VEDLIoT
Very Efficient Deep Learning in IoT

ICT-56-2020 — Next Generation Internet of Things

# D2.1
# First report on the methods for requirement, specification, performance metrics and verification of context limited AI processing systems

Version 1.00

| Document Information | |
|---|---|
| Contract Number | 957197 |
| Project Website | `https://vedliot.eu/` |
| Dissemination Level | PU (Public) |
| Nature | R (Report) |
| Contractual Deadline | 31 July 2021 |
| Author | UGOT & VEONEER |
| Contributors | Stefan Andersson (VEONEER), Oliver Bruneegard (VEONEER), Olof Eriksson (VEONEER), Hans-Martin Heyn (UGOT), Eric Knauss (UGOT) |
| Reviewers | Gunnar Billung-Meyer, Mario Porrmann |

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

## Change Log

| Version | Date | Description of Change |
|---|---|---|
| v0.01 | 2020-12-01 | Proposed table of contents and template by Veoneer and UGOT. <All> |
| v0.02 | 2021-01-26 | Split up .tex main file into subfiles for each chapter. Fixed table colours on front page. Added Acronym list. Addedexpected content. Added some content to Background. <Hans-Martin Heyn> |
| v0.03 | 2021-06-14 | Rewrote section 1.3 VEDLIoT thing. Added material aboutarchitecture framework. More work needed here. <Hans-Martin Heyn> |
| v0.035 | 2021-06-15 | Added material from 2021-Architecture paper. Somesmaller fixes. <Hans-Martin Heyn> |
| v0.04 | 2021-06-15 | Added material from Olof Eriksson. <Olof Eriksson> |
| V0.05 | 2021-06-16 | Added Terms and definitions. <Hans-Martin Heyn> |
| v0.06 | 2021-06-17 | Requirements method. <Eric Knauss> |
| v0.07 | 2021-06-18 | Intro and Executive summary. <Eric Knauss> |
| v0.08 | 2021-06-20 | Added material to chapter 7. <Oliver Brunnegard> |
| v0.09 | 2021-06-21 | Added content to Chapter 4 and 5, fixed reference list. <Hans-Martin Heyn> |
| v0.092 | 2021-06-21 | Finalised Chapter 8. <Eric Knauss> |
| v0.10 | 2021-06-21 | Edited for final release. <Hans-Martin Heyn> |
| v0.11 | 2021-06-28 | Included suggestions from Reviewers. <Hans-Martin Heyn> |
| v0.12 | 2021-06-30 | Finalised revision after review round 1. <All> |
| v0.13 | 2021-07-19 | Replaced Figure 1.1 ”VEDLIoT Overview” with an updated version. <Hans-Martin Heyn> |
| v0.14 | 2021-07-30 | Included suggestions from Jens Hagemeyer. <Hans-Martin Heyn> |
| v1.00 | 2021-07-30 | Changed design to fit common VEDLIoT design better. <Hans-Martin Heyn> |

This PDF was generated Friday 30th July, 2021, at 11:10.

# Table of Contents

# List of Figures

## Executive Summary

VEDLIoT supports building complex systems based on very efficient deep learning IoT components. From a system's point of view, these behave slightly 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 optimized. These assumptions include:

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

In this preliminary deliverable, we sketch a requirements method that allows to describe the context of operation, quality attributes (such as security, safety, robustness) for the full system, and functional requirements from the perspective of its customers and users. Based on this, we then suggest an approach for architectural decomposition of requirements that allows to specify VEDLIoT components. This allows to achieve the following:

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

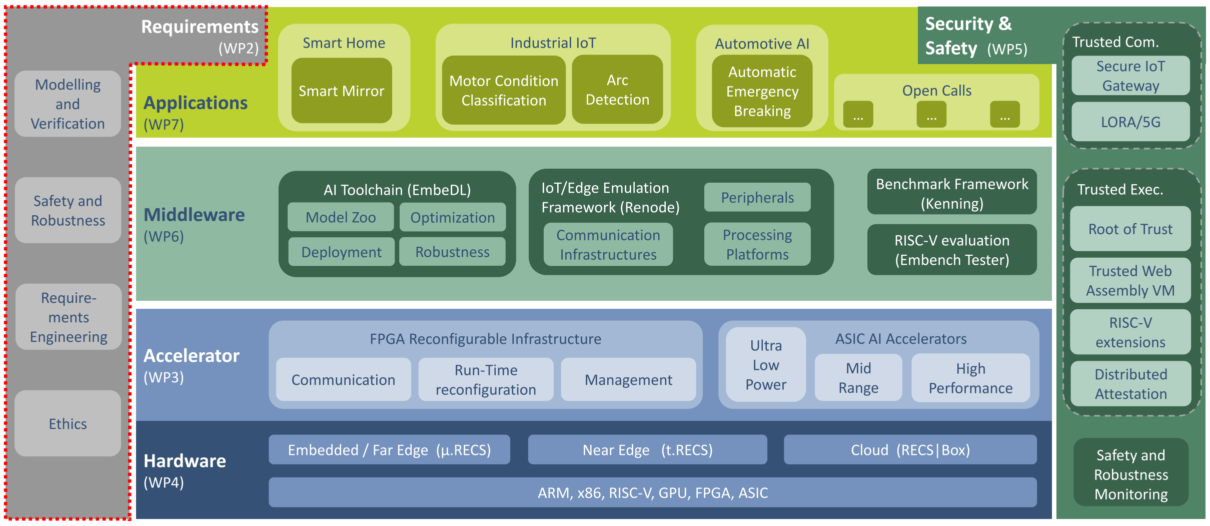


*Figure 1.1. Global picture of VEDLIoT. This deliverable will discuss WP2: Requirements.*

**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 components. 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 optimization toolchain sits between well-known and widely used DL frameworks (e.g. TensorFlow) and backend technologies involving compilers and hardware interfaces. The intermediate representation and optimization 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 specialized methods for defining requirements and architectures for systems built based on VEDLIoT, for specifying components. In addition, novel mechanisms to analyze 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 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 preliminary 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 fulfill 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.

It therefore covers the methods to elicit, analyse, refine, and specify requirements. We have actively arranged our way of working to provide guidance to Task 2.3, which covers the specification of pilots and use cases. Given that both Report D2.1 and D2.3 are published in parallel, only a certain level of consistency can be achieved. We expect to use examples from the pilot and use case specification to a larger extent in D2.5, which is the final report on the methods for requirement, specification, performance metrics and verification of context limited AI processing systems, and which will build on this intermediate report.

In this preliminary report D2.1, we include work in progress. In particular, we are citing Master's theses that are still ongoing and use results from papers that currently are in submission. We encourage readers to look for the updated report, in which the preliminary findings may be replaced by refined versions, and where a number of papers and theses that are currently in submission will be hopefully accepted.

## 1.4 The VEDLIoT Thing

The IEEE 2413:2019 Standard provides an Architectural Framework for the IoT [32]. 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 as-

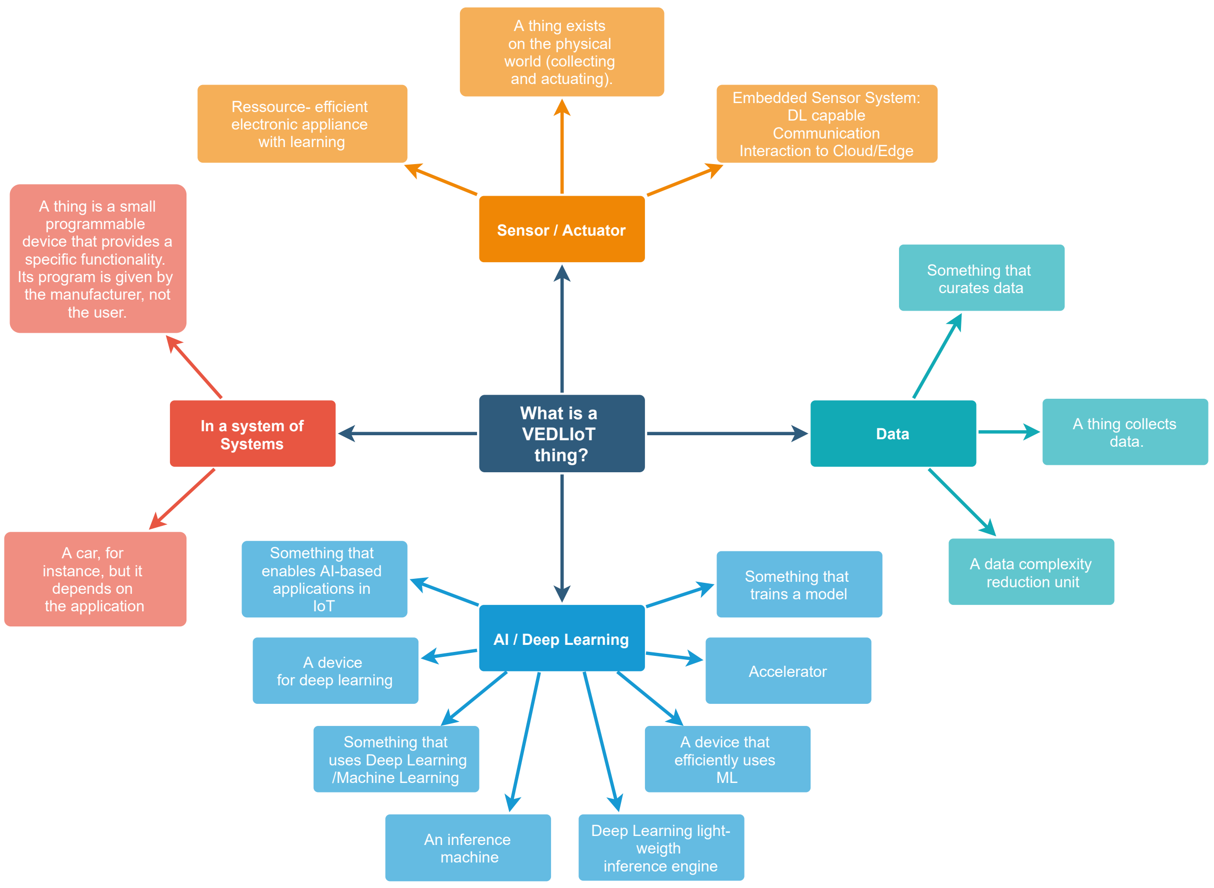

What is a VEDLIoT thing?
Sensor / Actuator
A thing exists on the physical world (collecting and actuating).
Ressource- efficient electronic appliance with learning
Embedded Sensor System: DL capable Communication Interaction to Cloud/Edge
Data
Something that curates data
A thing collects data.
A data complexity reduction unit
In a system of Systems
A thing is a small programmable device that provides a specific functionality. Its program is given by the manufacturer, not the user.
A car, for instance, but it depends on the application
AI / Deep Learning
Something that enables AI-based applications in IoT
A device for deep learning
Something that uses Deep Learning /Machine Learning
An inference machine
Deep Learning light-weigth inference engine
A device that efficiently uses ML
Accelerator
Something that trains a model

*Figure 1.2. Characteristics of "a Thing" in the context of Very Efficient Deep Learning in the IoT.*

sociated 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 [26]. Other examples of IoT components are data storage devices, networking equipment, or processing nodes.

Especially the availability of efficient hardware for massive 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 [58] for a comprehensive survey on Deep Learning empowered IoT applications. Compared to manual feature specification, Deep 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

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

IoT characteristics. The final map is shown in Figure 1.2. 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 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 [31], 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 1.3. 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.

*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 |

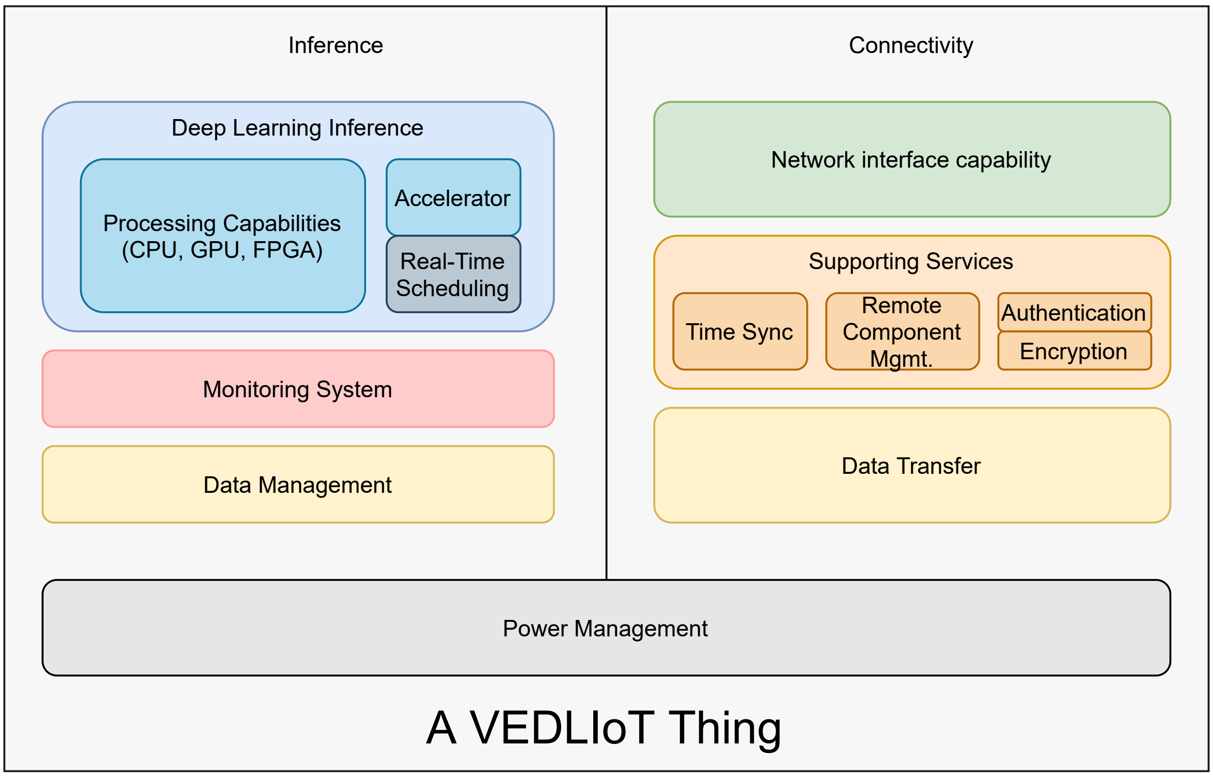


*Figure 1.3. A VEDLIoT Thing*

# 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, specialized methods for defining requirements and architectures need to be developed for VEDLIoT. In addition, novel mechanism to analyze these qualities during development and continuous integration of VELDIoT-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

#### 2.1.1.0 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.

#### 2.1.1.0 Standardisation of architectural frameworks

A natural starting point when discussing system architecture is the ISO 42010 standard on architecture description [37] and ISO 15288 [39] for the life cycle management of an architectural framework. Most terms and definitions in this paper will be taken from these standards.

#### 2.1.1.0 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 [31]. 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 [86, 74], and in the architecture section of [66].

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.

#### 2.1.1.0 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 [42]. 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 |

#### 2.1.1.0 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 [78]. However, as Woods emphasises, traditional architecture frameworks, such as the 4+1 architectural view model by Kruchten [53], does not account for data and algorithm concerns connected to AI component development [92]. 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 [67].

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, 84]. 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) [68]. 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.

#### 2.1.1.0 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 [31]. 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 [31] 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 [31] 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

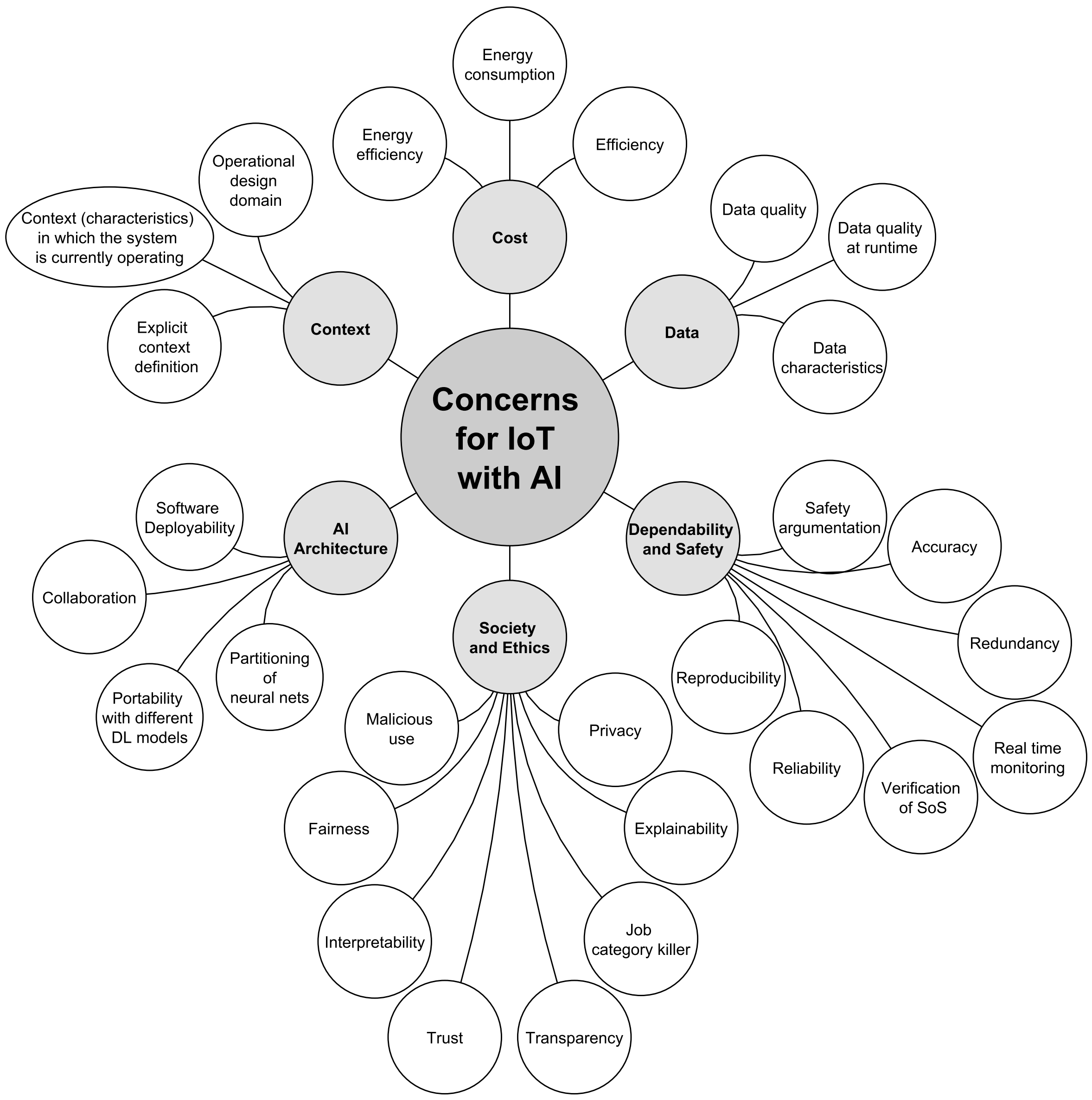


*Figure 2.1. Concerns relevant for systems of the IoT with AI components*

concerns, or are not mentioned by the standard.

The result was collected in a mind-map and, together with the participants, clustered 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 [37] or the IEEE 2413 [31] standard cannot cope with the large set of architectural views necessary to satisfy all

*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 Optimization | ✓ | |
| 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 | | ✓ |

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 frameworks is the lack of a clear system development hierarchy, which would support the early identification and mapping of dependencies between different architectural views [69].

#### 2.1.1.0 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 [44] and the IEEE standard for an architectural framework for the internet of Things (IoT) [32]:

**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" [37];

**architecture description:** "Work product used to express an architecture" [37];

**architecture framework:** "Conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders" [37];

**architecture view:** "Work product expressing the architecture of a system from the perspective of specific system concerns" [37];

**architecture viewpoint:** ”Work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns” [37];

**concern:** ”Interest in a system relevant to one or more of its stakeholders” [37];

**interface:** ”Shared boundary between two functional units, defined by functional characteristics, signal characteristics, or other characteristics as appropriate” [33];

**model kind:** ”Conventions for a type of modelling” [37];

**performance:** ”Quantitative or qualitative level of a property at any point in time considered” [31];

**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” [31];

**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” [31];

**reliability:** ”The ability of an item to perform a required function under given conditions for a given time interval” [35];

**resilience:** ”Ability of a system or component to maintain an acceptable level of service in the face of disruption” [31];

**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” [41];

**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” [38];

**service:** ”The means by which the needs of a consumer are brought together with the capabilities of a provider” [31];

**stakeholder:** ”An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system” [37];

**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” [36];

**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” [36];

**confidentiality:** ”Property that information is not made available or disclosed to unauthorized individuals, entities, or processes.” [40]

**data:** ”Representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by human beings or by automatic means” [31];

**data type:** ”A set of values together with a set of permitted operations” [31];

**device:** ”Independent physical entity capable of performing one or more specified functions in a particular context and delimited by its interfaces” [34];

**entity:** ”A particular thing, such as a person, place, process, object, concept, association, or event” [34];

**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” [31];

**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” [31]

**virtual entity:** ”Computational or data element representing a physical entity” [31]

**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 [62];

**AI component:** ”A part of a system. The component uses primarily AI” [62];

**AI library:** ”A special AI component that provides a concrete implementation of AI algorithms” [62].

#### 2.1.1.0 AI taxonomy

This reports follows the taxonomy proposed in [62] for AI as shown in Figure 2.2

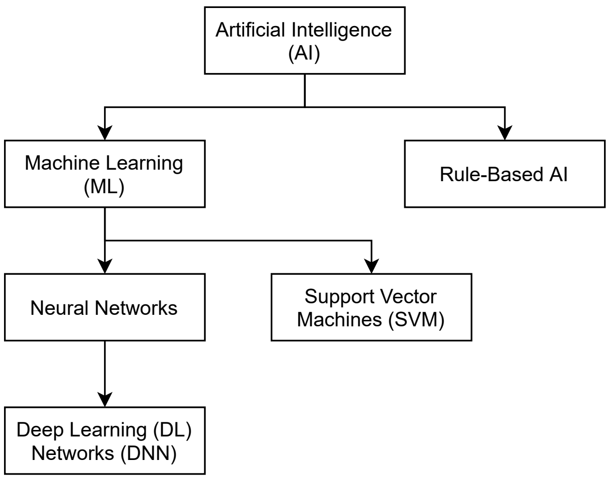

Artificial Intelligence (AI)
Machine Learning (ML)
Rule-Based AI
Neural Networks
Support Vector Machines (SVM)
Deep Learning (DL) Networks (DNN)

*Figure 2.2. Taxonomy for AI based on [62]*

## 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.

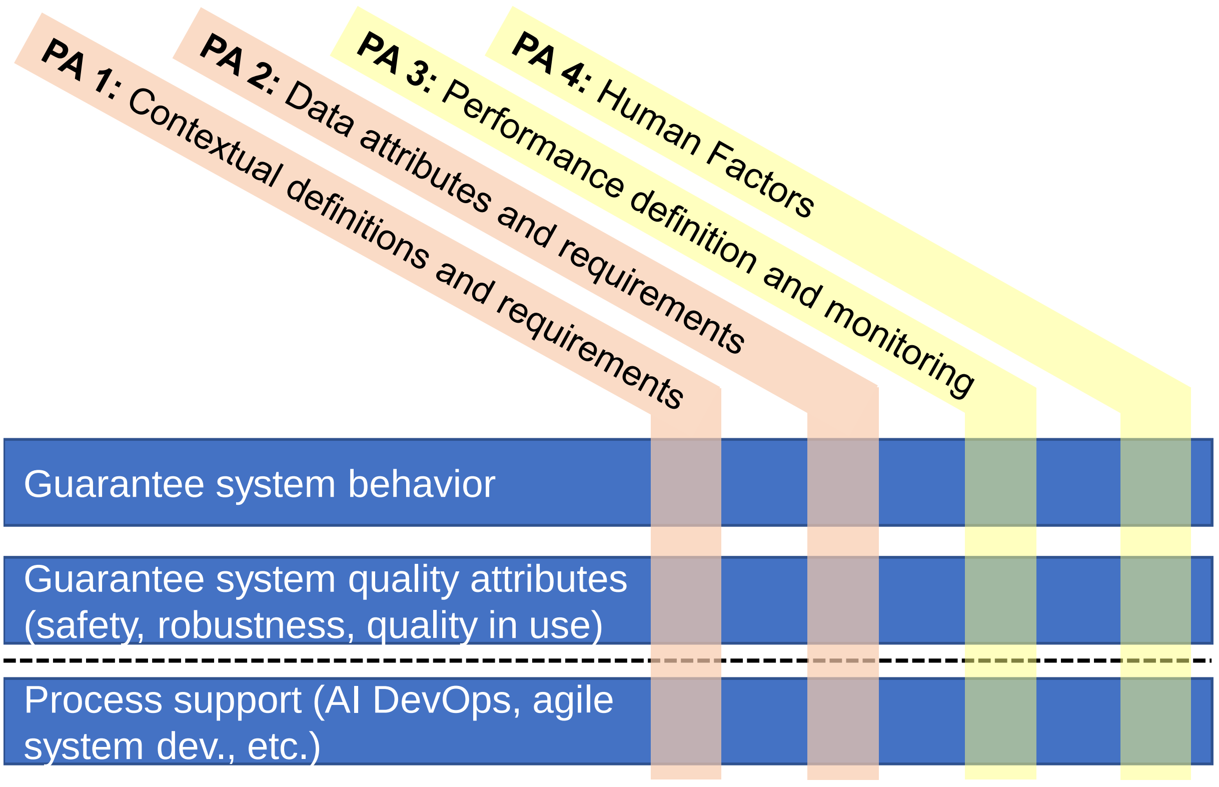

PA 1: Contextual definitions and requirements
PA 2: Data attributes and requirements
PA 3: Performance definition and monitoring
PA 4: Human Factors
Guarantee system behavior
Guarantee system quality attributes (safety, robustness, quality in use)
Process support (AI DevOps, agile system dev., etc.)

*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 [76], 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 [52, 27]. 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 [64].

The importance of proper context definition for machine learning becomes apparent in the *no-free-lunch-theorems* [91]. 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 [24]. 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?

#### 2.2.1.0 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 [87]. 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?

#### 2.2.2.0 Research roadmap

To find answers on the proposed research questions, we intend to combine literature review, data analysis, and a design science research methodology. From the automotive use case, typical data parameters such as precision, timeliness, dynamic range, and noise are 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 [71, 51], the principal goal of the research is 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 serves 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 [79].
- A data quality assessment and monitoring framework [3].
- Characteristics and challenges of big data environments [9].
- Reporting mechanisms of data quality in distributed networks [47].
- Requirements for data quality metrics [30].

### 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, is 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 allows 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 monitoring 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 [85] 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 [55].
- Hancock's warning that human factors must be integrated in automation design [29].
- A list of socio-technical challenges [28].
- A methodology based on multidisciplinary cognitive engineering (CE+) [83].
- The User Centered Ecological Interface Design (UCEID) that enables including HF considerations in the early stages of the overall system design processes [75].

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 building 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 organizations 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 behavior. 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 [60]. 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 increased responsiveness to change [25]. To achieve these goals, organisations rely on

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

empowered, self-organised teams that take responsibility for features from inception, over design, implementation, and test, to deployment [50]. 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 [48].
- A discussion of requirements information models that support collaboration across organizational boundaries [88].
- Considerations of challenges related to defining traceability strategies [59] and traceability information models for modern system development approaches [61, 89, 11, 13].
- Discussions of RE practices at scale [88] and boundary objects for requirements-based coordination [49, 90].

**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 [62]. 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 [70]. An architectural view expresses "the architecture of a system from the perspective of specific system concern" [37]. 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 [68]. 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" [69, 12].

Zardini et al. show how applied category theory supports the co-design of hardware and software for an autonomous driving system [93], 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`

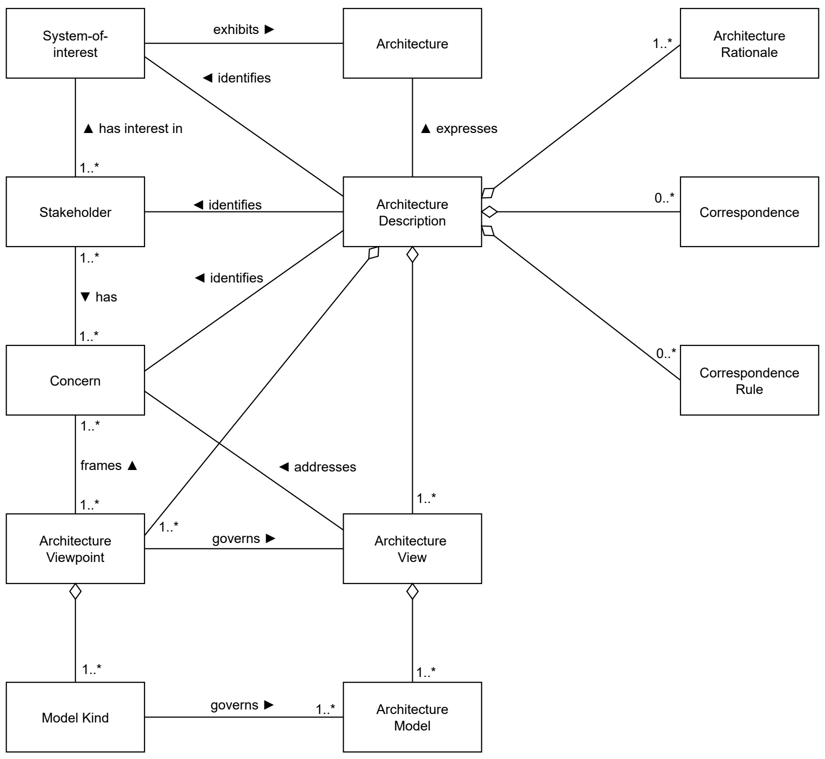


*Figure 3.1. Conceptual model of an architecture description [37]*

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 identifying 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.

## 3.3 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 [27]. 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 [45]. A usual assumption for the training and testing of ML models is that the data are drawn from stationary probability distributions. [56]. 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 [85].

## 3.4 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.

## 3.5 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.

## 3.6 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 [41]. 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 [31], 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".

### 3.6.1 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)"*.

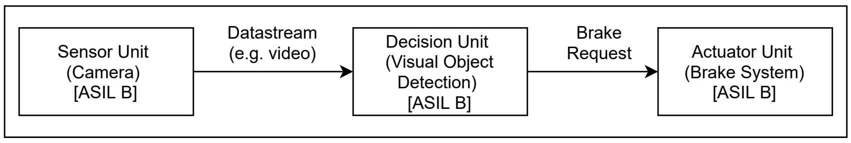


*Figure 3.2. Conceptual system architecture for an automotive automatic emergency brake system*

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

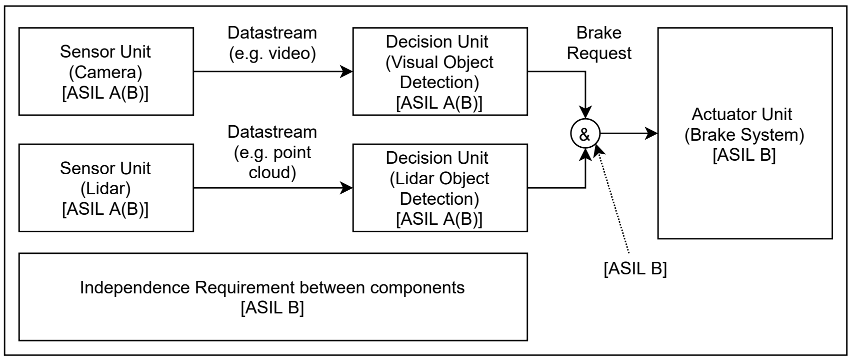


*Figure 3.3. Conceptual system architecture for an automotive automatic emergency brake system after safety decomposition*

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

[2]Automotive Safety Integrity Level

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.

## 3.7 Compositional architecture framework for VEDLIoT

The final conceptual model of a compositional architecture framework based on the stated propositions is illustrated in Figure 3.4. Figure 3.5 presents an compositional architectural framework that includes all earlier identified concerns for distributed AI systems. 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".

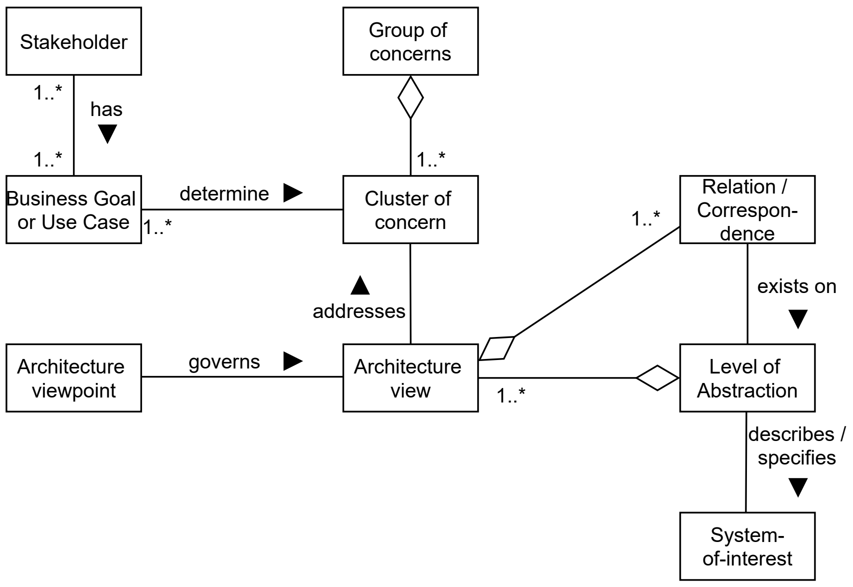
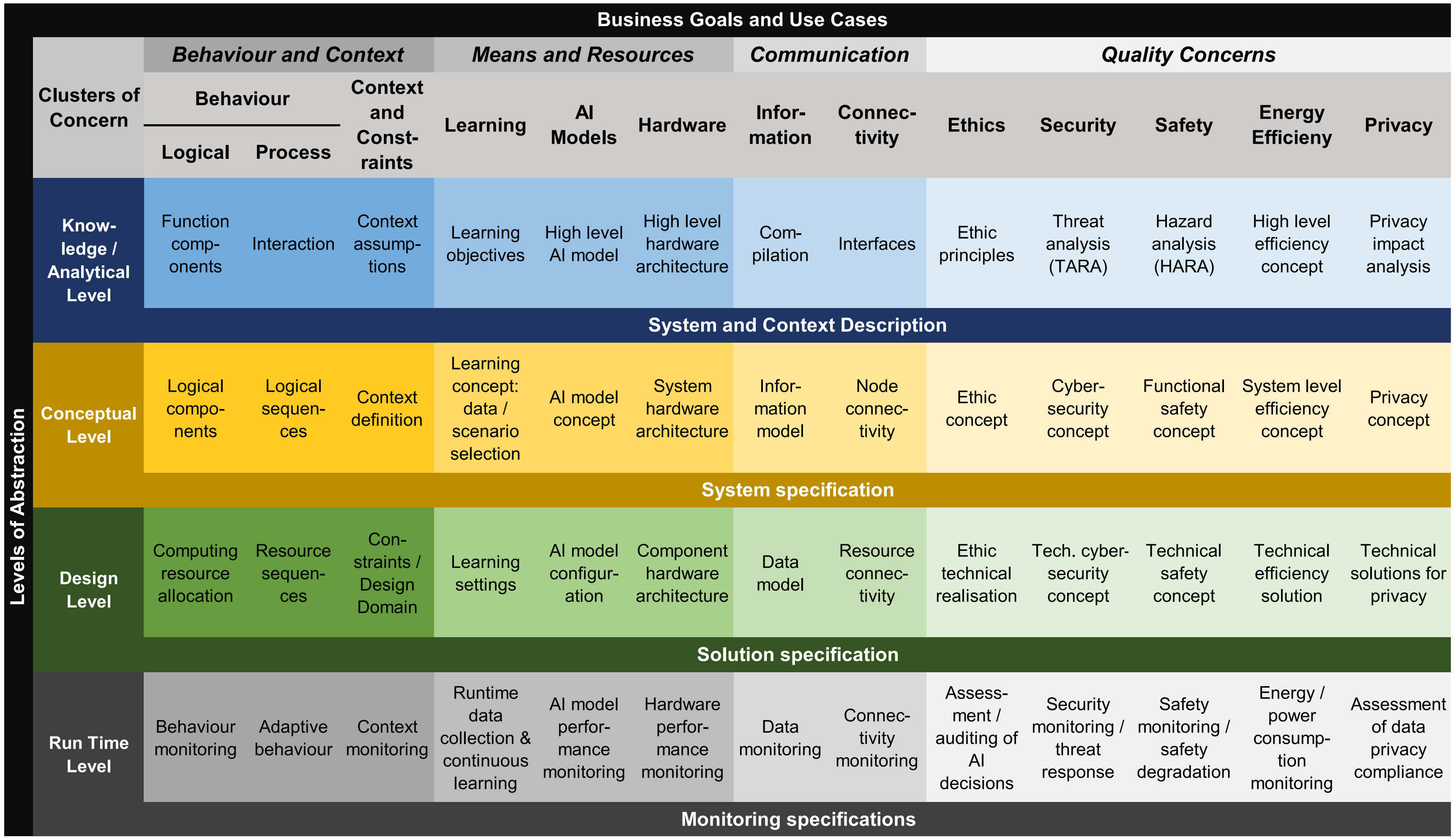


*Figure 3.4. Conceptual model of an compositional architecture framework*

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



| Levels of Abstraction | Business Goals and Use Cases | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Clusters of Concern | *Behaviour and Context* | | | *Means and Resources* | | | *Communication* | | *Quality Concerns* | | | | |
| | | Behaviour | | Context and Const-raints | Learning | AI Models | Hardware | Infor-mation | Connec-tivity | Ethics | Security | Safety | Energy Efficieny | Privacy |
| | | Logical | Process | | | | | | | | | | | |
| | Know-ledge / Analytical Level | Function comp-onents | Interaction | Context assump-tions | Learning objectives | High level AI model | High level hardware architecture | Com-pilation | Interfaces | Ethic principles | Threat analysis (TARA) | Hazard analysis (HARA) | High level efficiency concept | Privacy impact analysis |
| | | System and Context Description | | | | | | | | | | | | |
| | Conceptual Level | Logical compo-nents | Logical sequen-ces | Context definition | Learning concept: data / scenario selection | AI model concept | System hardware architecture | Infor-mation model | Node connec-tivity | Ethic concept | Cyber-security concept | Functional safety concept | System level efficiency concept | Privacy concept |
| | | System specification | | | | | | | | | | | | |
| | Design Level | Computing resource allocation | Resource sequen-ces | Con-straints / Design Domain | Learning settings | AI model configur-ation | Component hardware architecture | Data model | Resource connec-tivity | Ethic technical realisation | Tech. cyber-security concept | Technical safety concept | Technical efficiency solution | Technical solutions for privacy |
| | | Solution specification | | | | | | | | | | | | |
| | Run Time Level | Behaviour monitoring | Adaptive behaviour | Context monitoring | Runtime data collection & continuous learning | AI model perfor-mance monitoring | Hardware perfor-mance monitoring | Data monitoring | Connec-tivity monitoring | Assess-ment / auditing of AI decisions | Security monitoring / threat response | Safety monitoring / safety degradation | Energy / power consump-tion monitoring | Assessment of data privacy compliance |
| | | Monitoring specifications | | | | | | | | | | | | |

*Figure 3.5. Compositional architecture framework for VEDLIoT, categorising views in different clusters of concerns on different levels of abstraction.*

### 3.7.1 Levels of abstraction in the architecture

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.

On the next level of abstraction, the conceptual level, 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 combined on this level provide a system specification that sets the system-of-interest in context and elaborates on how the desired functionality is fulfilled.

The most concrete level at design time of the system is the design level, which includes views that clearly 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 designed. The solution specification concretely describes the final embodiment of the system-of-interest.

Lastly, the run-time level provides views that concern the continuous monitoring and run-time events of the system-of-interest. The views describe how the behaviour of the system shall change under a changing environment, which aspects of the system need to be monitored, and how the system can be updated / continuously improved after deployment. The result is a solution and monitoring specifications, which outline the required monitoring concept for a given system solution.

The connection between the system description and specification is illustrated in Figure 3.6.

### 3.7.2 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.

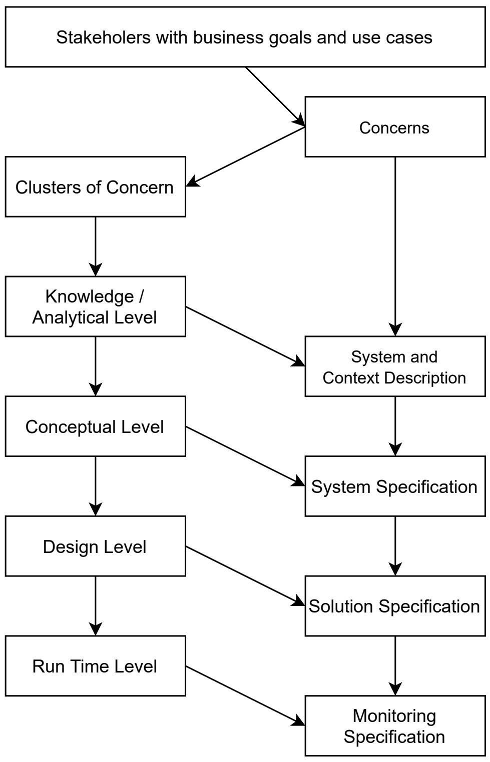


*Figure 3.6. Link between architectural levels of abstraction and specifications*

# 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 externalize 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 organizes 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 6 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, Prioritize Concerns

We perceive the development of a VEDLIoT based system as a complex, multi-organizational, 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 | |

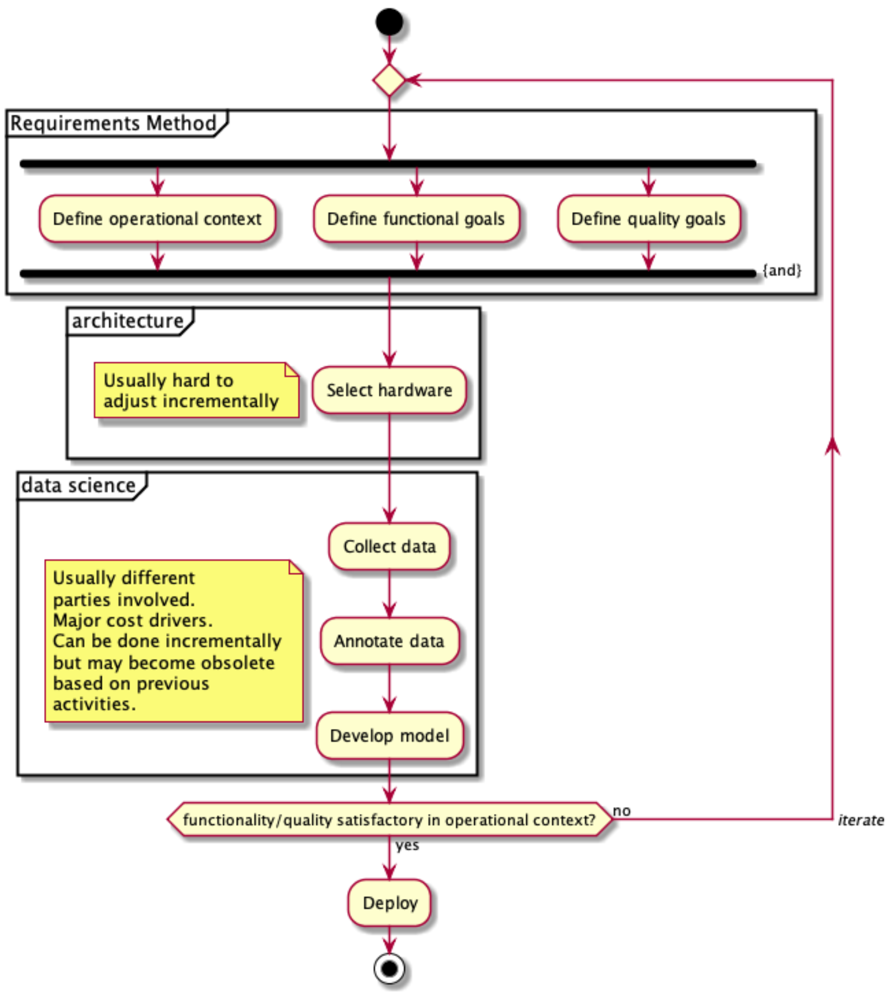

Requirements Method
Define operational context
Define functional goals
Define quality goals
{and}
architecture
Usually hard to adjust incrementally
Select hardware
data science
Usually different parties involved. Major cost drivers. Can be done incrementally but may become obsolete based on previous activities.
Collect data
Annotate data
Develop model
functionality/quality satisfactory in operational context?
no
iterate
yes
Deploy

*Figure 4.1. Basic flow of building VEDLIoT-based systems*

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-organizational 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 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.

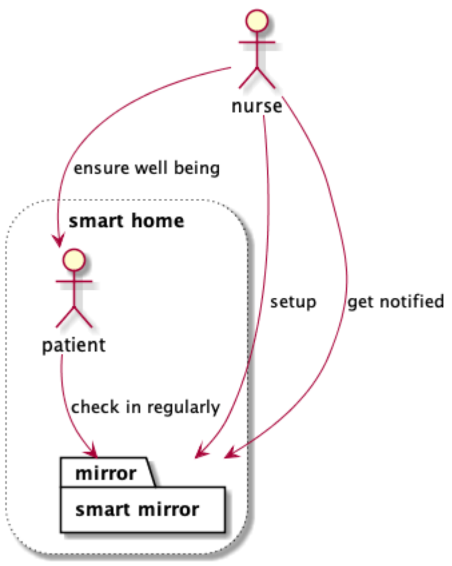


*Figure 4.2. Context diagram, describing scope and high-level functionality of a potential smart mirror based system*

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 utilize 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 optimize 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 [54]. 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 recognize 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 prioritization 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 [73].

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.

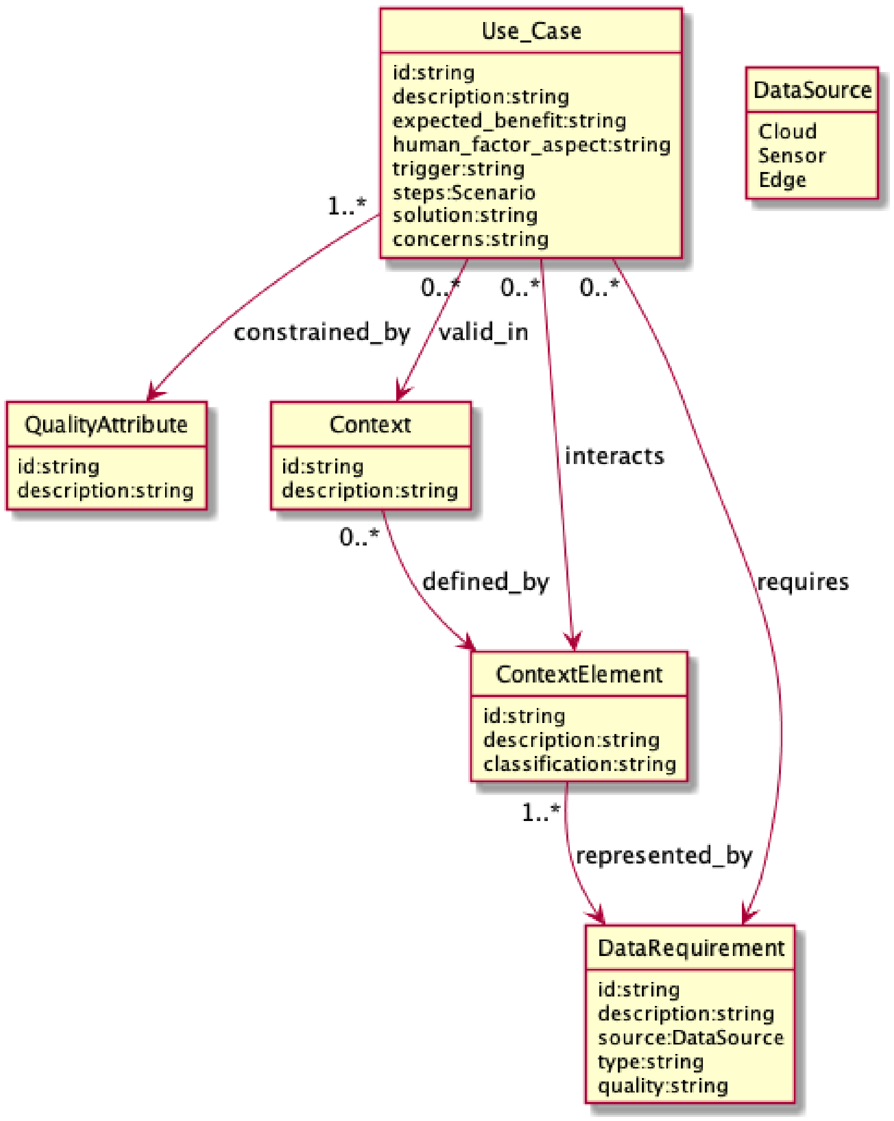


*Figure 4.3. High-level requirements information model that connects use cases to operational context (Operational Design Domain, ODD), Data Requirements, and Quality Requirements.*

*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, pre condition, 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> |

### 4.1.3 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 [23]. 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 prioritize 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.

Lauesen suggests the use of a quality grid [54]. The quality grid is essentially a table with relevant quality attributes in the first column and then an indication on whether

*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<br>UC1.b: Industrial Use Case Switchboard Monitoring<br>UC2: Automotive Use Case<br>UC3: Smart Home Use Case | | | | |
| Comments | [1] Potential ethical concerns are taken seriously in all use cases<br>[2] The automotive use case and one of the industrial use cases concern safety-critical products<br>[3] Depending of context of using the smart mirror, different privacy concerns occur<br>[4] Low power consumption is essential in one of the industrial use cases which is battery based | | | | |

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 a preliminary assessment for the three main use cases in the VEDLIoT project.

Reasons for prioritizing 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 standardization, 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 prioritization 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 then, in turn, 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.

## 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 organize 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.

### 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 [57]:

**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 organizations.

**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 standardized 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.

In terms of solution approaches, we are just at the beginning. 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 prioritized 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, should be a standardised and structured process to engineer requirements and context definitions consistently. Within VEDLIoT, we hope to propose a useful process for the scope of VEDLIoT based systems.

### 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 pre-condition>, 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].

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

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 [54].

> 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 [54], we provide the following example which includes hooks for working with open metrics and open targets.

| 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 [72].

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 fulfillment 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 [77]. Inspired by this, we preliminarily 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, customized))
  - 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?)

We believe that the challenges as well as the data quality attributes are a good starting point to describe data in a meaningful way.

# 5 Performance Metrics

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:

## 5.1 Functional Performance

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

| 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<br>inference including preprocessing |
| Peak performance [ops/s] | Peak performance when executed |
| Achieved performance [Inferences/s] | Performance achieved for a specific DL-model |

## 5.2 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.

| 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 |
| Reconfigu-<br>ration time [s] | Dead time from starting the update<br>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, ... |

## 5.3 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.

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

### 5.3.1 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:

| **Quality Metrics, Full table in [72]** | |
|---|---|
| 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 |

### 5.3.2 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.

# 6 Specification Methods

The VEDLIoT project is focused on systems with distributed AI. Will this affect the way a specification is written? This chapter will try to find any additional inputs or outputs needed for the system, hardware and software specifications due to the selected AI processing.

## 6.1 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 summarizes 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) [43].

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. [65] found the customer or client's specification can be classified into three different areas

- verbal
- semi-developed
- full specification

Ulrich and Eppinger [82] found that a specification consists of a metric and a value. Ullman [81] had 7 areas of classification which are

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

Salonen et al. [63] 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 lifecycle

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

Roozenburg and Eekels [46] have identified a method for making a design specification. The method consists of three phases that are:

- listing objectives
- analyzing of objectives
- editing objectives

In order to achieve a complete collection of objectives and to minimize 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 [94]. Although focusing on software development this may also apply to the full system specification.

## 6.2 Requirement and Specification processes

Most industries have processes and policy documents for the requirement process. The figure below (figure 6.1) is from a Veoneer standard showing the requirement steps as well as the corresponding verification in the typical V process.

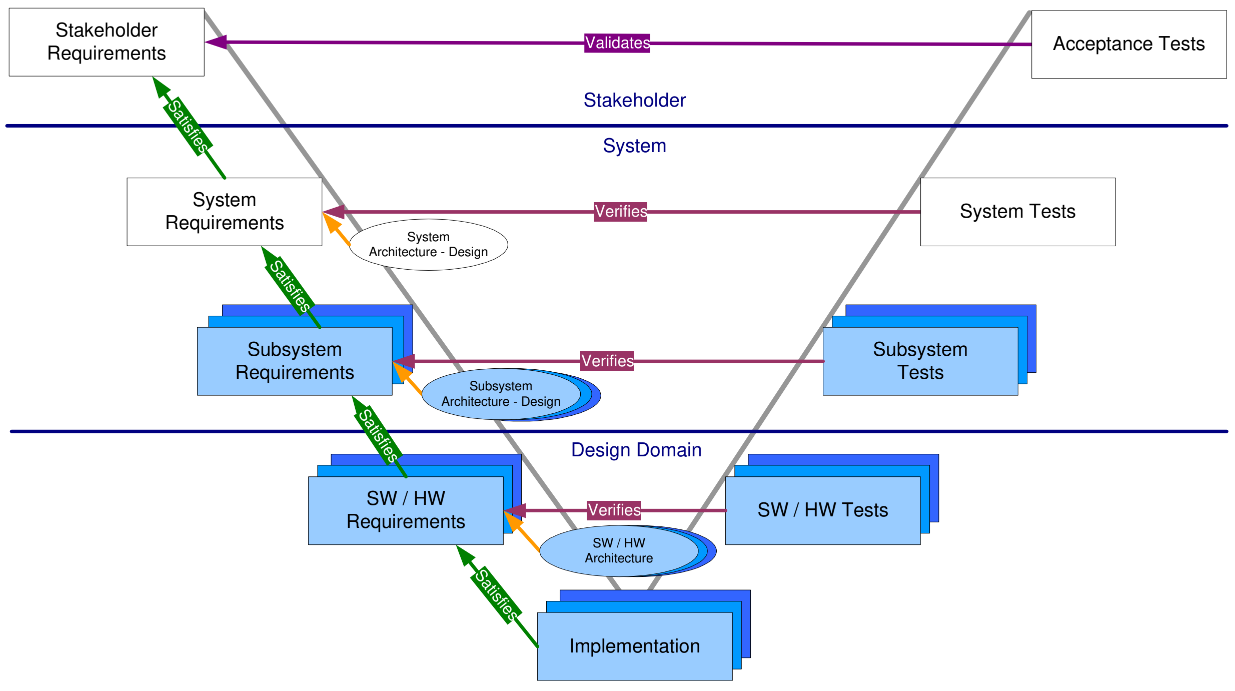


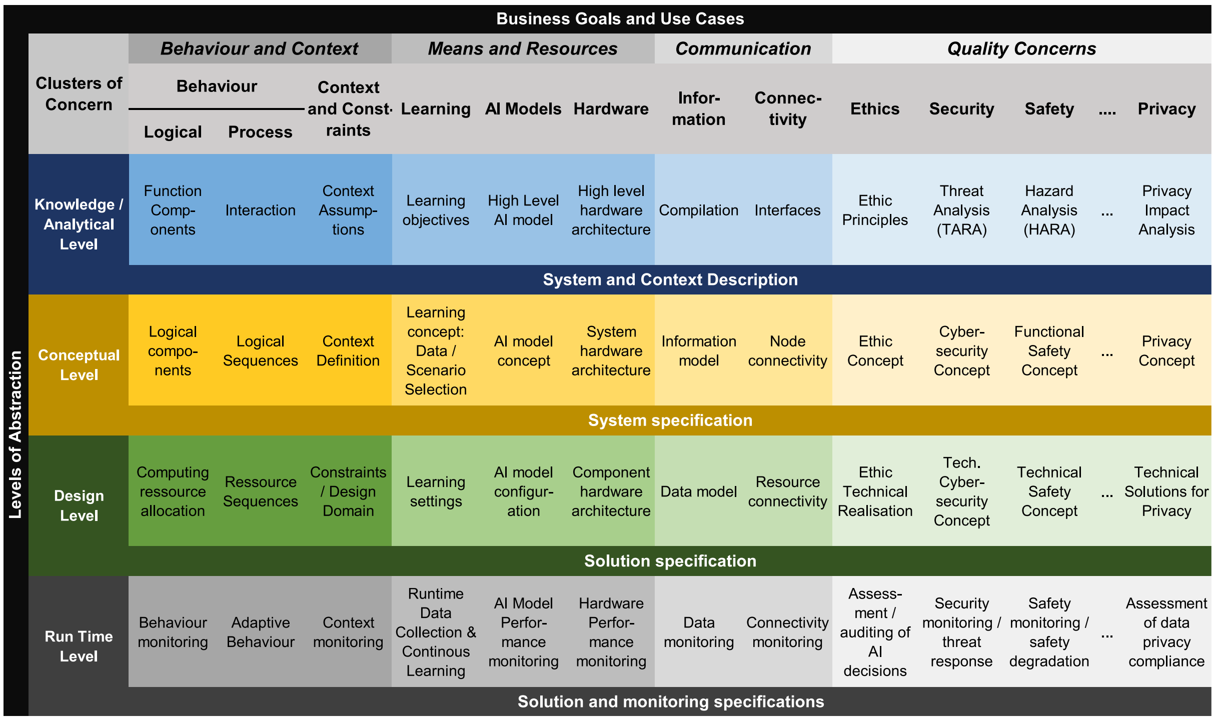
*Figure 6.1. Visualization of Veoneer requirement process*

Analyzing 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 describe by Roozenburg and Eekels [46] are covered by the requirement steps called “Elicit and analyse” and ”Structure and categorize in System level and below”.

As can be seen in Figure 6.1 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 6.1 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 6.2) for convenience.



Business Goals and Use Cases

| Levels of Abstraction | Behaviour and Context | | | Means and Resources | | | Communication | | Quality Concerns | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Clusters of Concern | Behaviour | | Context and Constraints | Learning | AI Models | Hardware | Information | Connectivity | Ethics | Security | Safety | .... | Privacy |
| | Logical | Process | | | | | | | | | | | |
| Knowledge / Analytical Level | Function Components | Interaction | Context Assumptions | Learning objectives | High Level AI model | High level hardware architecture | Compilation | Interfaces | Ethic Principles | Threat Analysis (TARA) | Hazard Analysis (HARA) | ... | Privacy Impact Analysis |
| System and Context Description | | | | | | | | | | | | | |
| Conceptual Level | Logical components | Logical Sequences | Context Definition | Learning concept: Data / Scenario Selection | AI model concept | System hardware architecture | Information model | Node connectivity | Ethic Concept | Cyber-security Concept | Functional Safety Concept | ... | Privacy Concept |
| System specification | | | | | | | | | | | | | |
| Design Level | Computing ressource allocation | Ressource Sequences | Constraints / Design Domain | Learning settings | AI model configuration | Component hardware architecture | Data model | Resource connectivity | Ethic Technical Realisation | Tech. Cyber-security Concept | Technical Safety Concept | ... | Technical Solutions for Privacy |
| Solution specification | | | | | | | | | | | | | |
| Run Time Level | Behaviour monitoring | Adaptive Behaviour | Context monitoring | Runtime Data Collection & Continous Learning | AI Model Performance monitoring | Hardware Performance monitoring | Data monitoring | Connectivity monitoring | Assessment / auditing of AI decisions | Security monitoring / threat response | Safety monitoring / safety degradation | ... | Assessment of data privacy compliance |
| Solution and monitoring specifications | | | | | | | | | | | | | |

*Figure 6.2. An architectural framework merging clusters of concerns for an AI enabled system*

## 6.3 Technology neutral sub system definitions

As indicated by the proposed architectural framework in chapter 3 (Figure 6.2) and in the V-model in Figure 6.1 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 6.2)

## 6.4 Synchronization requirements

In the proposed architectural framework shown in Figure 6.2, 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 fulfill 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.

## 6.5 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 6.3 below).

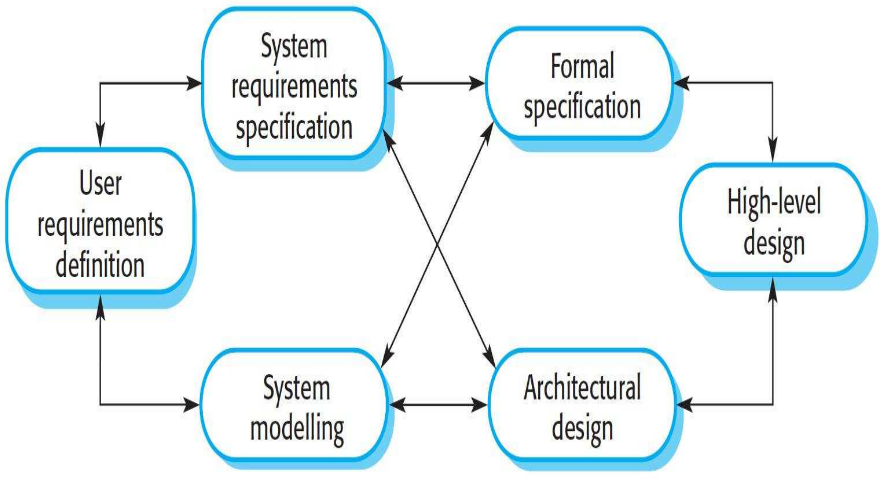


*Figure 6.3. An architectural formal specification as part of requirements specification [80]*

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

Referencing back to the architectural framework in figure 6.2, 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 fulfill the solution requirements and it is also based on a preferred architecture. 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 fulfillment of the requirements at all abstraction levels.

The implementation of the system shall now be based on the specification. Some requirement 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 optimized.

AI models may be optimized 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 fulfill 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.

## 6.6 Next steps

The task of developing the architectural framework and the specification methods will continue another 6 months within the VEDLIoT project. During this time, more effort will be put in structuring the methods taking the output from the proposed "cluster of concern" method and creating the system as well as the hardware and software specifications. This includes for example how to define the specifications for support functions like a possible operating system that manages distributed AI, the system power control and power-on behavior.

# 7 Verification Methods

This report has so far discussed the proposed VEDLIoT architectural framework, methodology for establishing the requirements, relevant performance metrics for DL IoT systems as well as methodology for creating the specification. This chapter will discuss how to tie those aspects together and verify the final work product by validating the fulfillment of the specification through measurable metrics. In the proposed VEDLIoT architectural framework, demonstrated in Figure 6.2, it's clear that verification methods should be applied on each individual box. 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.

## 7.1 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.

## 7.2 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 6.2, 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*.

## 7.3 Quality verification

Quality verification is today governed by the major standardization 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.

## 7.4 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 recognize 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 colors will have undefined result where the variations could be improperly ignored.

## 7.5 Field tests

At different stages through 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).

## 7.6 Next Steps

As previously outlined in this report, the report is a preliminary work report where improvements will continue during the second period of 6 months to refine the proposed methods. Naturally, the main focus of the first 6 months efforts have been on the methods and processes behind generating the adequate requirements for efficient DL based IoT systems, as a result of the projects early focus to obtain and document the requirements for the use-cases in the project as well as to prepare for the open calls. The second period will focus further on improving the validation and testing methods and integrate the new features in the flow of the overall structure.

# 8 Conclusion

This preliminary report shows the state of our works with respect to requirements engineering and specification method. While progress has been made, we have also been successful in identifying key challenges that must be addressed. These include the lack of reference architectures for systems such as the VEDLIoT based systems that we anticipate in this project.

With respect to such a reference architecture, we realised a conceptual distance between the three VEDLIoT use cases, which makes it difficult to define a single reference architecture for the entire project. 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. 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 preliminary requirements engineering method was defined. The requirements method provides initial input to the highest abstraction level of the architectural framework, preparing the selection and prioritization 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. The main effort for the second period is to improve the integration of the verification methods towards the architectural framework.

There remain however a number of severe challenges that are not easily solved. Awareness of these challenges is crucial when building VEDLIoT based systems and there is hope to make concrete suggestions for addressing them in the final report. In particular, it was found that there is a lack of standardized 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 have 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. These challenges will be further evaluated in the second half of this task.

The framework and concepts described in this work package serve as direct input to other work packages of VEDLIoT. 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 |

# 9 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 *Proc. of the 1998 Int. Conf. on Information Quality.* MIT Cambridge, 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] Martin Glinz. On non-functional requirements. In *Proc. of 15th IEEE Int. RE Conf. (RE)*, pages 21–26, New Delhi, India, 2007.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

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

[30] 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.

[31] 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.

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

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

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

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

[36] 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.

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

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

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

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

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

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

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

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

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

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

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

[52] 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.

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

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

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

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

[57] 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.

[58] 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.

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

[60] 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.

[61] 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.

[62] 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.

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

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

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

[70] 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.

[71] 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.

[72] 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.

[73] 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.

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

[75] 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.

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

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

[78] 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.

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

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

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

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

[83] 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.

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

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

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

[87] 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.

[88] 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.

[89] 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.

[90] 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.

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

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

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

[94] 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.