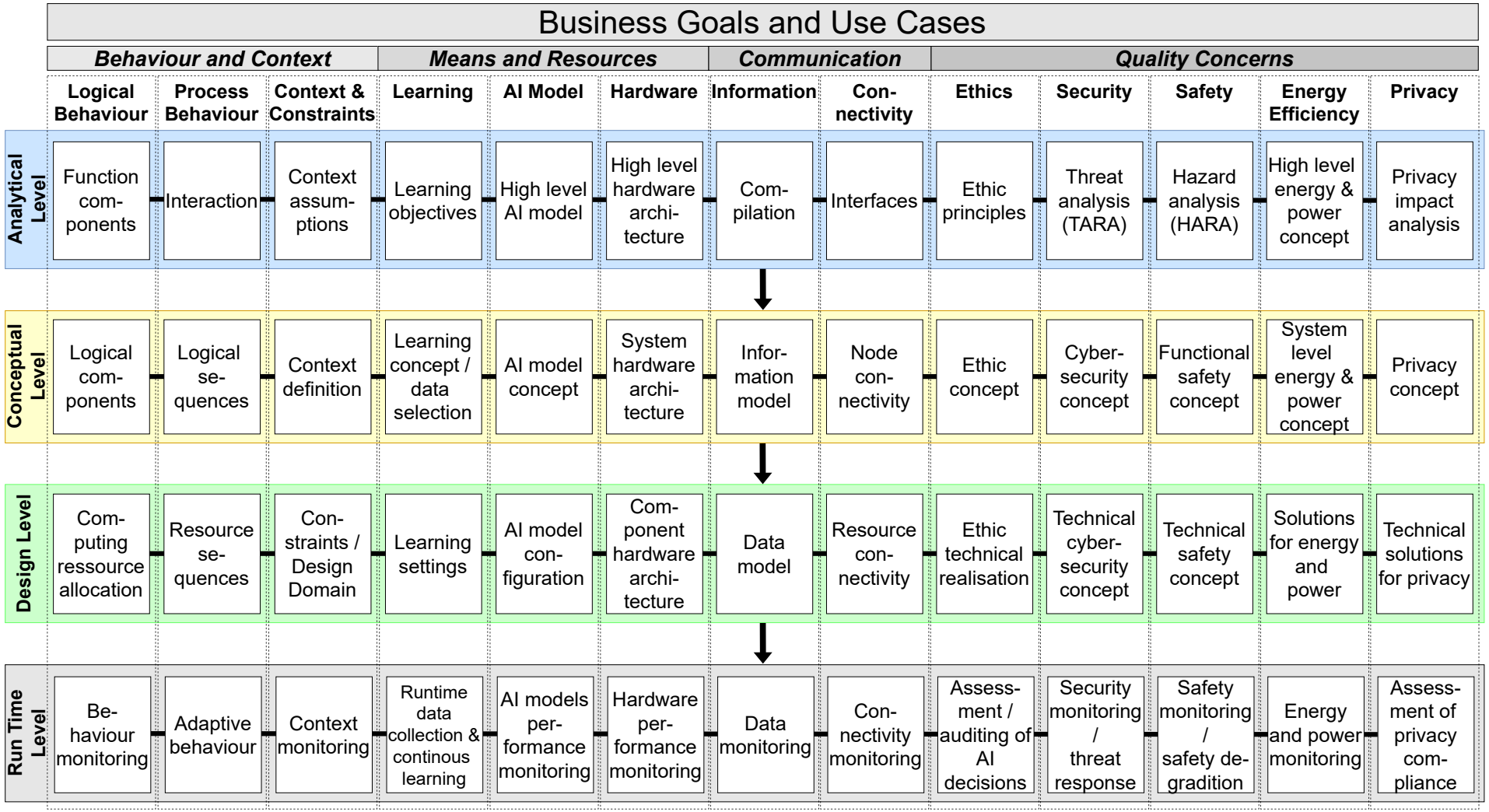

Figure 3.7. Compositional architecture framework for VEDLIoT, categorising views in different clusters of concerns on different levels of abstraction.

3.5 Monitoring and controlling concepts for the run-time behaviour of advanced AI systems

The main purpose of VEDLIoT is to allow for advanced AI systems empowered through deep neural networks and deep learning. Unlike rule-based AI, deep learning networks cannot be programmed using fixed objectives and rules. Instead, the neural network finds the rules based on presented training data and returns probabilities and error bounds when performing inference with previously unseen data.

For traditional software systems, the desired objectives of the system can a priori be well-defined in requirements and specifications and excessively tested and validated against the earlier specified stakeholders' preferences. The system then behaves static (except if dynamic aspects are explicitly included and specified in the system) and monitoring of these systems serves primarily to report on misbehaviour of the system caused by bugs or other mistakes during the design process. For probabilistic AI systems the assumption that all desired objectives can be defined a priori of system design does not hold. A probabilistic system, such as a system based on deep learning, does not follow a-priori defined rules. Instead, it tries to find rules (probability distributions) based on the data it is presented with. The challenge is, that during the design process it cannot exhaustively be ensured that the deep neural network indeed found the rule that matches the stakeholders' preferences with the training data it was presented with. The challenge becomes even more critical when the probabilistic system is not static in its behaviour (meaning it is trained once, and the trained behaviour is not changed at run-time), but instead can adapt and learn at run time autonomously based on the data the system encounters while operating. To ensure the desired behaviour of the overall AI system, the VEDLIoT architectural framework explicitly contains the previous mentioned run time level that contains views on how the system can be monitored and controlled at run time. Two crucial views are behaviour and context monitoring. The behaviour monitoring ensures that human stakeholders see and understand how the AI system behaves at run time. Furthermore, VEDLIoT systems will be designed and trained for a specific operational context. If the AI system leaves the specified operational context, the behaviour will most likely not follow the desired behaviour anymore. The system could react on a context change by adapting its behaviour (e.g. safe stop in an autonomous vehicle), or by employing continuous learning to adapt the behaviour to the new context. Allowing the AI system to autonomously adapt its behaviour based on continuous (reinforcement) learning is not without problems, because it could lead to violation of desired quality concerns. Therefore, all quality concerns (ethics, security, safety, energy, privacy, and more if needed) have their own monitoring and control concept that need to be fulfilled at run time.

4 Requirement Methods

This chapter discusses requirements methods for VEDLIoT based systems. The requirements method complements the architectural framework and focuses on activities and tools (templates, workshop guides) to elicit the required information.

- The architectural framework provides a refinement structure
  - The columns of the framework correspond to concerns that a VEDLIoT-based system either has or has not
  - The rows of the framework correspond to abstraction levels, from high-level on top to more concrete and specific on the bottom
- The requirements engineering method complements this:
  - *Business goals and Use Cases:* On a high-level, the scope and high-level goals about functionality and quality are described. This allows to reason about which concerns (columns in the architectural framework) are necessary and it also provides input towards the architectural views on the knowledge and analytical layer
  - *System and Context Description:* Based on the views on the knowledge and analytical layer, the system and its context can be described. This allows to externalise assumptions, and consider alternatives in which way the system can address business goals and use cases best. This provides input to the conceptual layer of views in the architectural framework.
  - *System Specification:* Based on the views on the conceptual layer, concrete requirements for the components of the system can be derived. In this report, we focus on functional requirements and quality attributes relevant for VEDLIoT components.

Note that while this report organises the content in a logical top-down fashion, the architectural framework and requirements engineering method in VEDLIoT assumes that knowledge can become available on all levels at any time. Thus, the focus is on describing how the description can be made complete and how the missing views on a conceptual or analytical level can be provided based on an existing design level view.

The following sections reason about *Business goals and Use Cases* and *System and Context Description*, while we discuss *System Specification* in Chapter 5 separately. Table 4.1 provides an overview of notations and practices that support each level and how it relates to the architectural framework.

4.1 Define Scope, High-level Goals about Functionality and Quality, Prioritise Concerns

We perceive the development of a VEDLIoT based system as a complex, multi-organisational, and multi-disciplinary endeavour. At its core is a simple design cycle:

Table 4.1. Overview of notations and practices that support each level of requirements and how it relates to the architectural framework.

	Functional requirements	Quality attributes	Operational context	Relation to architectural framework
High-level requirements (corresponds to highest level, business goals and use cases)	Business cases, scope, user stories, use cases	Quality grid	No best practice known	Inform selection of concerns and content of views on analytical layer
Stakeholder requirements	User stories, use cases, task descriptions	Quality scenarios	No best practice known	Provide detail about success criteria from stakeholder's point of view
system requirements	Feature requirements, (structural) data requirements	Planguage, testable quality requirements, open metric, open target, quality requirements on data	ODD and context description	High-level description on how system will address requirements based on views on analytical layer; informs conceptual layer views
Design requirements	Allocation of functionality to individual components	Quality contracts and constraints	Specification on how context can be assessed by system	Specification for VEDLIoT components based on design time views
Runtime requirements	Requirements monitors	Requirements monitors	Context monitors	

1. Understand the problem to be solved
2. Design and implement a solution
3. Verify design and implementation
4. Validate ability to solve problem

We recommend to anticipate these phases early on and to plan for them.

Figure 4.1 shows a high-level overview of this multi-organisational process of building VEDLIoT components for use in a VEDLIoT based system. At the beginning, goals with respect to functionality and quality must be defined for a certain operational context (often described as operational design Domain (ODD) in the automotive context). This must be done in sufficient detail so that high-level architectural decisions

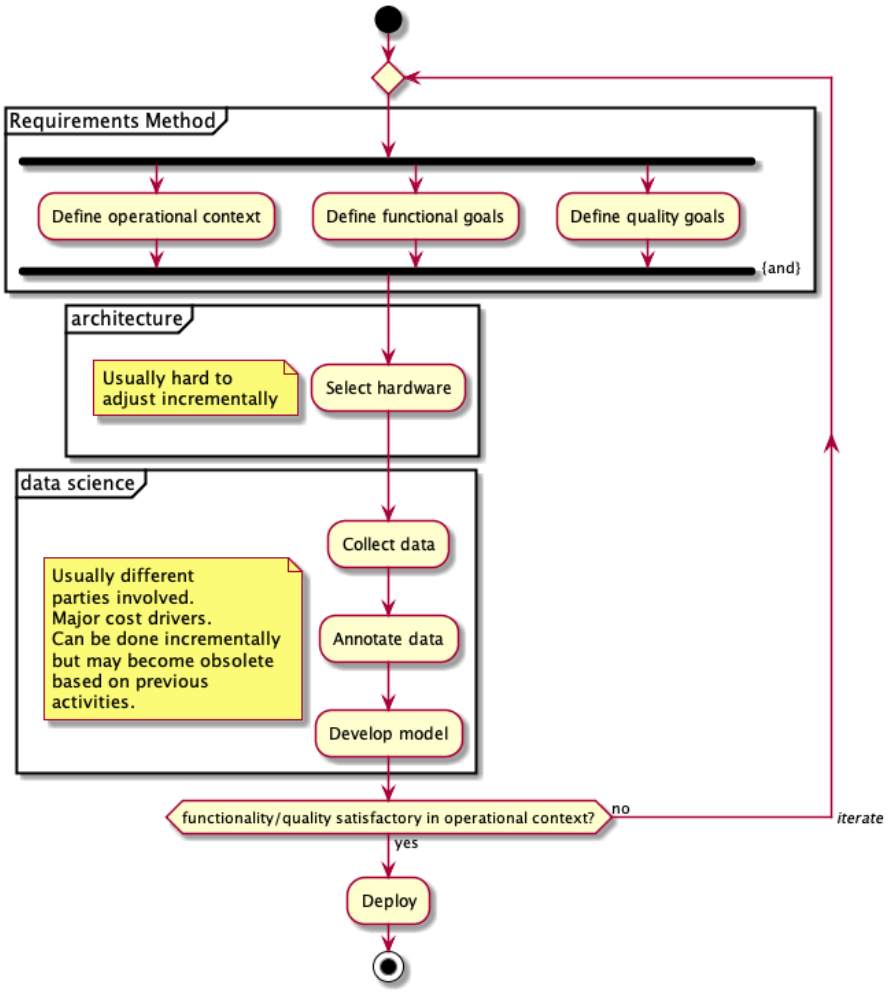

Figure 4.1. Basic flow of building VEDLIoT-based systems

can be made, including the selection of hardware and specifically the sensors. Since the interplay of hardware components and the placement of sensors in the overall system can affect data quality, it is important to collect data with the selected hardware. This data then needs to be annotated (at least in parts) and will then be used to develop and train the model. Only at this point, it can be evaluated whether the functional and quality goals can be reached in the operational context.

If the goals can be met with the current setup, the VEDLIoT components can be deployed into the VEDLIoT-based system.

If not, an iteration can be necessary, in which all or a subset of previous decisions can be adjusted. Often, it will be unfeasible to adjust functional or quality goals. The operational context can however often be constrained further, to reduce the amount of annotated data needed for model development.

A change in hardware and sensors may be necessary, but will result in high additional cost. To mitigate this, in VEDLIoT we utilise reconfigurable hardware, which can be adapted to changing requirements or environmental conditions at design time or even at run time. This however creates new challenges to practices of documenting architecture and requirements.

Annotation of data can be done incrementally and strategic approaches exist to opti-

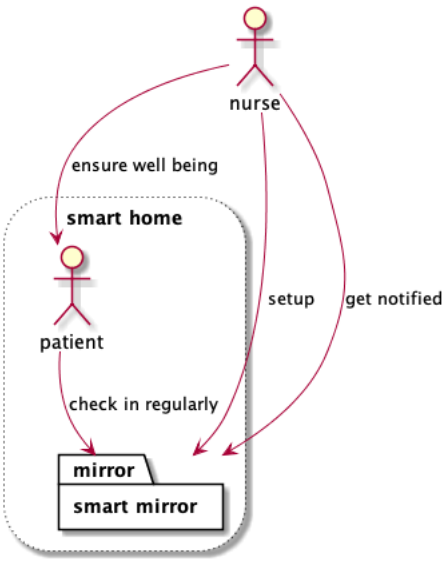

Figure 4.2. Context diagram, describing scope and high-level functionality of a potential smart mirror based system

mise the information gain per effort. However, if quality goals or operational context change, existing annotations may become obsolete, which is a cost factor to consider. It can be useful to have a strategy for partial annotation in an attempt to prove feasibility of the current setup.

4.1.1 Define Scope

Laueson suggests the use of context diagrams to clearly indicate the scope of a system [56]. In such a diagram, key stakeholders and their interaction within a given domain is shown. Laueson in particular emphasises the need to distinguish between domain events (e.g. a person attempting to achieve something within a smart home) and system events (e.g. a person interacting with the smart mirror to trigger a certain action). The outcome of this activity is a list of domain and system events (or: goals/tasks to achieve) and an initial characterisation of the operational context and scope.

The example in Figure 4.2 illustrates this. The overall goal for the nurse is to ensure the well-being of a patient living in a smart home. The domain level requirements for the smart mirror are therefore to send a notification to the nurse if the patient misses a regular check-in. The ability to support regular check-ins by the patient as well as specific setups by the nurse are then product-level requirements. This (fictive) example shows the importance to align on the basic goals, events, and requirements in a defined domain, when reasoning about a VEDLIoT component. In case of the smart mirror, a discussion can now take place that relates to the ability to recognise a particular patient and his/her well being, to the trade-off between ensuring the patient's privacy and the patient's well-being, and so on.

Clearly, this is not a sufficient description of operational context. It is a starting point for further refinement. In VEDLIoT, we argue that this refinement requires to iterate between problem and solution space. The analytical layer in the architectural framework allows to refine the knowledge about context and scope from a solution space perspective.

4.1.2 Functional goals

Functional requirements describe the behavioural aspects of a system, including the inputs to the system, the outputs of the system, and behavioural relationships. The domain and product events in the context diagram translate to high-level functional goals. It can be a real enabler for efficient development work, if these can also be traced to business goals, so that business value can be taken into account in prioritisation and design decisions. The focus in this report is however on the further refinement. User stories can be a good lightweight way to capture and discuss functional goals from a stakeholder's perspective. The typical template of a user story is as follows:

> As a <role> I want <feature> so that <value>

This is a great way to capture stakeholder requirements, especially, if the value for each stakeholder can be captured in a good way. Complex goals can also be captured as use cases. A common critique with use cases is that they often mix problem and solution descriptions. A good alternative can be task descriptions, that focus more on the tasks that must be supported and less on how to support them. Regardless of the concrete format (use case or task description), it is a best practice to make any ideas for technical solutions explicit. For use cases in the VEDLIoT context, we suggest the template in Table 4.2 and to embed the use cases with references to context and data definitions as suggested in Figure 4.3, which are based on works currently in review at a journal [77].

The template in particular emphasises the importance to capture the relation to any concern, including context and constrains of the use case as well as needs with respect to data and system quality.

In terms of input and output, it is common to describe data requirements as part of the functional requirements. On a high-level, we recommend to describe basic data structures and protocols. For the success of a VEDLIoT-based system, the quality of data must also be described, for which there is a lack of support in requirements engineering methods.

4.1.3 Quality goals

Since functional requirements cover only behavioural concerns of the system, other concerns must be handled separately. While often described as non-functional requirements, this term has been found inaccurate. Glinz suggests to instead distinguish quality attributes (such as performance requirements and specific quality requirements) and constraints [24]. The usual approach towards defining quality attributes is to start from a taxonomy of quality attributes.

Our investigation towards the architectural framework and its concerns offers a more targeted starting point for potentially relevant quality attributes.

Quality attributes are notoriously difficult to handle. Key challenges include the effort involved in making them testable. Thus, on a high-level of abstraction, it is most important to gain an overview of the relevant quality attributes, to prioritise them, and to identify a way to refine them into a testable state that allows to clearly state whether they are or are not fulfilled.

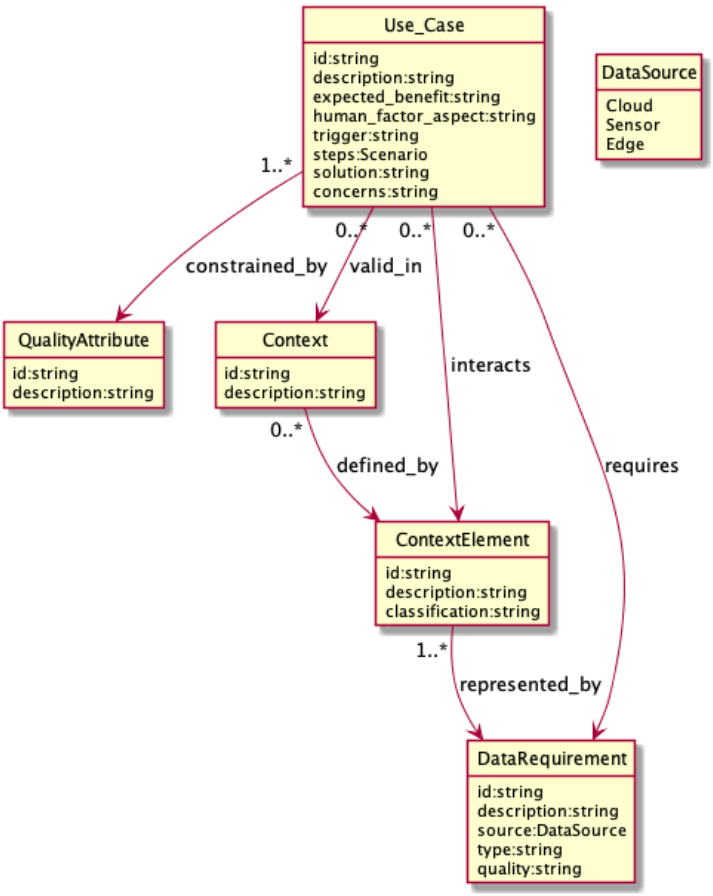

Figure 4.3. High-level requirements information model that connects use cases to operational context (Operational Design Domain, ODD), Data Requirements, and Quality Requirements.

Lauesen suggests the use of a quality grid [56]. The quality grid is essentially a table with relevant quality attributes in the first column and then an indication on whether this attribute is more or less important than in comparable products. Usually, a simple check mark or footnote would show the criticality, followed by a short explanation on why a concern is more or less important than usual. In Table 4.3, we provide an assessment for the three main use cases in the VEDLIoT project.

Reasons for prioritising one concern over the other can include:

- Market position: With maturing products, functionality starts to disappear as a distinction factor. Quality, in relation to competing products, becomes a key business driver.
- Domain needs: Regulation or standardisation, as well as the criticality towards safety or security of a product are drivers for the importance of certain quality attributes.

For a VEDLIoT-based system and its VEDLIoT components, we believe that the prioritisation is critical. It allows to select the minimal set of concerns to capture in the architecture (when following the architectural framework proposed in Chapter 3). The architectural framework provides guidance in how to explore each quality. We will describe in Section 4.2.3, how quality requirements can be described in a useful and testable way.

Table 4.2. Proposed use case template. Use underlined text to relate to other elements in the requirements information model (see Figure 4.3).

Use case ID	<ID> <the name should be the goal as a short active verb phrase>
Description	<a one or two line description of the goal>
Expected benefit	<list of all the possible benefits and areas the use case can be helpful>
Human factor aspect	<if applicable: describe human capabilities, limitations, objectives, and support to be provided; default: not applicable>
Valid in context	<A description of the context and constraints in which this use case is valid, e.g. as post-condition, precondition, or invariant. It is recommended to rely on ODD elements from the ODD ontology in [16].>
Trigger	<the action that starts the use case>
Steps	1. <steps of the scenario from trigger to goal delivery and any cleanup after> 2. <...>
Existing or proposed solution	<list of systems that already implement the use case; list proposed solutions to keep rest of use case technology neutral>
Concerns/constraints	<list of important quality attributes or other VEDLIoT concerns that apply to this use case>
Data requirement	<list of data required, data qualities (e.g. volume in bytes per hour), source and direction of data transfer>

4.2 Define System and Context Description

Based on the analytical layer in the architectural framework, a description of the system and its context can be specified. The goal is to derive and organise requirements that relate to the individual views in the architectural framework and to create a solid foundation for the architectural views on the conceptual layer. In this context, it is important to provide the following information:

1. Define a system context
2. Define system requirements
3. Define quality requirements
4. Derive quality requirements for data and learning

The following sections give details on each of these topics.

Table 4.3. Adopting the quality grid to VEDLIoT based systems.

Quality factors for VEDLIoT-based system	Critical	Important	As usual	Unimportant	Ignore
Ethics		UC1.a, UC1.b, UC2, UC3[1]			
Security	UC1.a, UC1.b	UC3	UC2		
Safety	UC2[2], UC1.b		UC1.a	UC3	
Privacy	UC3[3]		UC2		UC1.a, UC1.b
Efficiency	UC1.a[1]	UC2, UC3	UC1.b		
…					
Legend	UC1.a: Industrial Use Case Motor Monitoring UC1.b: Industrial Use Case Switchboard Monitoring UC2: Automotive Use Case UC3: Smart Home Use Case				
Comments	[1] Potential ethical concerns are taken seriously in all use cases [2] The automotive use case and one of the industrial use cases concern safety-critical products [3] Depending of context of using the smart mirror, different privacy concerns occur [4] Low power consumption is essential in one of the industrial use cases which is battery based				

4.2.1 Define System Context

There is a lack of consistent requirements engineering method to support specifying requirements efficiently in context. Within VEDLIoT, we have investigated the state of practice with consortium partners and uncovered the following challenges [59]:

Standards and Regulations At the moment, it is unclear how requirements with validity in a specific context relate to ethics or to other standards/regulations. This makes it challenging to build arguments, e.g. for safety or fairness in a given context. Further, laws and regulations differ between countries, causing additional complexity to international companies.

Deriving Context In many cases, the environment of operation can be hard to predict, especially, if it is dynamic. It is challenging to anticipate which aspects are important, e.g. for the effective use of a particular sensor. Some environmental or contextual aspects are hard to describe, and practitioners find it challenging to argue for the completeness of a context definition.

Specifying Context The environment and context of operation is hard to model with today's methods. The lack of a clear industry standard for specifying/defining context hinders efficient communication between organisations.

Validation Due to the difficulties in deriving and specifying context and the lack of standards, it is difficult to validate whether the system fulfils its desired functional and quality goals in the intended context.

Process and responsibility Since the context definition is not a requirements artifact, it will evolve independently. There might be a lack of synchronisation and often, it is even unclear who is responsible for a change. The lack of an established common process hinders effective collaboration. Communication of requirements in context is in particular difficult, since the lack of standardised representation causes misinterpretations among stakeholders.

With these challenges, companies are forced to err on the safe side and to consider margins for achieving quality goals. Consequently, the cost of building VEDLIoT based systems will significantly be increased by the lack of requirements methods.

The goal is to *identify distinct properties of context*, relying on established ontologies and taxonomies. However, it is clear that an *iterative process with a strong feedback loop* must be strived for. Starting from simple tests and building knowledge in a strategic and prioritised way is a promising approach. Use cases can be an effective tool for communication, if they are sufficiently traced to context. The ultimate goal, however, is a standardised and structured process to engineer requirements and context definitions consistently.

4.2.2 Define System Requirements

Various notations have been suggested to specify functional system requirements. Most commonly used is a one-sentence style as follows:

> The system/component shall <requirements>

In terms of notation, we would however recommend a look into EARS (Easy Approach to Requirements Syntax) by Mavin. The basic structure of an EARS requirement is

> While <optional precondition>, when <optional trigger>, the <system name> shall <system response>

Based on this, it becomes easy to specify requirements that are ubiquitous, state driven, event driven, relate to optional features, or are unwanted.

Beyond the immediate syntax for specifying requirements, though, it is critical to ensure that requirements are up to date and consistent with system implementation and testing. In the case of VEDLIoT, this is especially true, since we anticipate that overall quality of the VEDLIoT based systems will depend on runtime monitors. If these are inconsistent with the specified requirements, the system will be hard to manage or to continuously develop.

The core reasons for changes in requirements include:

- Knowledge about the problems to solve has changed; this includes changes to the high-level domain- and product-events to consider, as well as to the operational context that must be supported.

- Cross-cutting design decisions affect requirements for depending components.

We therefore recommend to manage system requirements close to code (particularly runtime monitors) and tests. A recent trend to achieve this are systems to manage textual requirements in version control systems, as for example the open source tool TReqs[1].

A key concern for VEDLIoT based systems relates to the probabilistic behaviour of deep learning components, which render specification of a deterministic system difficult. While this remains a challenge open for further investigation, also in the remainder of the VEDLIoT project, we sketch a potential solution approach. Describe functional requirements towards the VEDLIoT component: Classify objects in video stream, identify patient in front of smart mirror. Add additional requirements on reporting the confidence or likelihood that an object indeed is in the reported class. These are functional requirements that can be specified towards a deep learning component. Add additional quality requirements with respect to classification accuracy in a given context. Based on these requirements, develop a strategy to reach functional and quality goals on system level.

4.2.3 Define Quality Requirements

The key difficulties of defining quality requirements are

- Challenging to select a metric for measuring quality
- Challenging to select a target value

A common bad practice is to specify metrics and target values early on, thus, creating a false sense of accuracy. Instead, it should be clearly mentioned if there is room for negotiation, since often just small deviations from an established quality goal can allow for much simpler solutions.

A good practice is to defer the selection of a metric or its target value, or even leave it to the supplier to specify these. Lauesen refers to these approaches as open metric and open target respectively [56].

> QR1: The smart mirror shall identify a patient with _____ accuracy (customer expects higher than 0.9)

> QR2: The model supplier shall specify the identification accuracy for similar use cases as ours.

However, once knowledge about appropriate metrics and target values becomes available, Gilb's PLanguage notation can be a good way to document quality requirements. Inspired by Lauesen [56], we provide the following example which includes hooks for working with open metrics and open targets.

[1] available at https://gitlab.com/treqs-on-git/treqs-ng

Tag: Accuracy	How often the system correctly identifies a patient
Scale:	(Supplier, please specify exact way of measuring accuracy)
Meter:	Use n-fold cross validation with training data
Must:	90%
Plan:	(Supplier, please specify)
Wish:	99%
Past:	Manual authentication, error prone

4.2.4 Derive Quality Requirements on Data and Learning

For the construction of a VEDLIoT based system, a given set of functional and quality requirements from a system's perspective in a given context has implications on the quality of input and output data as well as the learning strategies that must be guaranteed.

As with context, we identify this as a key weakness in the state of the practice in requirements engineering, where data requirements are usually described in a functional and static fashion. An empirical study within the VEDLIoT consortium reveals a number of data related challenges [76].

Accordingly, challenges around data quality relate to the availability, the management, the sources, the structure, and the trustability of data. The quality of software-based systems is commonly distinguished as internal quality (structural properties such as maintainability of the software) or external quality (the fulfilment of user requirements—i.e., providing the desired functionality and quality) [21]. In contrast, agile methods have introduced the idea to judge the quality of a system while in use in a realistic or real target environment, as for example visible in agile practices such as the on-site customer [4] and sprint demos [82]. Inspired by this, we cluster important quality attributes of data into internal quality (can be checked when looking at the data directly), external quality (can be checked when executing the system in a lab environment), and quality in use (can be checked with real users in realistic context).

- Internal quality
  - Compliance (The degree to which data has attributes that adhere to standards, conventions or regulations in force and similar rules relating to data quality in a specific context of use)
  - Correctness (Every set of data stored represents a real world situation)
  - Consistency (Measures whether or not data is equivalent across systems or location of storage.)
  - Portability (The degree to which data has attributes that enable it to be installed, replaced or moved from one system to another (while) preserving the existing quality in a specific context of use)
  - Structure (It refers to the level of difficulty in transforming semi-structured or unstructured data to structured data through technology)
  - Traceability (The extent to which data are well documented, verifiable, and easily attributed to a source)

- External quality
  - Completeness (Refers to whether all required data is present)
  - Confidentiality (A property of data indicating the extent to which their unauthorised disclosure could be prejudicial or harmful to the interest of the source or other relevant parties)
  - Cost effectiveness (The extent to which the cost of collecting appropriate data is reasonable)
  - Efficiency (The degree to which data has attributes that can be processed and provide the expected levels of performance by using the appropriate amounts and types of resources in a specific context of use)
  - Latency (The time between when the data was created and when it was made available for use)
  - Relevance (The extent to which data are applicable and helpful for the task at hand)
  - Representational consistency (The extent to which data are always presented in the same format and are compatible with previous data)
  - Usefulness (Extent to which information is applicable and helpful for the task at hand)
  - Validity (Refers to whether data values are consistent with a defined domain of values)
- Quality in use
  - Accessibility (The extent to which data are available or easily and quickly retrievable)
  - Availability (The degree to which data can be consulted or retrieved by data consumers or processes)
  - Credibility (The extent to which data are trusted or highly regarded in terms of their source or content)
  - Currency (The measure of whether data values are the most up-to-date version of the information)
  - Ease of operation (The extent to which data are easily managed and manipulated (i.e., updated, moved, aggregated, reproduced, customised))
  - Fitness (It has two-level requirements: 1) the amount of accessed data used by users and 2) the degree to which the data produced matches users? needs in the aspects of indicator definition, elements, classification, etc)
  - Interpretability (The extent to which data are in an appropriate language and units and the data definitions are clear)
  - Lineage (Lineage measures whether factual documentation exists about where data came from, how it was transformed, where it went and end-to-end graphical illustration)
  - Timeliness (Length of time between data availability and the event or phenomenon the data describe)
  - Usability (Is it understandable, simple, relevant, accessible, maintainable and at the right level of precision?)

4.3 Quality aspects of data used for systems with deep learning

Although data is of key importance to achieve a desired behaviour from a deep learning system, there is no proper procedure to determine and manage the quality of the data. There exists a need of a framework for identifying data quality challenges and defining relevant data quality attributes. As of now, most of the information regarding data quality assessment for the deep learning system is based on expert knowledge.

Effects of data quality on machine learning algorithms are studied in [84]. The authors assume that data quality has impact on the effectiveness of machine learning algorithms. They devise procedures of developing ”more robust and useful” algorithms by using data quality assessments. They evaluate the need of good data quality by developing and testing with three Bayesian networks. Their experiment concludes that data quality has an ”enormous effect” on results of machine learning algorithms.

Challenges pertaining to data quality and method of assessing them are presented in [9]. A hierarchical data quality framework is developed. The challenges of data quality the paper identifies include difficulty in data integration, large volume of data, fast-changing data, and lack of data quality standards. Furthermore, their framework is composed of big data quality dimensions, quality characteristics, and quality indices. They list and explain 14 attributes of data quality. Finally, a sophisticated data quality assessment process is presented in the paper.

A data quality assessment and monitoring framework is developed in [3]. The authors improve upon the Basel II operational risk evaluation methods. They devise a data quality assessment methodology called *ORME-DQ*. The methodology contains four phases for data quality risk prioritisation, identification, measurement, and monitoring. The authors develop an architectural framework that is composed of five modules which support the phases of the assessment methodology. For evaluation, the prototype framework is tested on the Pentaho software.

A study develops a data quality framework for distributed computing environment [20]. The author presents two models for data quality and distributed environment and combines the models to propose a measure called *Data Quality Risk Exposure Level (DQREL)*. DQREL is an attribute-dimensions matrix and it includes eight dimensions and three attributes. As stated by the author, DQREL matrix can be used to understand ”data quality pitfalls” in a system.

A set of guidelines for quality assurance of AI applications is proposed in [29]. The guideline includes three parts - a list of quality evaluation aspects, a catalogue of existing techniques, and domain-specific discussions. The evaluation aspects include data integrity, model robustness, system quality, process agility, and customer expectation. The authors also include five ”state-of-the-art trends” in the initial version of the guideline. They evaluate the guideline through a survey with over 77% of the participants agreeing on its usefulness.

Requirements engineering for machine learning-based application is studied in [89]. Among the five challenges and requirements identified by the authors, data requirements is one. Data requirement is divided into data quantity and data quality. The authors present a requirements engineering process as well. The process includes steps such as elicitation, analysis, specification, and verification & validation. These

steps can also be utilised to study requirements to identify and manage challenges of data quality.

The Open Measured Data Management Working Group (2021) has developed a vendor-neutral platform called OpenMDM[2] to manage measured data. This platform is primarily used by automotive companies to build in-house applications. It can, however, also be used to develop other solutions. It includes components and concepts that can be used to "compose applications for measured data management systems." OpenMDM can manage measurement data, evaluation results, and the descriptions.

4.3.1 A data quality assessment and maintenance framework

An artifact titled *Data Quality Assessment and Maintenance Framework* (DQA-MF) was developed in cooperation with the use case owners. It includes four components, namely, *Data Quality Workflow*, *List of Challenges*, *List of Data Quality Attributes*, and *Potential Solutions*. For more details about the concrete implementation of the framework components, see [75].

Table 4.4. Framework Components and Their Purpose (Refer Fig. 4.4 for the steps in the workflow)

Component	Purpose	Associated Workflow Step
Data Quality Workflow	Provides a step-by-step workflow to assess and manage data quality	
List of Challenges	Provides a template for challenges	S1, S6
List of Data Quality Attributes	Provides a template for data quality attributes. Also, includes information regarding which challenges affect a particular attribute, metrics & their formula, and Planguage-inspired fields	S2, S3, S4, S6
Potential Solutions	Provides a template for potential solutions to reduce or mitigate the identified challenges. Also, includes requirements specifications and implementation details (flowcharts) of the solutions	S5, S6

4.3.2 Data Quality Workflow

This component presents a step-by-step workflow for assessing and managing data quality and its pertaining requirements. It includes six steps as shown in Fig. 4.4. Most of the steps can be performed in parallel, as depicted by dotted line in Fig. 4.4. Loops indicate that the steps can be done in an iterative fashion.

4.3.3 List of Challenges

Table 4.5 presents the template of *List of Challenges* component. It includes eight fields. In its concrete implementation, 27 challenges are presented (see [76]). The challenges are divided into five broad categories pertaining to data availability, data management, data source, data structure, and trust in data.

[2] https://openmdm.org

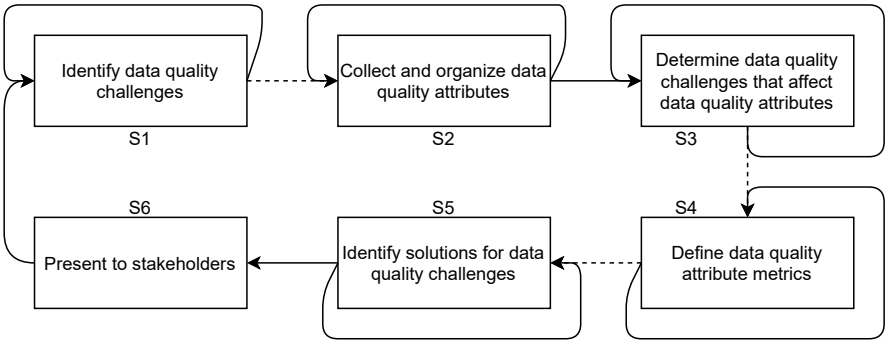

Figure 4.4. Data Quality Workflow Framework Component

Table 4.5. Template for List of Challenges Framework Component

Field	Description	Example
Name	Name of the data quality challenge	Low Labelled Data Volume
Reference	Reference that denotes the identification of the challenge	Interviews
Description	Description of the data quality challenge	In the training dataset, the volume of the data that is labelled is significantly lesser than the volume of the data that is unlabelled. Since a large volume of data is unlabelled, the unlabelled data is useless and the deep learning models cannot be properly trained. For e.g., if only 30% of the traffic signs in a scene are labelled, it would be "more difficult for the neural network to learn traffic signs, since there are quite a lot of traffic signs among the negative samples."
Type	Type of challenge. Could be AVAILABILITY, MANAGEMENT, SOURCE, STRUCTURE, or TRUST.	AVAILABILITY
Directly affects AI Functions	Boolean value to denote whether the data quality challenge directly affects AI functions or not	Yes (1.0), No (0) (Note: All participants in focus group affirmed that this challenge affects AI functions)
Challenge Score	A calculated value that denotes the ranking of the data quality challenge in terms of severity	Survey 1 - 4.333 (Rank 1/31), Survey 2 - 3.750 (Rank 1/25)
Responsible Stakeholder	People and/or department in an organisation responsible to handle the challenge.	Data Collection Department, Data Manager, Data Collector
Impact Level	Degree to which the challenge affects the AI models. Could have values such as HIGH, MEDIUM, LOW.	HIGH

4.3.4 List of Data Quality Attributes

Table 4.6 presents the template of *List of Data Quality Attributes* component. It includes 11 fields. Eighty-two data quality attributes are presented in the concrete implementation of this component. Some examples of data quality attributes include *availability*, *fitness*, *timeliness*, etc. Additionally, 30 data quality attribute metrics are presented in the concrete implementation of this component. For example, *degree of timeliness* is a metric to measure timeliness of data. Furthermore, fields from Planguage, a quality factors notation [23], can also be adapted for this component.

Table 4.6. Template for List of Data Quality Attributes Artifact Component (italics: Fields from Planguage)

Field	Description	Example
Name	Name of the data quality attribute	Timeliness
Reference	Reference that denotes the identification of the attribute	Cai & Zhu (2015), Bobrowski et al. (1970), Sidi et al. (2012), Wang & Strong (1996), Earley & Henderson (2017), CDDQ (2017)
Definition	Description of the data quality attribute	Length of time between data availability and the event or phenomenon the data describe. (Eur 2020) The extent to which the age of the data is appropriate for the task at hand. (Wang & Strong 1996)
Challenges affecting the DQ Attribute	List of challenges that affect the data quality attribute	Data Delay (1, 0.75), Data Drop (0.6, 0.25), Manual Data Collection (0.2, 0.66), Manual Data Labelling (, 0.8)
Metric	Name of the data quality attribute metric	Degree of timeliness
Formula	Formula used to calculate a given metric	Number of data records that is received within an acceptable time / Total number of received data records
Scale	The scale of measurement of the metric	The average time (in milliseconds) it takes for data to be received by AI models
Meter	The process or method of measuring the metric	Measured every time a data packet arrives by computer clock
Must	The lowest/highest acceptable threshold value that the metric must have	300 ms
Plan	The value of the metric, if surpassed, is regarded as accepted	100 ms
Wish	The optimum value for the metric	50 ms

4.3.4.1 Potential Solutions

Table 4.7 presents the template of the *Potential Solutions* Artifact component. It includes four fields. In the concrete implementation of this component, 13 potential solutions are provided. Some examples of potential solutions are *Automated Labelling* to solve Low Labelled Data Volume and Manual Data Labelling challenges, *Corroboration of Data with Central Data Repository* to solve Data Dependent on External Conditions challenge, etc. It should be noted, however, that a single solution can be suitable to solve more than one challenge.

Table 4.7. Template for Potential Solutions Artifact Component

Field	Description	Example
Name	Name of the potential solution	Corroboration of Data with Central Data Repository
Challenge it Tries to Solve	Denotes the challenge(s) a particular solution tries to solve	Data Dependent on External Conditions
Requirement Specifications	These are the requirements that should be specified before implementing the solution	1. Define the central data repository, its structure, and address, 2. Define the procedure if central data repository cannot be contacted, 3. Define the way AI disengagement notification is sent to the user
Implementation Details	This presents the step-wise implementation of the solution	Refer to [76] for implementation details

5 The influence of distributed AI on methods for specification, performance criteria and verification

The VEDLIoT project is focused on systems with distributed AI. In this chapter we will describe the change in current methods for setting the specification with performance criteria as well as how the verification of these due to the distributed AI design approach. We have identified additional inputs or outputs needed for all levels of the system.

5.1 Specification content

The definition of what constitutes a specification varies depending on at what level of requirements or even which design engineering it is written for. The specification shall fulfil some basic purposes. It shall

- tell the developers what to build
- let the testers know what testing is needed
- inform the stakeholders what they are getting

In most cases, requirements and specifications are interchangeable. But it can be said that requirements refer to a stakeholder need while the specification refers to a detailed, usually technical description of how that need will be met.

The following three paragraphs are an excerpt from Wikipedia which summarises the relationship between requirements and specifications quite well [1].

A functional specification in systems engineering and software development is a document that specifies the functions that a system or component must perform (ISO / IEC / IEEE 24765) [45].

The requirements typically describe what is needed by the system user as well as requested properties of inputs and outputs (e.g. of the software system). A functional specification is the more technical response to a matching requirements document. Thus, it picks up the results of the requirements analysis stage. On more complex systems multiple levels of functional specifications will typically nest to each other, e.g. on the system level, on the module level and on the level of technical details.

A functional specification does not define the inner workings of the proposed system; it does not include the specification of how the system function will be implemented. Instead, it focuses on what various outside agents (people using the program, computer peripherals, or other computers, for example) might ”observe” when interacting with the system.’

A good specification document is always reflected by the quality of requirements, because the main elements in a specification are the requirements. Through the task clarification process, the stakeholders’ needs, which are in the form of customer language, are transformed into engineering terms for the designing tasks [10]. The required content of a specification has been investigated by numerous groups. Some findings are listed below.

Sudin et al. [68] found the customer or client's specification can be classified into three different areas

- verbal
- semi-developed
- full specification

Ulrich and Eppinger [87] found that a specification consists of a metric and a value. Ullman [86] had 7 areas of classification which are

- functional performance requirement
- human factor requirement
- physical requirement
- reliability requirement
- life cycle concern requirement
- resource concern requirement
- manufacturing requirement.

Salonen et al. [66] also had seven classes but with slightly different focus: requirement related to feasibility

- technical requirement
- requirement related to size and appearance
- requirement for manufacturing and assembly
- requirement related to installation and use
- requirement for service
- requirement related to life cycle

Salonen also stated that requirements also could be classified based on their importance to the design process.

Roozenburg and Eekels [48] have identified a method for making a design specification. The method consists of three phases that are:

- listing objectives
- analysing of objectives
- editing objectives

In order to achieve a complete collection of objectives and to minimise the chances of missing relevant objectives, they referred to a checklist comprised of three major elements that were: the stakeholder, the aspects and the product life cycle.

Therefore, to imagine the solution, while deriving requirements, could be a good technique for specifying requirements, but ideally it should be stated as a solution-neutral statement in a specification to avoid bias. Even though the solution is requested by the customer, design engineers need to investigate the reasons behind this solution preference before the decision to include it in the specification is made.

There is also a trade-off between the level of detail in the specification and the speed of the development as shown in [100]. Although focusing on software development this may also apply to the full system specification.

5.2 Requirement and Specification processes

Most industries have processes and policy documents for the requirement process. Figure 5.1 is from a Veoneer standard showing the requirement steps as well as the corresponding verification in the typical V process.

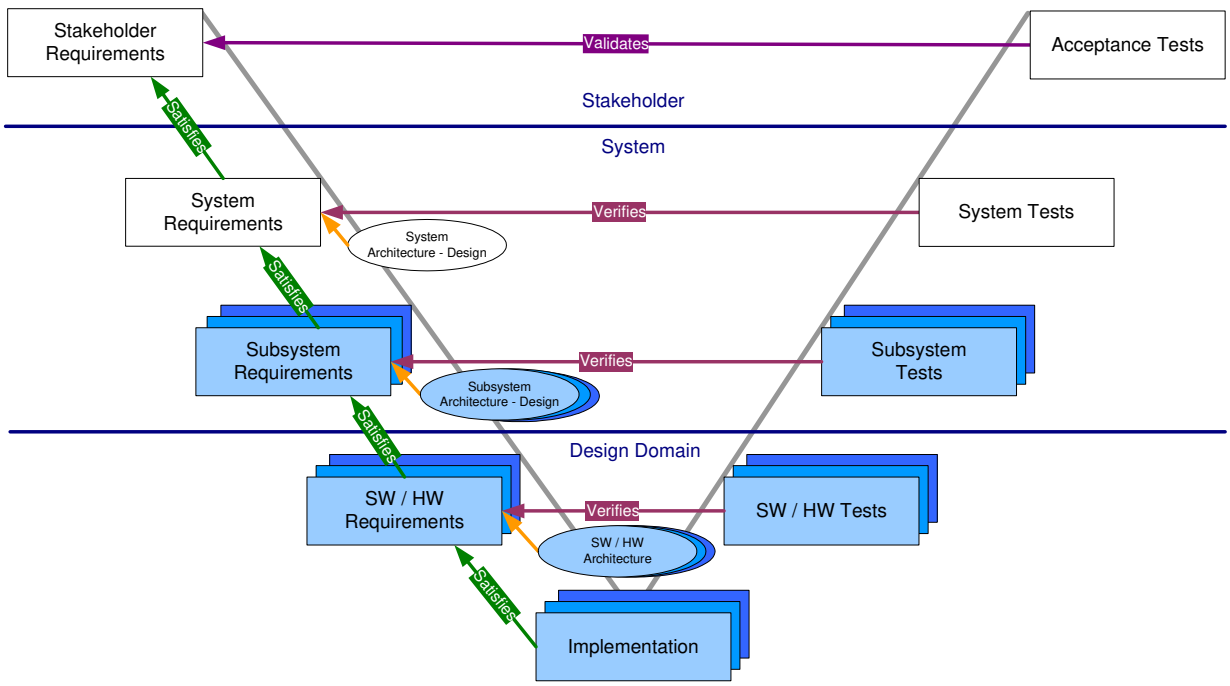

Figure 5.1. Visualisation of Veoneer requirement process

Analysing the detailed process documents, used by Veoneer, shows a conformance with the previously discussed areas of specification and the description of these areas. Also the different phases, as described by Roozenburg and Eekels [48] are covered by the requirement steps called “Elicit and analyse” and ”Structure and categorise in System level and below”.

As can be seen in Figure 5.1 there is a parallel track, to the requirements path, that describes the architecture at different levels of detail. In the VEDLIoT project, we propose a requirements managing method (see Chapter 3) that combines this into one strict and visual process that will enable us to generate the wanted specification.

It is obvious from Figure 5.1 that there is a feedback path from each of the verification boxes to the corresponding requirement level. But there is also a feedback from each lower level, in the requirement and architecture paths to the previous higher level. At some point these feedback loops have to stop and this is referred to as “Freeze requirements” in the Veoneer process.

In the VEDLIoT project we are developing an architectural framework (chapter 3) and requirement method for (dynamically) distributed AI systems. This is illustrated in chapter 3 but repeated here (Figure 5.2) for convenience.

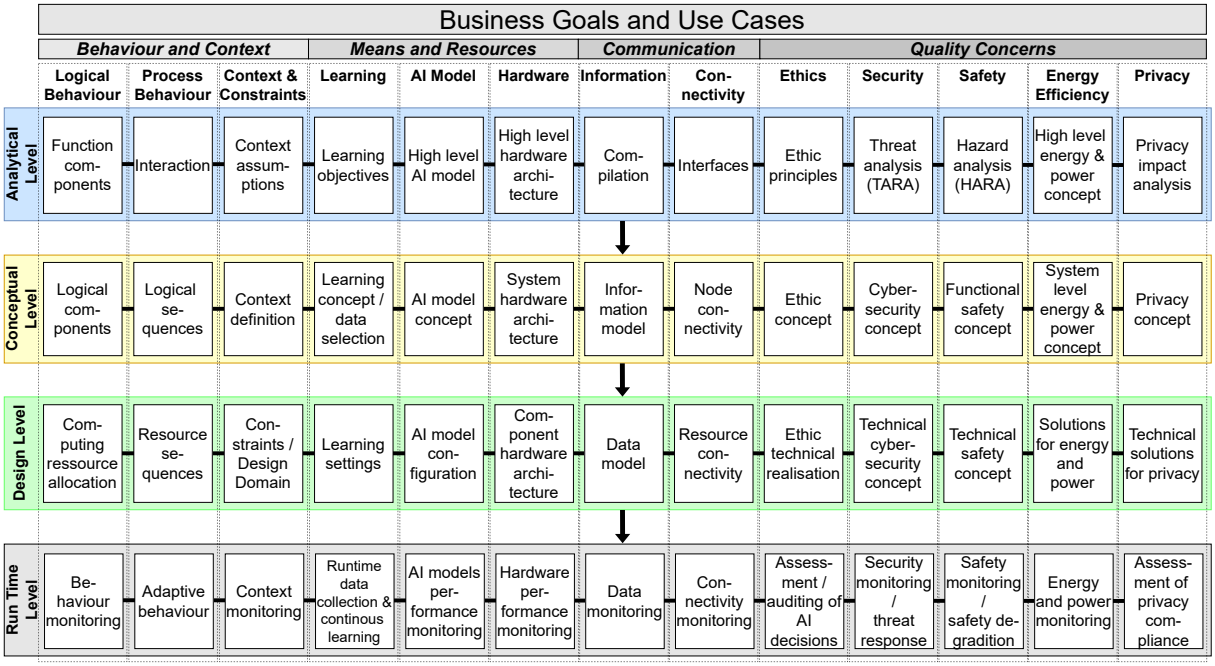

Figure 5.2. An architectural framework merging clusters of concerns for an AI enabled system

5.3 Technology neutral sub system definitions

As indicated by the proposed architectural framework in chapter 3 (Figure 5.2) and in the V-model in Figure 5.1 the actual selection between hardware and software implementation is done at the final requirement level. The sub-system requirements should to some extent be technology neutral. Although some function sub-systems are initially envisioned to be for example software, the influence of other stakeholders, or clusters of concern, like for example functional safety, may force the selected technology to be shifted. This may not have been obvious in the higher levels of abstraction. Also, not selecting the solution technology early may reduce the need for multiple iterations between the different levels of abstraction.

For a dynamically distributed AI system this adds another level of complexity as the full system is not under full control of the implementation engineers. If some function processing are to be done in the cloud or edge or in an IoT device, depending on the actual context, the implementation may have to shift between hardware and

software. The requirements, for example what kind of AI processing is needed, shall be fully covered by the proposed architectural framework but the final design or implementation is not set until the final level of abstraction ("design level" in Figure 5.2)

5.4 Synchronisation requirements

In the proposed architectural framework shown in Figure 5.2, there are functional descriptions and operational flow charts describing the application. These are then converted into sub components or modules. These components/modules must operate in a defined time sequence to fulfil the functional and operational flow charts. These timing requirements are not necessarily described in the architectural framework and they will also be different depending on the choice of implementation.

Although some timing requirements are obvious, like data have to be available before the processing of data starts, the actual margin needed is probably very implementation dependent. This is especially important in a distributed AI system where the amount of data and the time to transport data to the processing node is very dependent on the system design and the final implementation.

5.5 System specification method

Many methods already exist for creating a specification of a system. The method that fits the proposed architectural framework is a model oriented formal specification (see Figure 5.3 below).

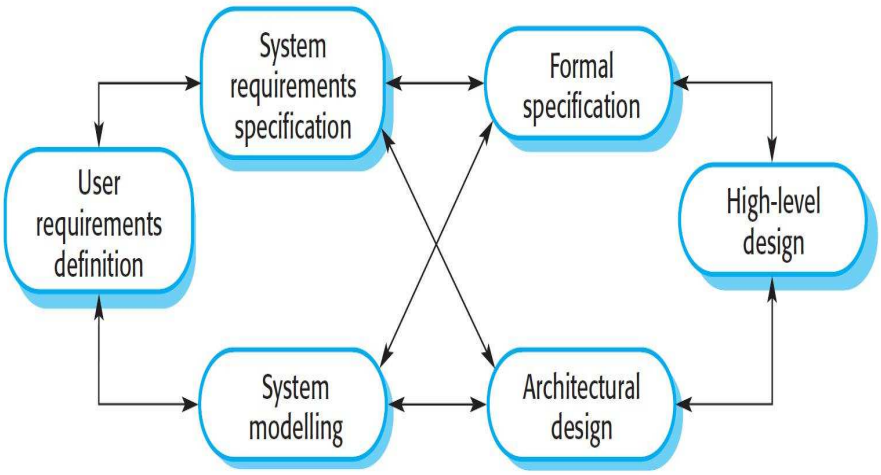

Figure 5.3. An architectural formal specification as part of requirements specification [85]

In the above method all activities, except the formal specification, are covered by the proposed architectural framework method.

Referencing back to the architectural framework illustrated in Figure 5.2, the method for deriving the formal specification is based on the output of the design level.

The output is called a solution prototype and this includes all the components/modules needed to fulfil the solution requirements and it is also based on a preferred ar-

chitecture. At this level, all the identified individual components/modules have their own requirements.

At this point, all the clusters of concern by all stakeholders shall have been considered and included. The necessary iterations between the different levels of abstraction as well as the different clusters of concerns have to be halted ("freezed" in the Veoneer nomenclature). All abstraction levels of the system architecture shall now have sufficient information to create a specification that can be used for verification. That is, it shall enable creation of parameters and KPI's for which it shall be possible to setup tests and measurements with thresholds and rules indicating the fulfilment of the requirements at all abstraction levels.

What are the parameters and KPI's that are suitable for distributed AI. The following chapters describe possible types.

5.6 AI performance criteria

Deliverable 3.1 of this project provides a detailed discussion on suitable performance metrics for very efficient deep learning in the Internet of Things[1]. Here we provide an extract of the most relevant metrics:

5.6.1 Functional Performance

Suitable metrics for functional performance are execution time of a software component, time until successful inference, and latency.

Table 5.1. Performance Metrics

Performance Metrics (Application), Full table will be part of Deliverable 3.1[1]	
Execution time [s]	Total execution time for inference
Latency [s]	Time from recorded event inference including preprocessing
Peak performance [ops/s]	Peak performance when executed
Achieved performance [Inferences/s]	Performance achieved for a specific DL-model

5.6.2 Qualitative Performance

In contrast to the functional performance, the qualitative performance measures the fulfilment of the qualitative aspects of the system, such as accuracy and efficiency. The desired quality of the inference in most cases depends on the functional performance goals.

Table 5.2. Quality Metrics

Quality Metrics, Full table will be part of Deliverable 3.1[1]	
I/O Bandwidth	Bandwidth for data transfers
I/O Latency	Latency for data transfers
Reconfiguration time [s]	Dead time from starting the update process till system is running again
Power [W]	Total system power, averaged over one cycle
Energy [J]	Total energy consumed in one cycle
Accuracy	Accuracy, Precision, Recall, F1-score, F2-score, ...

[1]The Deliverable 3.1 can be downloaded at **http://www.vedliot.eu**

5.6.3 Relation to intended use-case context

A key concern of performance metrics related to the intended use-case context is the quality of training data. In addition, a measurement plan must be provided.

5.6.4 Training Data Quality

Training data quality can be described based on the quality attributes in Section 4.2.4. In a thesis, we currently investigate possible data quality attributes for datasets used to train deep learning models. Potential metrics include:

Table 5.3. Quality Metrics of data

Quality Metrics of Data, full table in [76]	
Accuracy	Percentage of errors
Age of data	Resiliency
Availability	Scalability
Completeness	Signal strength
Correctness of Data	Signal-to-noise ratio
Data Loss Rate	Standard deviation
Elasticity	Update Rate
Error Margin	System Size
Error rate	Useability
Estimated bias	Variance
Frequency	Velocity of Data
Mean of Data	Volume

5.6.5 Safeguarding automation

It will be critical to automatically identify scenarios, where the desired quality and functional goals cannot be reached. This can be due to the fact that the system is not operated in the context it was designed for or due to other reasons that cause the input data to the inference mechanism to deviate from the established data quality goals. In such cases, the system must react appropriately, e.g. by deactivating certain functionality, by shutting down, or by transferring into a mode that aims to bring the system back to a safe state.

5.7 Additional considerations for distributed AI

The implementation of the system shall now be based on the specification. Some requirements may not have a clear path to the specification parameters. This may be especially true for AI systems. A requirement could be the function reliability which indicates certain accuracy in the AI processing network based on the initial learning set. This set may not have been sufficiently broad and during development and implementation with a wider learning set, as defined by the requirement process, it is discovered that a higher accuracy is needed. This will then impact the processing hardware and/or the processing time. The solution would be to have larger specification margins in case of AI systems but this would then not be cost optimised.

AI models may be optimised after training, and there are always trade-offs. For example the size (depth) and arithmetic precision can be modified to reduce the hardware demands but at the cost of lower performance with respect to inference accuracy and precision.

Another issue is distributed processing that relies on connectivity and third party processing resources. Both of these resources will have very dynamic performance, with respect to availability and latency, and need to be included in the solution. An example is when an edge processing resource in a cellular base station is used for inference and data need to be transferred from and back to the user over a cellular communication link. Both the communication and the processing capacity is dependent on other concurrent users which will vary over time. It will put hard demands on the selection of use-cases but also the reconfigurability of the real time system. It will absolutely require a system monitoring process to detect situations that cannot be handled. Some of these will be covered by the specification but it may be hard to manage all combinations of environment conditions.

In parallel with the system specification the product/application operational design domain (ODD) specification will be generated. The ODD is based on the selection of system and sub-system functionality as produced through the requirements process. The ODD may be limited for example due to conscious selection of sub-components based on availability and/or cost. The product/application will still fulfil the requirements in most cases but in special cases like for example in hard to detect weather conditions (low sun and very light rain) and similar the system will not work fully and it will not be able to detect this condition. This has to be stated in the ODD.

5.8 Verification Methods

Based on the specification and derived performance metrics, a methodology which acknowledges the impact of distributed AI needs to be established. In the proposed VEDLIoT architectural framework, demonstrated in Figure 5.2, it is clear that verification methods should be applied on each individual "cluster-of-concern". It's also important to highlight the need of verification methods on the horizontal dimension, providing a methodology to ensure the integration of the clusters of concerns, as well as the vertical dimension, providing a methodology to verify the maintained functionality as requested.

The developed functions should be tested by different tools and methods throughout the development process, including conceptual, as well as functional validation. The verification process should be initiated already early on in the levels of abstractions of the architectural viewpoints and intensified throughout the process.

5.8.1 Simulations

Computer simulations should be considered during the early stages of the development process as a cost effective measure to generate preliminary results. This is no different for deep learning than traditional software development.

Simulations are mainly targeted towards the clusters of concerns *Behaviour and Context*, *Means and Resources* and *Communication*. The *Quality Concerns* doesn't necessarily benefit from simulations because they are governed by existing standards such as ISO 26262.

Simulations can be used to target individual boxes in the VEDLIoT architectural framework or target multiple boxes along the horizontal and vertical dimension of the framework by increasing the scope through additional mathematical models.

5.8.2 System Integration testing

Verification efforts needs to go into integration testing in an iterative manner to ensure compatibility between subsystems. This is valid for subsystems with and without components based on deep learning. This should be done through a number of steps, including for example Software-in-the-loop testing, hardware-in-the-loop testing and system-in-the-loop testing. Referring back to the architecture framework for VEDLIoT in Figure 5.2, system integration testing should involve the horizontal of clusters of concern, focusing on the *Behaviour and Context*, *Means and Resources* and *Communication* while complying with *Quality Concerns*.

5.8.3 Quality verification

Quality verification is today governed by the major standardisation bodies, e.g. IEEE and ISO, which provides a thorough guide through the necessary documentation that should be created and provided. This is the current practice to follow regardless of involvement of deep learning or in the traditional software.

However, it is important to notice that the functional safety standard ISO 26262 in today's form does not cover deep learning. This has been identified by ISO and the work groups are currently working on amendments to the standard to also cover deep learning. VEDLIoT will follow the progress of the work groups.

5.8.4 Validation data selection

For any system using DL, the selection of validation data is crucial to properly estimate the correctness of DL performance. Hence, the validation data needs to be carefully compiled to represent the likely variations in the observations where the system will operate. An example might be a vision system trained to recognise the location of an eye. If said system was solely trained and validated towards a data set containing a population of blue eyes only, other variations of eye colours will have undefined result where the variations could be improperly ignored.

5.8.5 Field tests

At different stages throughout the development process, field tests should be performed in settings defined for the targeted function design and the intended ODD. This includes field tests in closed and controlled environments (such as test tracks) and real life test (such as on public roads).

5.9 Runtime Monitoring

The Vedliot project aims at finding requirement, hardware and software solutions for Very Efficient Deep Learning in IoT. As part of this, efficient methods for specification and verification are being investigated. In this document we have identified a method based on an Architectural Framework, see also Chapter 3. A part of this framework defines a system Real-Time Monitor (RTM) which should be operational during the life-time of the product. The requirements for this RTM are derived from all viewpoints of the Architectural Framework.

The RTM may be implemented as multiple components of both hardware and software. It may be connected to the functional system at multiple interfaces and at different levels of functionality. It may e.g. be simple monitoring of power supply voltage levels as well as checking the statistical distribution of some processing out-

put parameters. The aim of the RTM is to verify that the system is operating within the ODD as described in the product specification and against which the system has been verified. To satisfy the functional safety requirements, the RTM has to consider both FuSa (ISO 26262) and SotIF (ISO 21488) standards. This functional safety aspect is discussed further in WP5 of the Vedliot project. The V-model as shown in Figure5.4 indicates that the RTM must handle all functional safety cases.

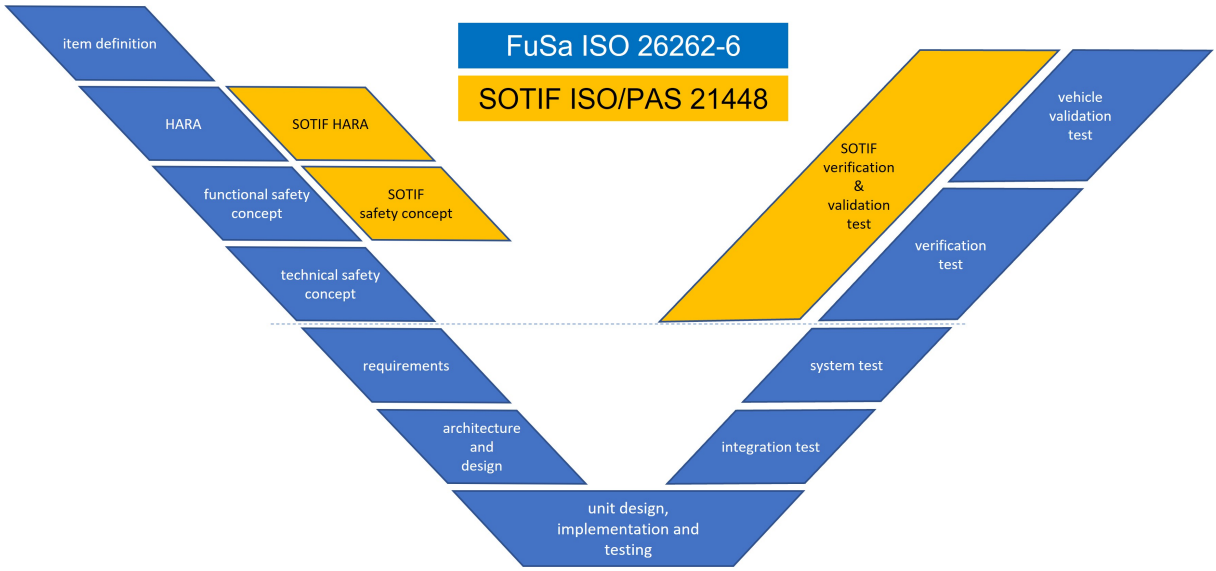

Figure 5.4. V-model with Functional Safety and SotiF

An additional benefit of the RTM is that if the system fails the RTM data may be used for forensic analysis of the system to identify the root cause of the failure. The RTM may also be used to identify correlated changes in the behaviour of the system that would indicate that a ML/DL design has not had sufficient training data from some scenarios within the ODD.

5.9.1 Real-Time Monitoring functionalities

The basic system concept is illustrated in Figure 5.5.

Function Level

This is where the actual real-time application or function runs. This could be based on a model or rule based algorithm or a DL/ML algorithm. It is using input data from sensors and output control data for actuators. This level does not include FuSa functionality.

Function Monitoring Level

This layer executes safety functions that monitor the functionality of layer 1. Here is the implementation of the safety functions on the real-time application, such as plausibility checks of the input signals.

Module Monitoring Level

This layer monitors hardware and software modules that executes the algorithms of *Layer 1: Function Level* and the monitoring modules of *Layer 2: Function Monitoring Level*. The safety functions of layer 3 are less application-specific and more general and can be extracted by using the FuSa process.

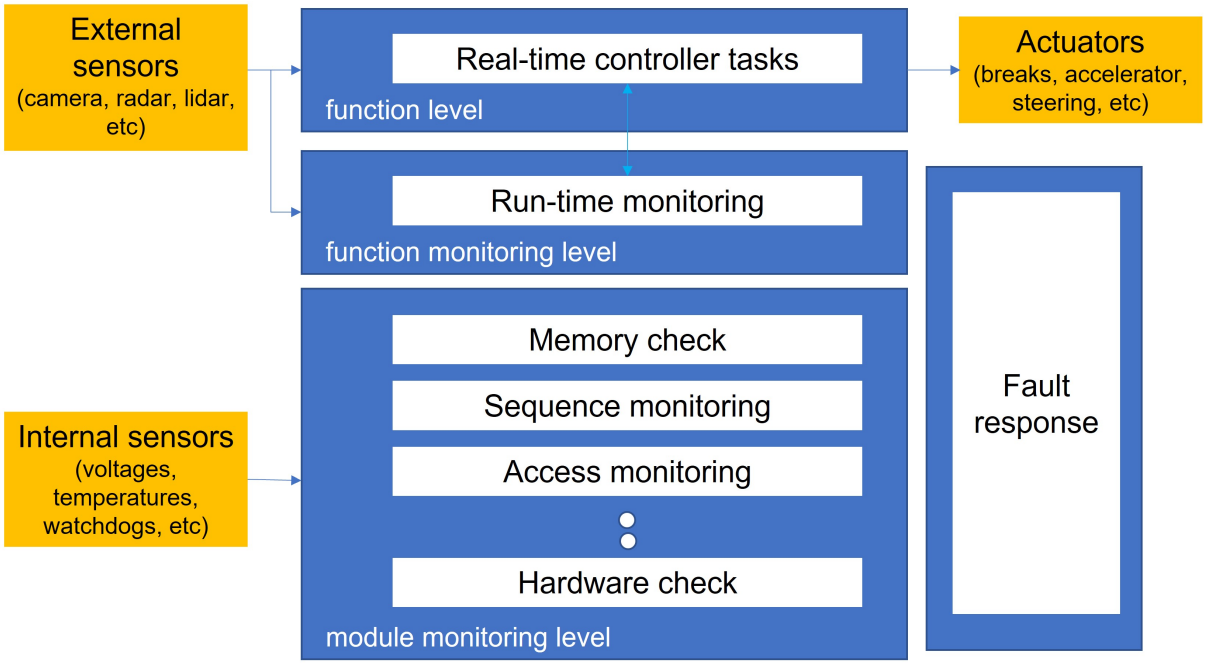

Figure 5.5. RTM system concept

The RTM should be designed to cover some basic areas. These are described in the following sub-chapters. They may be defined as reactive versus pro-active monitoring.

5.9.1.1 Anomalies and fault detection

The monitoring is aimed at finding parameters that are outside of the operational limits. This is normally achieved by setting thresholds with sufficient margin with respect to the operational limits of individual components. Typical examples are again power supply voltage and current levels. It could also be a software watchdog which checks the maximum processing time of software components or functions. These anomalies and fault detections are mainly included to fulfil the ISO 26262 standard. Typical monitoring areas include

- Hardware power supplies
- Hardware temperature
- System clocks stability
- CPU and memory utilization
- Memory integrity check
- Application response time
- Network latency
- Access to system resources

5.9.1.2 Trends and Performance

Although this solution may look at the same parameters as described in "Anomalies and fault detection", in this case we are more looking at the derivative of the parameters trying to estimate if they at some point in the future will exceed the operational thresholds. This method also examines the system performance by looking at the statistical distribution of the function module outputs. One important part of this method is to look at the quality of system input data, e.g. sensor data, trying to determine if the system is operating within the specified ODD. Below is some important parameters for both deterministic or probabilistic methods (also including ML/DL models)

- Model Drift (distribution variability, input-output data relationship)
- Model Performance (output parameter distribution)
- Model Outliers (detection and statistics)
- (ML) Data Quality (input and output data)

The monitoring analysis is fully based on input and output data and the actual signal processing model, which could be an ML/DL model, may be considered as a black box as indicated by Figure 5.6. The figure illustrates an automotive use-case but is obviously valid for any application and processing model.

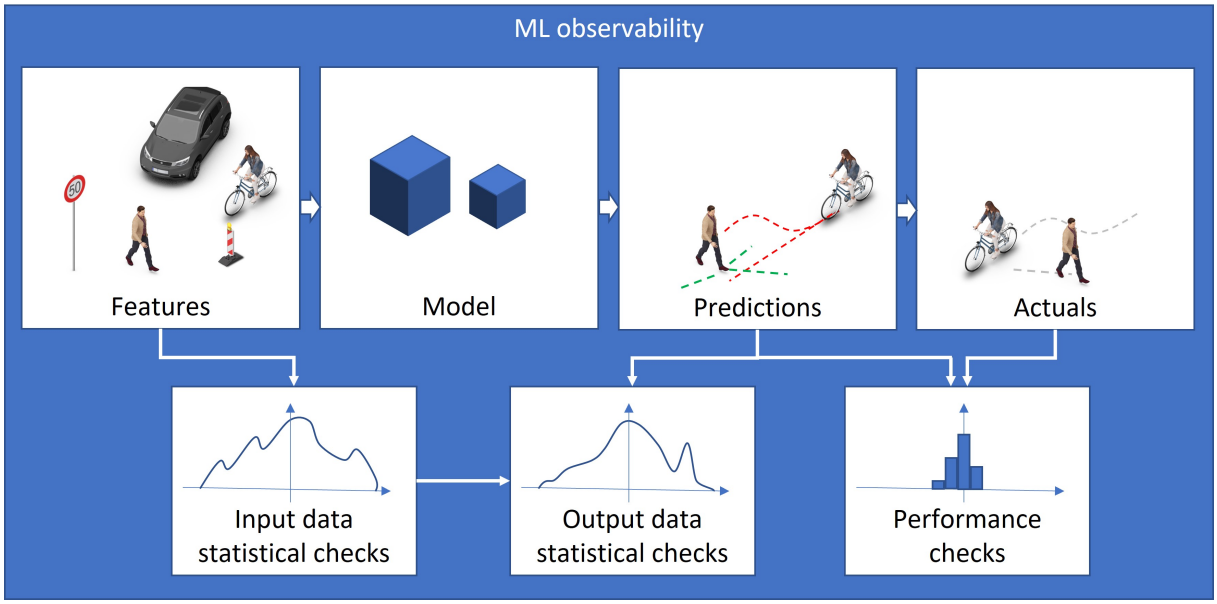

Figure 5.6. Trends and Performance monitoring

5.9.1.3 Challenge-Response Monitoring

This is a way of injecting fault or near-boundary signals on the inptus of the system and checking the response at the output. This includes verifying the output response timing and quality.

5.9.2 RTM and functional Safety

The RTM shall be designed to verify in real-time that the system is operating under the specified conditions. The RTM shall warn if the system is close or beyond any operational limits to fulfil the Functional Safety requirements. The two different areas for RTM described in the previous chapter matches the two areas of functional standards FuSa (ISO 26262) and SotIF (ISO 21488).

5.9.2.1 Existing solutions

Many processing devices, e.g. ARM processors [63] already incorporate real-time monitoring functions. These could be

- **Spatial isolation** which is set by MPU Protected Zones that use a processing device's Memory Protection Unit (MPU) to shield access to memory and peripherals. Access to RTOS objects and Kernel operations could be controlled with assigned Safety Classes.
- **Temporal isolation** is set with Thread Watchdogs which monitors the system timing constraints.
- **Controlled system recovery** is a way of putting the system in a safety state and/or prevent the execution of non-safety components.

An example of this is shown below in Figure 5.7 where safety critical functionalities and uncritical functionalities are running together on a single core Arm processor.

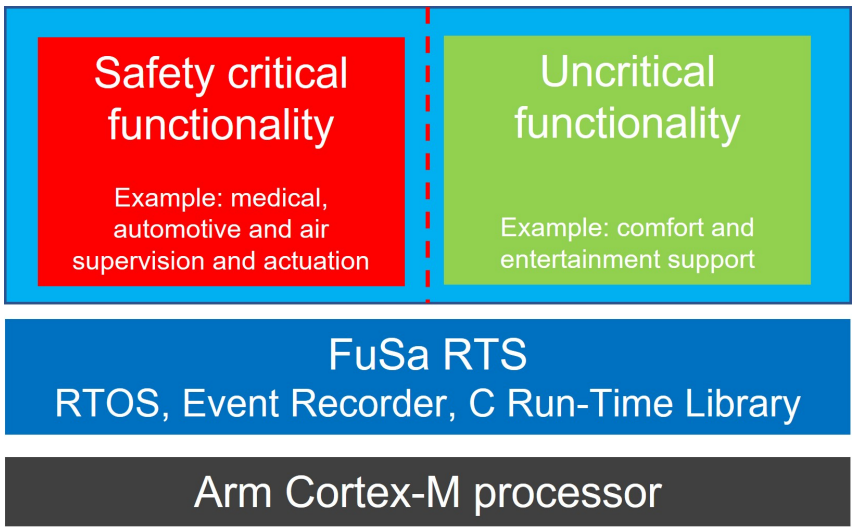

Figure 5.7. Proposed real-time monitoring for a single core Arm processor

5.9.3 RTM and AI/ML

The real time monitoring of signal processing modules based on ML/DL is in general the same as described in the previous chapters. The main difference is that there may not be an easily understandable connection between input and output data. The monitoring has to be based on results created during the learning phase.

It may be hard to set a deterministic model of the correlation between input and output data for an ML system. One way would be to use another ML module for

monitoring this correlation. Due to entanglement this will force re-learning of the monitor when updating the application ML/DL model.

Modelling assumptions remain fairly constant and are obviously based on historical data. The real world is dynamic, so the input data will change over time due to natural reasons and not sensor performance. This could lead to a degraded model performance. The ML models must adapt to this changing environment. The monitoring conditions may therefore also have to change over time.

Also changes to the system components over the life-time of the system may have a large impact on the ML system outputs. The robustness of the model predictions may be influenced both on learning settings (e.g. hyper-parameters, sampling methods, convergence thresholds, and data selection) as well as input data quality change (both improved or degraded) can cause unpredictable changes to model output.

All of the above may be referred to as CACE: Changing Anything Changes Everything.

5.9.4 RTM and continuous learning

This means that we need to continuously monitor whether the assumptions baked into the model at training time continue to hold at inference time. This form of monitoring is extremely difficult because it requires advanced statistical capabilities and careful tuning in order to prevent “alert fatigue” i.e. too many false positives. But failing to catch violations of model assumptions negatively impacts both the user experience and business KPIs.

5.9.5 RTM and post-event data correlation

A straightforward way to monitor your ML model is to constantly evaluate your performance on real-world data. Any significant changes in metrics such as accuracy, precision, or F1 should be detected and create notifications to the system.

The system RTM may use external data to evaluate the limits and boundaries of the metrics. For example another sensor system could provide stored ground truth data for a specific event and this data could be used by the RTM to compare the operational result of the ML model. By doing this we can have a confirmation of the RTM specification.

Since any external data either have to be moved to the RTM system for local evaluation or the RTM data has to be moved to an external processing node for the RTM performance evaluation it will involve data transfer which may be costly and time consuming.

5.9.6 Summary

The real-time monitor is a critical component in any safety related application. Most of the requirements on the RTM stems from functional safety requirements and analysis as defined in the FuSa and SotIF standards (ISO26262 and ISO21488). The RTM is the guarantee that the complete system operates within the defined limits as described in the ODD and the system specification.

Further investigation in the design of the RTM is needed. Since there is a close connection to the functional safety requirements we will proceed with this work in WP5 of the Vedliot project.

6 Conclusion

The conceptual distance between the VEDLIoT use cases, and the necessary integration of additional use cases from the open call, made it difficult to define a single reference architecture valid for all use cases. For example, the smart home use cases must put significantly more emphasis on data privacy than the industrial IoT use cases. On the other hand, functional safety is paramount to the automotive use case, but of less importance to the smart home use case.

Therefore, instead of a single architectural framework, a compositional architectural framework was derived during focus groups within the project consortium. Compositional thinking allows for an effective co-design of all relevant concerns of the system-of-interest. Especially for AI components, the architectural framework allows for effective data selection, AI model developing, and hardware design. Qualitative aspects, such as safety, security, privacy, but also ethical aspects are explicitly considered throughout the design process. Furthermore, to ensure functionality and quality aspects of the system, the architectural framework considers monitoring concepts for run-time operations of the system.

Based on the architectural framework and its architectural decomposition mechanisms across the key concerns of VEDLIoT based systems, a requirements engineering method was defined. The requirements method provides initial input to the highest abstraction level of the architectural framework, preparing the selection and prioritisation of concerns, and providing the key input for architectural analysis. It then continues to describe the VEDLIoT system under construction based on the highest architectural framework layer and provides the foundation for further architectural analysis. In this way, the requirements method aims to support the twin peaks model, where work on architecture and requirements evolve in parallel. It also allows to mix top-down and bottom-up work, providing the conceptual glue to connect existing hardware concepts to high-level system concerns.

Verification methods have been described for the four main clusters of concerns.

There remain however a number of severe challenges that are not easily solved. It was found that there is a lack of standardised processes to define context and quality of data in relation to requirements. We list potentially relevant data quality attributes as well as key difficulties in negotiating operative context across the value chain involved in building VEDLIoT based systems.

The work in this task of defining specifications, which include AI/DL processing, has identified some challenges which may be solved by more stringent evaluation of the higher levels of abstraction or using a limited specification, with respect to AI and DL, and with an adapted monitoring function. It may also require a more limited ODD.

The framework and concepts described in this work package serve as direct input to other work packages. The architectural framework and the requirement framework are already implemented for describing the use cases and the open call in VEDLIoT as part of Work Package 7. Concepts, such as the monitoring concept, will, for example, enable safety and security aspects discussed in Work Package 5.

Acronyms

Acronym	Meaning
ADAS	Advanced driver assistance system
AEB	Automatic emergency braking
AI	Artificial intelligence
DL	Deep Learning
E/E	Electric and electronic
GDPR	General data protection regulation
IoT	Internet of Things
KPI	Key performance indicator
ML	Machine learning
NLP	Natural language processing
OEM	Original equipment manufacturer
ODD	Operational design domain
RE	Requirement engineering
SAE	Society of Automotive Engineers
SHAPE-IT	Supporting the Interaction of Humans and Automated Vehicles: Preparing for the Environment of Tomorrow
VEDLIoT	Very efficient deep learning in Internet of things

7 References

[1] Wikimedia Foundation Authors. Wikipedia: Functional specification.

[2] Georgios Bakirtzis, Eswaran Subrahmanian, and Cody H. Fleming. Compositional Thinking in Cyber-Physical Systems Theory. *arXiv*, pages 1–9, may 2021.

[3] Carlo Batini, Daniele Barone, Michele Mastrella, Andrea Maurino, and Claudio Ruffini. A framework and a methodology for data quality assessment and monitoring. In *ICIQ*, pages 333–346. Citeseer, 2007.

[4] Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 1999.

[5] Lucas Bernardi, Themis Mavridis, and Pablo Estevez. 150 successful machine learning models: 6 lessons learned at Booking.com. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1743–1751, 2019.

[6] Mónica Bobrowski, Martina Marré, and Daniel Yankelevich. A software engineering view of data quality. *Proc. of Second Int. Conf. Software Quality in Europe*, 1998.

[7] Markus Borg, Cristofer Englund, Krzysztof Wnuk, Boris Duran, Christoffer Levandowski, Shenjian Gao, Yanwen Tan, Henrik Kaijser, Henrik Lönn, and Jonas Törnqvist. Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry. *Automotive Software Engineering*, 1(1):1–19, 2019.

[8] Jan Bosch, Ivica Crnkovic, and Helena Holmström Olsson. Engineering AI Systems: A Research Agenda. *arXiv*, pages 1–19, jan 2020.

[9] Li Cai and Yangyong Zhu. The challenges of data quality and data quality assessment in the big data era. *Data science journal*, 14, 2015.

[10] Amaresh Chakrabarti. *Research Into Design: Supporting Sustainable Product Development*. Research Publishing Service, 2011.

[11] Jane Cleland-Huang. Safety stories in agile development. *IEEE Software*, 34(4), 2017.

[12] Jane Cleland-Huang, Robert S. Hanmer, Sam Supakkul, and Mehdi Mirakhorli. The twin peaks of requirements and architecture. *IEEE Software*, 30(2):24–29, 2013.

[13] Jane Cleland-Huang, Mats Heimdahl, Jane Huffman Hayes, Robyn Lutz, and Patrick Mäder. Trace queries for safety requirements in high assurance systems. In *Proc. of Int. Working Conf. on Requirements Eng.: Foundation for Software Quality (REFSQ)*, pages 179–193, Essen, Germany, 2012.

[14] Command and Control Board. *NATO Architecture Framework*. NATO Science and Technology Organization, 2020.

[15] John W Creswell and J David Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.

[16] Krzysztof Czarnecki. Operational design domain for automated driving systems - taxonomy of basic terms. Technical report, Waterloo Intelligent Systems Engineering (WISE) Lab, University of Waterloo, 07 2018.

[17] M. J. de Vries. *Philosophy of Technology*. Technology Education for Teachers, Rotterdam, 2012.

[18] Parijat Dube and Eitan Farchi. *Automated Detection of Drift in Deep Learning Based Classifiers Performance Using Network Embeddings*, volume 1272. Springer International Publishing, 2020.

[19] European Commission. REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCILLAYING DOWN HARMONISED RULES ON ARTIFICIAL INTELLIGENCE (ARTIFICIAL INTELLIGENCE ACT) AND AMENDING CERTAIN UNION LEGISLATIVE ACTS, 2020.

[20] Finbar Fletcher. A framework for addressing data quality in distributed computing systems. In *IQ*, pages 265–282, 1998.

[21] Steve Freeman and Nat Pryce. *Growing Object-Oriented Software, Guided by Tests*. Addison-Wesley Professional, 1st edition, 2009.

[22] Greg Giaimo, Rebekah Anderson, Laurie Wargelin, and Peter Stopher. Will it Work? *Transportation Research Record: Journal of the Transportation Research Board*, 2176(1):26–34, jan 2010.

[23] T Gilb. A handbook for systems-engineering, requirements-engineering and software-engineering using planguage, 2005.

[24] Martin Glinz. On non-functional requirements. In *Proc. of 15th IEEE Int. RE Conf. (RE)*, pages 21–26, New Delhi, India, 2007.

[25] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *The No No Free Lunch Theorem*, chapter 5.2.1, pages 114–116. MIT press Cambridge, 2016.

[26] Lucas Gren and Per Lenberg. Agility is responsiveness to change: An essential definition. In *Proc. of the Evaluation and Assessment in Software Engineering*, pages 348–353, 2020.

[27] Edward R. Griffor, Christopher Greer, David A. Wollman, and Martin J. Burns. Framework for Cyber-Physical Systems: VOlume 1, Overview. Technical Report 1500-201, National Institute of Standards and Technology (NIST), June 2017.

[28] Magnus Gyllenhammar, Rolf Johansson, Fredrik Warg, DeJiu Chen, Hans-Martin Heyn, Martin Sanfridson, Jan Söderberg, Anders Thorsén, and Stig Ursing. Towards an operational design domain that supports the safety argumentation of an automated driving system. In *10th European Congress on Embedded Real Time Systems (ERTS 2020)*, 2020.

[29] Koichi Hamada, Fuyuki Ishikawa, Satoshi Masuda, Tomoyuki Myojin, Yasuharu Nishi, Hideto Ogawa, Takahiro Toku, Susumu Tokumoto, Kazunori Tsuchiya, Yasuhiro Ujita, et al. Guidelines for quality assurance of machine learning-based artificial intelligence. In *SEKE*, pages 335–341, 2020.

[30] P. Hancock. Some pitfalls in the promises of automated and autonomous vehicles. *Ergonomics :1*, 2019.

[31] P. A. Hancock. Imposing limits on autonomous systems. *Ergonomics 60 (2)*, page 284–291, 2017.

[32] Bernd Heinrich, Diana Hristova, Mathias Klier, Alexander Schiller, and Michael Szubartowicz. Requirements for data quality metrics. *Journal of Data and Information Quality (JDIQ)*, 9(2):1–32, 2018.

[33] IEEE SA Board of Governors/Corporate Advisory Group (BoG/CAG). *IEEE Std 2413: Architectural Framework for the Internet of Things (IOT)*. IEEE Computer Society, 2019.

[34] IEEE Std 2413-2019. IEEE Standard for an Architectural Framework for the Internet of Things (IoT), 2019.

[35] International Electrotechnical Commission. *IEC 60050-351:2013: International Electrotechnical Vocabulary*. International Electrotechnical Commission, Geneva, 2013.

[36] International Electrotechnical Commission. *Function blocks - Part 1: Architecture*. International Electrotechnical Commission, Geneva, 2014.

[37] International Electrotechnical Commission. *Guidelines for the design of interconnected power systems*. International Electrotechnical Commission, Geneva, 2014.

[38] International Electrotechnical Commission. *Function blocks (FB) for process control and electronic device description language (EDDL) - Part 2: Specification of FB concept*. International Electrotechnical Commission, Geneva, 2018.

[39] International Organization for Standardization. *ISO / IEC / IEEE 42010:2012: Systems and software engineering — Architecture description*. Swedish Standards Institute, Stockholm, swedish standard edition, 2012.

[40] International Organization for Standardization. *ISO / IEC 27032:2012: Information technology — Security techniques — Guidelines for cybersecurity*. International Organization for Standardization, Geneva, 2012.

[41] International Organization for Standardization. *ISO / IEC / IEEE 15288:2015: Systems and software engineering — System life cycle processes*. International Organization for Standardization, Geneva, 2015.

[42] International Organization for Standardization. *Information technology — Security techniques — Information security management systems — Overview and vocabulary*. International Organization for Standardization, Geneva, 2018.

[43] International Organization for Standardization. *ISO 26262:2018: Road vehicles — Functional safety*. International Organization for Standardization, Geneva, 2018.

[44] International Organization for Standardization. *ISO/IEC TR 20547:2020: Information technology — Big data reference architecture*. International Organization for Standardization, Geneva, 2020.

[45] ISO/IEC/IEEE 24765:2017. Systems and software engineering — Vocabulary, 2017.

[46] ISO/IEC/IEEE 4201:2011. Systems and software engineering — Architecture description, 2011.

[47] Syed Muslim Jameel, Manzoor Ahmed Hashmani, Hitham Alhussain, Mobashar Rehman, and Arif Budiman. A critical review on adverse effects of concept drift over machine learning classification models. *International Journal of Advanced Computer Science and Applications*, 11(1):206–211, 2020.

[48] J.Eekels and N.F.M.Roozenburg. A methodological comparison of the structures of scientific research and engineering design: their similarities and differences. *Design Studies*, 12:197–203, 2005.

[49] Michael G Kahn, Jeffrey S Brown, Alein T Chun, Bruce N Davidson, Daniella Meeker, Patrick B Ryan, Lisa M Schilling, Nicole G Weiskopf, Andrew E Williams, and Meredith Nahm Zozus. Transparent reporting of data quality in distributed data networks. *Egems*, 3(1), 2015.

[50] Rashidah Kasauli, Eric Knauss, Jennifer Horkoff, Grischa Liebel, and Francisco Gomes de Oliveira Netoa. Requirements engineering challenges and practices in large-scale agile system development. *Systems and Software*, 2020.

[51] Rashidah Kasauli, Rebekka Wohlrab, Eric Knauss, Jan-Philipp Steghofer, Jennifer Horkoff, and Salome Maro. Charting coordination needs in large-scale agile organizations with boundary objects and methodological islands. In *Proc. of the Int. Conf. on Software and System Processes (ICSSP)*, Seoul, South Korea, 2020.

[52] Eric Knauss. The missing requirements perspective in large-scale agile system development. *IEEE Software*, 36(3):9–13, 2019.

[53] Eric Knauss. Constructive master's thesis work in industry: Guidelines for applying design science research. *arXiv preprint arXiv:2012.04966*, 2020.

[54] Philip Koopman, Uma Ferrell, Frank Fratrik, and Michael Wagner. A safety standard approach for fully autonomous vehicles. In *Int. Conf. on Computer Safety, Reliability, and Security*, pages 326–332. Springer, 2019.

[55] Phillippe Kruchten. *Architecture blueprints—the "4+1" view model of software architecture*, volume 12. ACM Press, New York, New York, USA, 1995.

[56] Søren Lauesen. *Software Requirements*. Pearson / Addison-Wesley, 2002.

[57] John D. Lee. Humans and automation: Use, misuse, disuse, abuse. *HUMAN FACTORS, Vol. 50, No. 3,*, page 404–410, 2008.

[58] Timothée Lesort, Massimo Caccia, and Irina Rish. Understanding Continual Learning Settings with Data Distribution Drift Analysis. *arXiv*, pages 1–9, 2021.

[59] Jennifer Lindern and Padmini Subbiah. Under preparation: Deriving contextual definition and requirements from use cases of autonomous drive. Master's thesis, The University of Gothenburg, Sweden, 2021.

[60] Xiaoqiang Ma, Tai Yao, Menglan Hu, Yan Dong, Wei Liu, Fangxin Wang, and Jiangchuan Liu. A survey on deep learning empowered iot applications. *IEEE Access*, 7:181721–181732, 2019.

[61] Patrick Mäder, Paul L. Jones, Yi Zhang, and Jane Cleland-Huang. Strategic traceability for safety-critical projects. *IEEE Software*, 30, 2013.

[62] JB Manchon, Mercedes Bueno, and Jordan Navarro. From manual to automated driving: how does trust evolve? *Theoretical Issues in Ergonomics Science*, pages 1–27, 2020.

[63] Vladimir Marchenko. Process isolation with arm fusa runtime system.

[64] Salome Honest Maro, Jan-Philipp Steghöfer, Eric Knauss, Jennifer Horkoff, Rashidah Kasauli, Jesper Lysemose Korsgaard, Florian Wartenberg, Niels Jørgen Strøm, and Ruben Alexandersson. Managing traceability information models: Not such a simple task after all. *IEEE Software*, 2020.

[65] Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, and Stefan Wagner. Software Engineering for AI-Based Systems: A Survey. *Preprint*, 1(1), 2021.

[66] Matti Perttula Mikko Salonen, Claus Thorp Hansen. Evolution of property predictability during conceptual design. *ICED 05, Melbourne, August 15–18, 2005*, 2005.

[67] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proc. of the Conf. on Fairness, Accountability, and Transparency*, page 220–229, New York, NY, USA, 2019. Association for Computing Machinery.

[68] S. Ahmed-Kristensen M.N. Sudin and M.M. Andreasen. The role of a specification during the design process: A case study. *INTERNATIONAL DESIGN CONFERENCE - DESIGN 2010*, 2010.

[69] Azana Hafizah Mohd Aman, Elaheh Yadegaridehkordi, Zainab Senan Attarbashi, Rosilah Hassan, and Yong-Jin Park. A Survey on Trend and Classification of Internet of Things Reviews. *IEEE Access*, 8:111763–111782, 2020.

[70] Henry Muccini and Karthik Vaidhyanathan. Software Architecture for ML-based Systems: What Exists and What Lies Ahead. *Proceedings of the 43rd International Conference on Software Engineering,*, mar 2021.

[71] Anitha Murugesan, Sanjai Rayadurgam, and Mats Heimdahl. Requirements reference models revisited: Accommodating hierarchy in system design. *Proceedings of the IEEE International Conference on Requirements Engineering*, 2019-September:177–186, 2019.

[72] Bashar Nuseibeh. Weaving Together Requirements and Architectures. *Computer*, 34(3):115–119, 2001.

[73] Patrizio Pelliccione, Eric Knauss, Rogardt Heldal, S. Magnus Ågren, Piergiuseppe Mallozzi, Anders Alminger, and Daniel Borgentun. Automotive Architecture Framework: The experience of Volvo Cars. *Journal of Systems Architecture*, 77:83–100, 2017.

[74] K Pfeffers, Tuure Tuunanen, Charles E Gengler, Matti Rossi, Wendy Hui, Ville Virtanen, and Johanna Bragge. The design science research process: A model for producing and presenting information systems research. In *Proc. of the First Int. Conf. on Design Science Research in Information Systems and Technology (DESRIST 2006), Claremont, CA, USA*, pages 83–106, 2006.

[75] SHAMEER KUMAR PRADHAN and SAGAR TUNGAL. Quality attributes of data in distributed deep learning architectures. 2021.

[76] Shameer Kumar Pradhansagar and Sagar Tungal. Under preparation: Quality attributes of data in distributed deep learning architectures. Master's thesis, The University of Gothenburg, Sweden, 2021.

[77] Supriya Rao, Eric Knauss, Md Abdullah Al Mamun, and Amna Pir Muhammad. Managing requirements-knowledge for developing cloud-based support of autonomous vehicles and transportation as a service: A design science research. *Systems and Software*, 2021. In review.

[78] P. P. Ray. A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, 30(3):291–319, 2018.

[79] Kirsten Revell, Pat Langdon, Mike Bradley, Ioannis Politis, James Brown, and Neville Stanton. User centered ecological interface design (uceid):a novel method applied to the problem of safe and user-friendly interaction between drivers and autonomous vehicles. *Intelligent Human Systems Integration,Advances in Intelligent Systems and Computing*, 2018.

[80] Stuart Russel. *Human Compatible: AI and the Problem of Control*. Penguin Books, 2020.

[81] SAE. SAE J3016:201806 - SURFACE VEHICLE RECOMMENDED PRACTICE - Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, 2018.

[82] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall, 2002.

[83] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean François Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems*, 2015-January:2503–2511, 2015.

[84] Valerie Sessions and Marco Valtorta. The effects of data quality on machine learning algorithms. *ICIQ*, 6:485–498, 2006.

[85] Ian Sommerville. Software engineering 9, 2009.

[86] David G. Ullman. *The mechanical design process*. McGraw Hill, 2003.

[87] Karl T. Ulrich and Steven D. Eppinger. *Product design and development*. McGraw Hill, 2007.

[88] Peter-Paul van Maanen, Jasper Lindenberg, and Mark A. Neerincx. Integrating human factors and artificial intelligence in the development of human-machine cooperation. *IC-AI 2005*, 2005.

[89] Andreas Vogelsang and Markus Borg. Requirements engineering for machine learning: Perspectives from data scientists. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pages 245–251. IEEE, 2019.

[90] Zhiyuan Wan, Xin Xia, David Lo, and Gail C. Murphy. How does Machine Learning Change Software Development Practices? *IEEE Transactions on Software Engineering*, 2020.

[91] Michael Weiss and Paolo Tonella. Fail-Safe Execution of Deep Learning based Systems through Uncertainty Monitoring. *ICST*, 2021.

[92] Michael Weyrich and Christof Ebert. Reference architectures for the internet of things. *IEEE Software*, 33(1):112–116, 2016.

[93] Oliver Willers, Sebastian Sudholt, Shervin Raafatnia, and Stephanie Abrecht. Safety concerns and mitigation approaches regarding the use of deep learning in safety-critical perception tasks. In *International Conference on Computer Safety, Reliability, and Security*, pages 336–350. Springer, 2020.

[94] Rebekka Wohlrab, Eric Knauss, and Patrizio Pelliccione. Why and how to balance alignment and diversity of requirements engineering practices in automotive. *Systems and Software*, 162, 2019.

[95] Rebekka Wohlrab, Eric Knauss, Jan-Philipp Steghöfer, Salome Maro, Anthony Anjorin, and Patrizio Pelliccione. Collaborative traceability management: A multiple case study from the perspectives of organization, process, and culture. *Requirements Engineering (REEN)*, 25:21–45, 2020.

[96] Rebekka Wohlrab, Patrizio Pelliccione, Eric Knauss, and Mats Larsson. Boundary objects and their use in agile systems engineering organizations. *Journal of Software: Evolution and Process*, 31:1–24, 2019.

[97] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[98] Eoin Woods. Software Architecture in a Changing World. *IEEE Software*, 33(6):94–97, 2016.

[99] Gioele Zardini, Dejan Milojevic, Andrea Censi, and Emilio Frazzoli. A Formal approach to the co-design of embodied intelligence. *arXiv*, 2020.

[100] S. Magnus Ågren, Eric Knauss, Rogardt Heldal, Patrizio Pelliccione, Gösta Malmqvist, and Jonas Bodén. The impact of requirements on systems development speed: a multiple-case study in automotive. *Requirements Engineering*, 24(3):315–340, 2019.