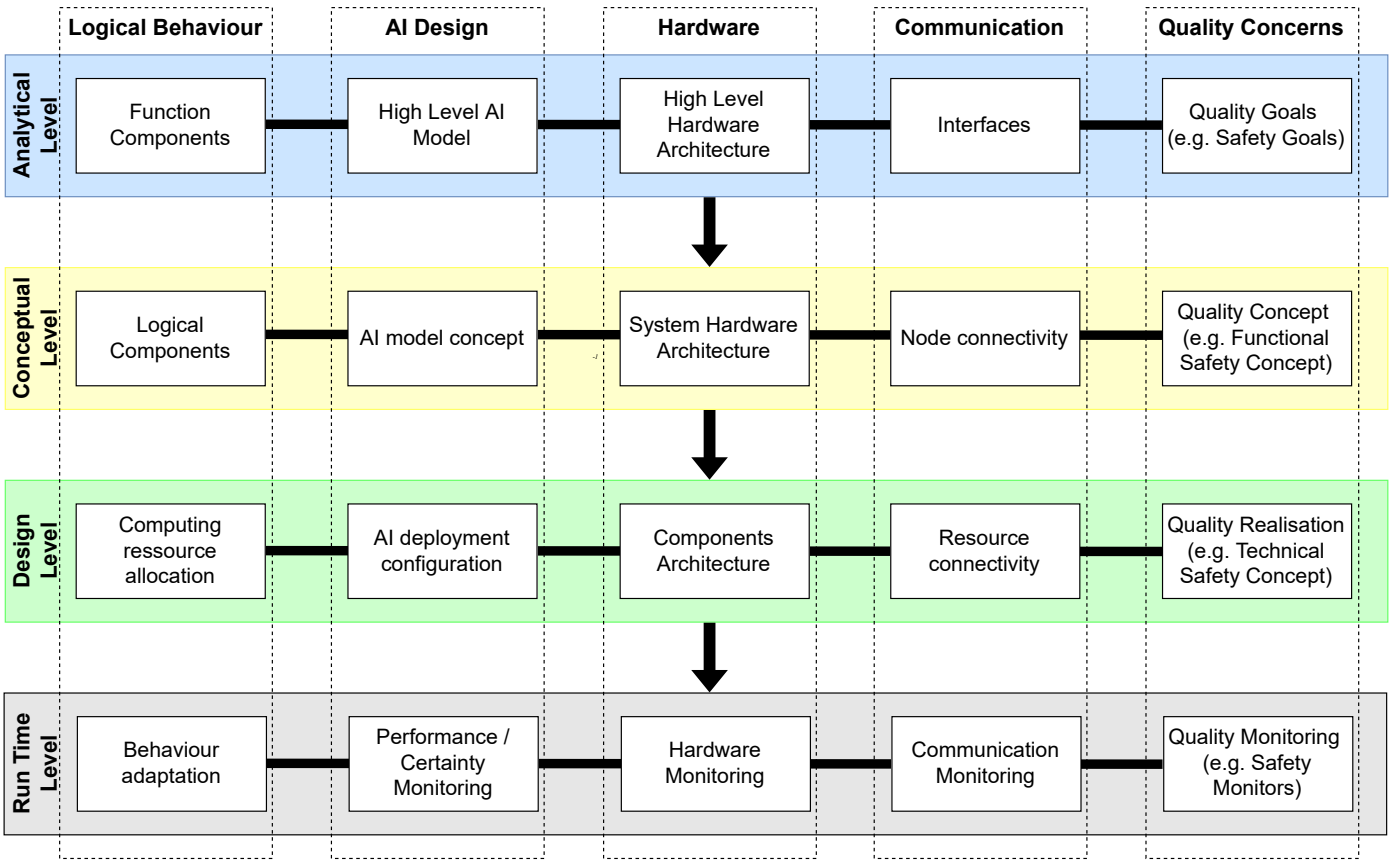
Graphical Abstract

A Compositional Approach to Architecture Frameworks for distributed AI Systems

Hans-Martin Heyn, Eric Knauss, Patrizio Pelliccione



Appendix A. Data produced for this study

This section contains a description of the data collected during each cycle of the design science study.

Problem identification (Section 3)

Literature review. The survey of literature was not conducted as a systematic literature survey. Instead, already existing literature surveys on software engineering method for AI-based system Martínez-Fernández et al. (2021) and IoT architectures Ray (2018) served as starting point. Literature pointing towards challenges and limitation of current architectural approaches were used to identified further relevant prior and derivative work through the tool *Connected Papers*[11].

Workshop 1: Identification of challenges. A first workshop with a focus group of 11 experts from the VEDLIoT project was conducted on 17th February 2021. The list of participants is given in Table 2. The workshop lasted 2 hours. It was conducted through the online meeting tool *Zoom* and used the tool *Mentimeter* for interactive elements. A recording of the workshop can be produced upon request. Prior to the workshop, the participants received a list of stakeholders and challenges as they are mentioned in the IEEE 2413 standard. This list was discussed during the workshop, and additional challenges or stakeholders added if needed.

Workshop 2: Confirmation of challenges. A second workshop with the same focus group from VEDLIoT was conducted on the 31st March 2021. The same participants as listed in Table 2 joined through the online meeting tool *Zoom*. The aim of the second workshop was to validate the findings of the first workshop. The workshop lasted for 2 hours. A recording of the workshop is available.

Creation of a compositional architecture framework for VEDLIoT (Section 5)

Applying the theory of compositional architectural frameworks to VEDLIoT was conducted through bi-weekly meetings with altogether 16 experts from both academia and industry. The meeting occurred every second Thursday between 25th February to 17th June 2021 (8 occurrences). The meetings were conducted online through the meeting tool *Zoom*. Each meeting was recorded. Again, interactive elements such as *Mentimeters* were used to collected opinions from the group of experts.

Review interviews: Experiences from practitioners (Section 6)

After the first four use cases of VEDLIoT applied the architecture framework, we invited the use case owners (i.e., project managers) for interviews to collect feedback on the usefulness and problems when applying the framework. We interviewed one project responsible per use case (i.e., 4 interviewees). Two interviewees were interviewed on-site, two interviewees via a remote Zoom meeting. An interview guide was created beforehand and only available to the interviewer. The interviewer took notes, and recorded the meetings.

Appendix B. Description of the VEDLIoT project

The VEDLIoT project has run since the beginning of 2021 as a cooperation between six academic and five industrial partners from different industry sectors who commonly face challenges when developing AI systems. VEDLIoT aims at creating a toolchain for the development of distributed (IoT) systems that employ deep learning models. Use cases from industry partners serve as demonstrators of the new VEDLIoT toolchain. Modern system development faces two major challenges: First, more and more traditional algorithms are replaced by models based on deep learning. Deep learning has proven to be successful in solving problems of large complexity, such as natural language processing or facial recognition tasks. Second, systems tend to be broken down into different components, in order to be placed where they are most needed and can be most efficient. By establishing high-bandwidth connections between all kind of different devices and allowing many different system configurations, the components of the distributed system become part of what is known as the Internet of Things (IoT).

The aim of the VEDLIoT project is to provide tools, methodologies, and experience for supporting the development and deployment of distributed systems with components that rely on deep learning. The ability to decompose requirements and architecture appropriately is a key enabler for VEDLIoT.

Initially, the VEDLIoT toolchain and methods were applied to four use cases. One use case emerged from the automotive sector (automatic emergency braking), two from industry applications (fault detection in high voltage switches and operation condition classification for electric motors), and one use cases is in the field of smart homes (smart mirror). Over the course of the project, additional use cases will join the project as part of an open call.

[11] https://www.connectedpapers.com/

Appendix C. Mindmap created in the VEDLIoT workshop on challenges with distributed AI system

The mindmap depicted in Figure C.12 was created during a joint industry - academic workshop on the challenges when designing distributed AI systems in VEDLIoT.

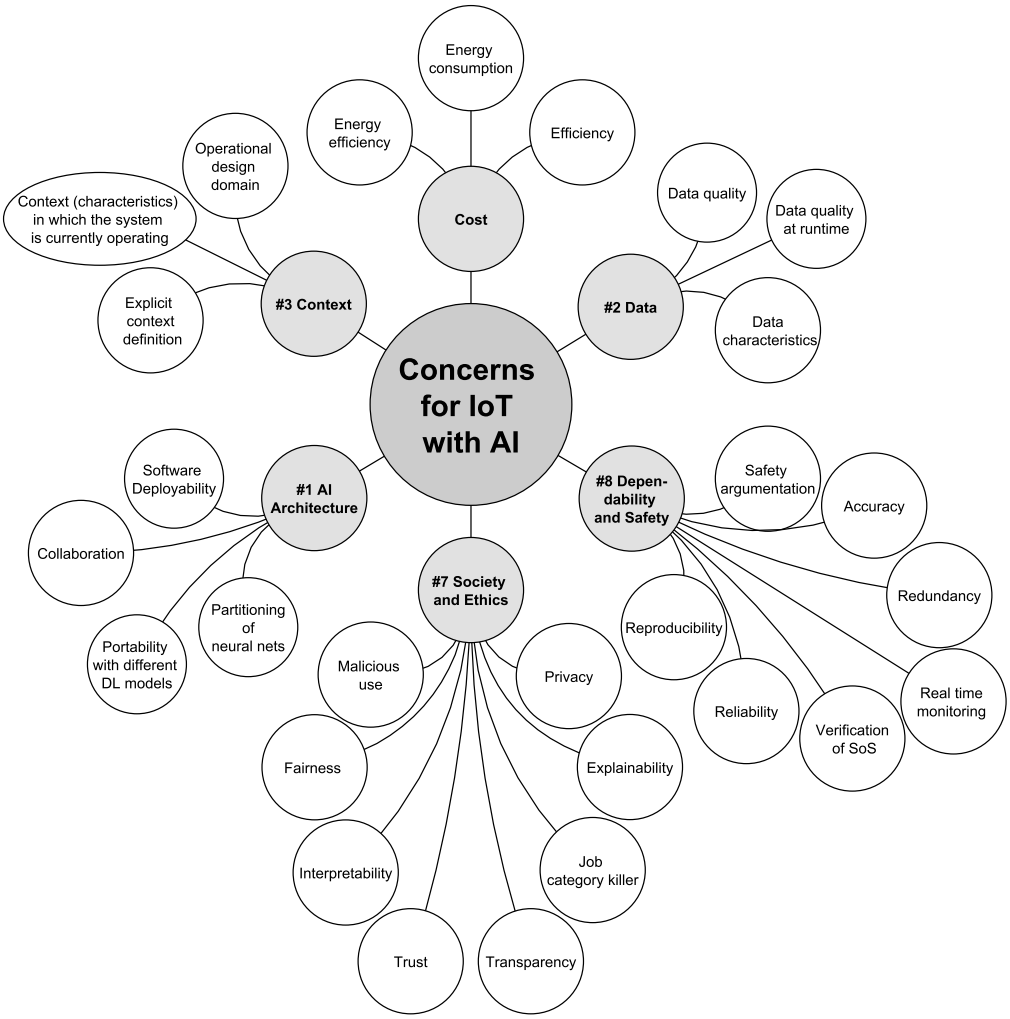


Figure C.12: Concerns relevant for systems of the IoT with AI components.

Appendix D. Concepts from category theory compositional architecture thinking

Category theory can be used to mathematically describe *compositionality*, i.e., "how systems or relationships can be combined to from new systems or relationships" (Fong and Spivak, 2018, Chapter1). System and software architectures consist of architecture building blocks, which combined describe the desired systems (Mitra, 2015). Richards and Ford (2020) suggests a "component-based thinking" for software architectures to easier communicate with developers and other stakeholders. Category theory can provide a mathematical foundation for a component-based construction of architectures, specifically supporting the co-design of different system aspects (Zardini et al., 2020). In our proposed compositional approach to an architecture framework, we utilise four concepts from category theory: *Ordered sets*, *Categories*, *Morphisms*, and *Functors*.

Ordered sets. Fong and Spivak (2018) states that "category theory is all about organising and layering structures", which is why we apply the concept of ordered sets for differentiating and sorting architectural views into different levels of abstraction. A pre-order is a set of elements, that can be sorted such that a) *reflexivity* and b)*transitivity* holds, i.e., a) $a \leq a$ and b) if $a \leq b$ and $b \leq c$, then also $a \leq c$. It just says, that the elements in the set can be sorted by some logic, e.g., how detailed they describe a system aspect. Additionally, because a higher level of abstraction means a loss

of details, i.e., information, the order is *antisymmetric*[12]. Therefore, the set of architectural views in a cluster of concern are a *partial order*.

Categories. A category is a collection of elements, called objects. The type of object is irrelevant, but a pair of objects in a category is connected through *morphisms*. For every object, there is also defined an *identity morphism*. In our case, a category contains architectural views on the same level of abstraction.

Morphisms. Morphisms exist between pairs of object (e.g. a and b) in a category, denoted $f : a \rightarrow b$ and includes a set called "hom-set", indicated as $\mathrm{Hom}(a; b)$. This "hom-set" could for example be a description how to "move" from one architectural view to another. For three (or more) objects a, b, c in a category composition operations are allowed, i.e., $\mathrm{Hom}(a; b)x\mathrm{Hom}(b; c) \rightarrow \mathrm{Hom}(a; c)$.

Functors. Given two categories **A** and **B**, a functor F maps all relations from A to B. It allows us to keep correspondences defined on a higher level of abstraction intact when moving to the next lower level of abstraction.

Appendix E. An antipattern in a compositional architecture framework

The cybersecurity concept in Figure 3 (on the conceptual level) should not have a direct (diagonal) correspondence rule to the component hardware architecture (on the design level). Instead, following Proposition 2, the cybersecurity concept can have a relation to the technical cybersecurity concept (on the design level). According to Proposition 4, this relation must then also exist on the next lower level of abstraction (i.e., between the technical cybersecurity concept and the component hardware architecture). If the higher level correspondence is not mapped onto lower levels, it is questionable why it exists at all (because the correspondence on the conceptual level has in that case no effect on the design level).

[12] If you loose information from a more detailed system view, you cannot reconstruct it from the higher level of abstraction, but vice versa is possible

Appendix F. Description of AI specific architectural views

Table F.7 contains descriptions and examples of architectural views that are specific for the AI component in the VEDLIoT project.

Table F.7: Description of AI specific architectural views. CoC: Cluster of Concern

CoC	View name	Description	Example
Data Strategy	Data ingestion	Contains information on how data is going to be captured and stored.	Definition of a data storage structure and necessary meta information.
	Data selection	Provides rules on how the data is selected for training. Defines necessary data quality attributes.	Specific features of the data are defined and used for filtering the data set. Furthermore, data quality attributes are set, such as image resolution and minimum brightness.
	Data preparation/ manipulation	Describes necessary preparations and modifications on the dataset.	The camera can tilt, therefore the images are rotated. Randomly, a percentage of the pixels are deactivated to increase robustness of detection.
	Runtime data monitoring	Concepts for ensuring that the data specifications are met at runtime.	A monitor is defined that checks the brightness level of the image and the percentage of broken pixels.
Learning	Learning objectives	High level architecture that outlines which functions / models shall be trained through learning.	One function is depicted that shall be able to recognise pedestrians on the road. Three output classes are defined.
	Learning concept	Describes how the learning objective shall be achieved. This can include a description on how a simulation environment is set up, which datasets are required, if transfer learning should be used, etc.	The learning environment is a dataset containing pedestrians on the road at different positions, and empty roads.
	Learning settings	Detailed description of the learning regime's configuration and setup.	Settings for learning, e.g. metric to use, stop condition, loss function, optimiser, batch size, data split for training and testing.
	Manage continous improvements	A view on how continous improvements, i.e., re-training, can be achieved.	AI models can become outdated quickly through changing environments. Continous data collection for retraining can for example be specified here.
AI Model	High level AI model	High Level architecture showing which software components contain an AI model. Also outlines, which kind of AI is intended to be used.	A deep neural network is assigned to the function "Object detection".
	AI model concept	Setup of the AI model outlining the architecture of the AI.	For object detection using a sequential deep neural network with a number of 2D-convolution layers, a 2D-pooling layer, dropout, flatten, and a dense layer. Input, and output definition.
	AI model configuration	Detailed description of the AI model configuration.	For a deep neural network this can be kernel size of convolution layers, dropout rate, activation function, etc.
	AI model performance monitoring	Concept for monitoring the performance of the AI.	For example recording of inference time and uncertainty monitoring.
Context & Constraints	Context assumptions	Assumption on the context (this can be the environment, involved users, etc.).	Object detection shall operate at all times on Swedish roads.
	Context definition	Detailed description of the context in which the desired behaviour will be executed.	Swedish highways and motorways. All possible weather conditions in Sweden. Behaviour should be stable at all times of the day and night.
	Constraints / Design Domain	Lists constraints, and design domain, in which the desired behaviour will be expected.	Object to detect is not more than 1 metre above the road. At night, headlights illuminate road at least 20 metres ahead.
	Context monitoring	Concept for monitoring that the AI operates in the desired context of operation.	System monitors the geographical location of the vehicle (e.g., through GPS) and prevents activation of system if outside of desired context (e.g., Swedish highways and motorways).

Appendix G. Table of morphisms / relations between architectural views

Table G.8 provides examples of correspondences between architectural views in the VEDLIoT architecture framework. The examples are from an extract of the framework used for the automotive use case illustrated in Figure 9. The correspondences are visualised in Figure 11.

Table G.8: Morphisms / relations between the architecture views in the VEDLIoT use case architecture framework extract illustrated Figure 9. LoA: Level of Abstraction, A: Analytical Level, B: Conceptual Level, C: Design Level, D: Runtime Level

ID	LoA	Origin view	Effecting component	Target view	Affected component	Description	Inherited
A-01	A	Context assumptions	context description	Training objectives	classification categories	Definition of pedestrians	
A-02	A	Training objectives	data labels	Data ingestion	strings of data labels	Names of labels	
A-03	A	Context assumptions	context description	High level AI model	output definition	Number of output classes	
B-01	B	Context definition	context description	Training concept	classification categories	Definition of pedestrians	A-01
B-02	B	Training concept	data labels	Data selection	strings of data labels	Names of labels	A-02
B-03	B	Context definition	context description	AI model concept	output layer	number of neurons, type of output classes	A-03
B-04	B	System hardware architecture	camera	Context definition	context description	Position of camera min. height of pedestrian	
B-05	B	System hardware architecture	camera	Data selection	data quality attributes	input data type, resolution	
B-06	B	Context definition	context description	Data selection	feature attributes	feature descriptions	
B-07	B	System hardware architecture	camera	AI model concept	input layer	input data type, input layer size, resolution	
B-08	B	AI model concept	size of DNN	System hardware architecture	processing unit	memory and processing needs	
C-01	C	Design domain	context description	Optimiser settings	classification categories	Definition of pedestrians	B-01
C-02	C	Optimiser settings	data labels	Data preparation	strings of data labels	Names of labels	B-02
C-03	C	Design domain	context description	AI model configuration	output layer	number of neurons, type of output classes	B-03
C-04	C	Component hardware architecture	camera	Design domain	context description	Position of camera min. height of pedestrian	B-04
C-05	C	Component hardware architecture	camera	Data preparation	data quality attributes	input data type, resolution	B-05
C-06	C	Design domain	context description	Data preparation	feature attributes	feature descriptions	B-06
C-07	C	Component hardware architecture	camera	AI model configuration	input layer	input data type, resolution	B-07
C-08	C	AI model configuration	size of DNN	Component hardware architecture	Jetson NX	memory and processing needs	B-08
C-09	C	Component hardware architecture	Jetson NX	AI model configuration	configuration of DNN	model optimisation for hardware	
C-10	C	Component hardware architecture	camera	Data preparation	data manipulation	error rate of pixels	
C-11	C	Design domain	design domain description	Data preparation	data manipulation	random tilting of input image	
C-12	C	AI model configuration	type of DNN	Optimiser settings	Optimiser setings	set of parameters for optimiser	
D-01	D	Context monitoring	context description	Continous improvements	classification categories	Definition of pedestrians	C-01
D-02	D	Continous improvements	data labels	Runtime data monitoring	strings of data labels	Names of labels	C-02
D-03	D	Context monitoring	context description	AI performance monitoring	uncertainty monitoring	number of neurons, type of output classes	C-03
D-04	D	Hardware performance monitoring	camera	Context monitoring	context description	Position of camera min. height of pedestrian	C-04
D-05	D	Hardware performance monitoring	camera	Runtime data monitoring	check resolution	input data type, resolution	C-05
D-06	D	Context monitoring	context description	Runtime data monitoring	data testing	feature descriptions	C-06
D-07	D	Hardware performance monitoring	camera	AI performance monitoring	test and validation	input data type, resolution	C-07
D-08	D	AI performance monitoring	size of DNN	Hardware performance monitoring	hardware watchdog	memory and processing needs	C-08
D-09	D	Hardware performance monitoring	Jetson NX	AI performance monitoring	test and validation	model optimisation for hardware	C-09
D-10	D	Hardware performance monitoring	camera	Runtime data monitoring	check broken pixels	error rate of pixels	C-10
D-11	D	Context monitoring	design domain description	Runtime data monitoring	test and validation	random tilting of input image	C-11
D-12	D	AI performance monitoring	type of DNN	Manage continous improvements	Optimiser setings	set of parameters for optimiser	C-12
D-13	D	Runtime data montoring	check input image	Continous improvements	capturing of new data	quality requirements towards new images	