RE'23 Tutorial

Requirements Analysis and Decomposition for Distributed Systems based on Deep Learning

Eric Knauss, Hans-Martin Heyn

Eric.Knauss@cse.gu.se

Hans-Martin.Heyn@gu.se

IEEE RE'23 Conference

Introduction

- Distributed AI systems combine properties of AI systems with properties of systems in the IoT.
- Challenges from both AI system design and IoT system design need to be accounted for.
- Architecture frameworks provide knowledge structures through architeectural views that help us to solve all relevant challenges in the system design.

Very efficient deep learning in the IoT

Requirements

Modelling & Verification

Safety & Robustness

Applications

Smart Home

Industrial IoT

Automotive AI

Security & Safety

Middleware

Toolchain

embedl

Emulation

RENODE

Benchmarking & Deployment

Kenning

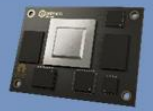
Microserver & Accelerators

Xilinx
Kria

Coral SoM

COM-HPC
Xilinx Zynq
UltraScale+

Jetson AGX
NVIDIA Xavier



RPi CM4
ARVSOM

SMARC
Xilinx Zynq
UltraScale+

Hardware Platforms

Embedded/
Far Edge

u.RECS

Near Edge

t.RECS

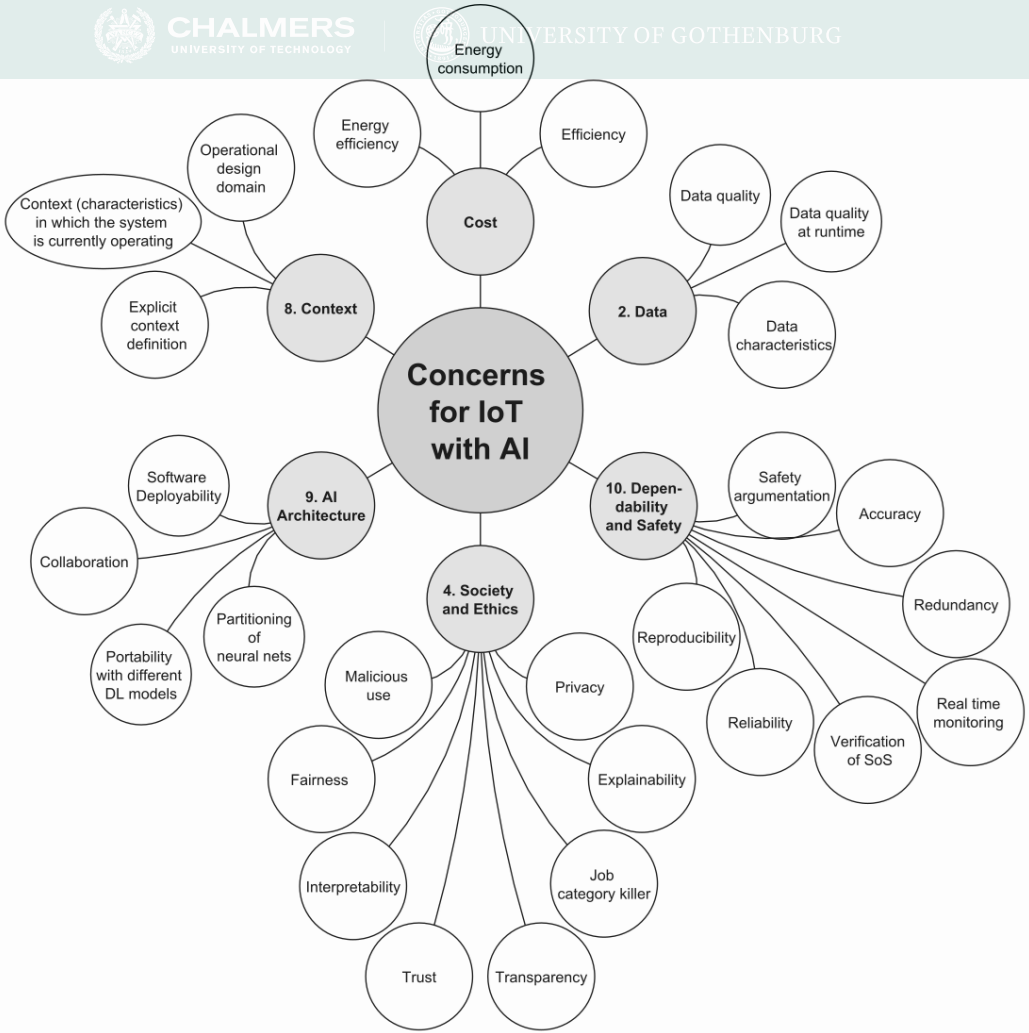
Cloud

RECS|Box

Trusted Execution & Communication

Monitoring

RISC-V extensions



Concerns in (AI) system design

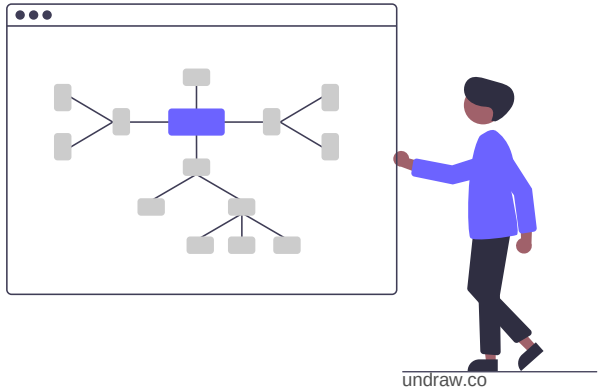
- AI, and especially highly complex DNN, cause more concerns to be included in the system design.
- A reason for the additional concerns can be uncertainty in how to build these systems right.

Co-design of a system

- Co-design can stand for “**collaborative design**”.
- All required stakeholders should be actively involved in the design process.
  - Developers of different disciplines, Data Experts, Customers, Business Owners, …
- See for example
  - Fitzgerald, J., Larsen, P.G., Verhoef, M., 2014. Collaborative design for embedded systems. Academic Press 10, 978-3.
  - Nalchigar, S., Yu, E., Keshavjee, K., 2021. Modeling machine learning requirements from three perspectives: a case report from the healthcare domain. Requirements Engineering 26, 237-254.

Co-design of a system

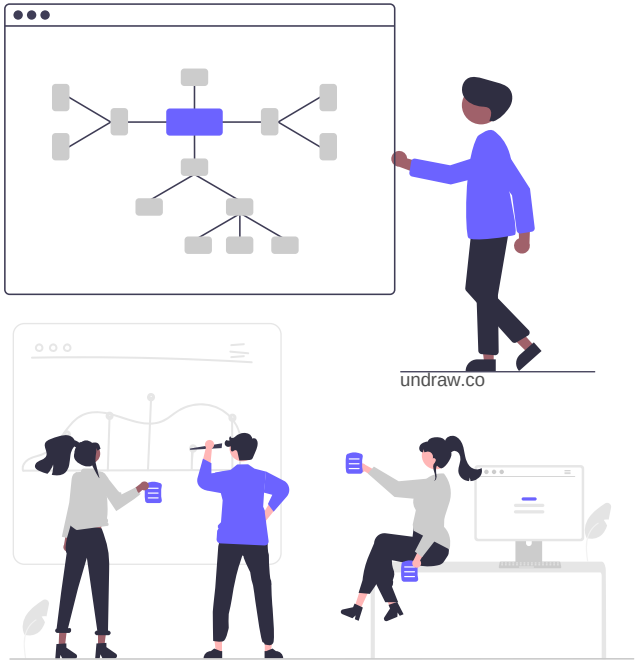
- Co-design can mean “**integrated design**”.
- Different design aspects of the system, e.g., hardware, software, but also quality aspects are closely coupled to each other.
- This can create a high dimensional design space that needs to involve in parallel to ensure “safe by design”, “secure by design”, “fair by design”, or any other “quality by design” necessary.



Co-design of a system

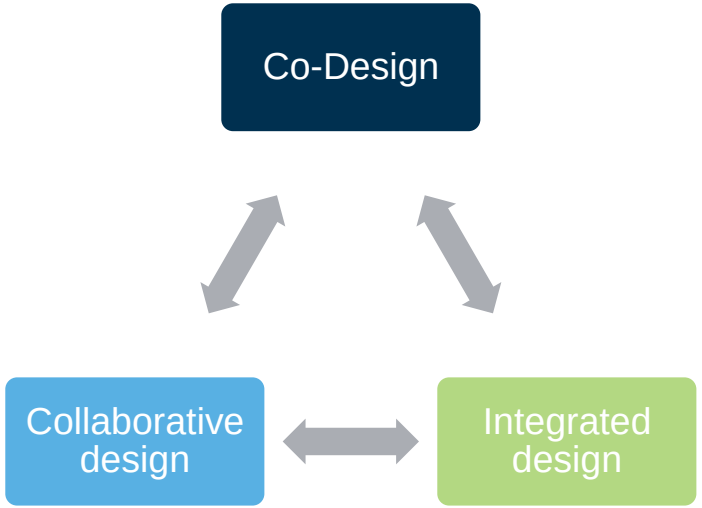
- Designing a complex and distributed system is a hierarchical process.
  - Several, sometimes highly specialized views allow for decomposition of the design task.
  - Requirements and architecture often co-evolve (Twin Peaks).

- Developing complex system is a highly collaborative act between many stakeholders.



Problem definition

- We needed to define an architectural framework, that supports both aspects of co-design.
- The framework must support explicitly aspects of distributed systems (IoT) and AI system development.
  - Learning and data management
- The framework also needed to be flexible enough to cover all current use cases, and new future use cases.
  - A special focus therefore lies on the support of non-functional requirements / quality views
  - Traceability of design decisions
- A single reference architecture would have been too limiting for allowing the variety of (open) use cases in VEDLIoT.



Heyn, H. M., Knauss, E., Muhammad, A. P., Eriksson, O., Linder, J., Subbiah, P., ... & Tungal, S. (2021, May). Requirement engineering challenges for ai-intense systems development. In *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)* (pp. 89-96). IEEE

Current approaches to architecture did not help

- Providing the right learning setting / training data
  - No explicit views on the learning perspective of an AI system in common architecture approaches (Bosch et al., 2020, Muccini et al., 2021).
- Monitoring solutions must be represented explicitly in the architecture
  - Some flaws can only be detected after deployment
  - Therefore, monitoring is needed to ensure functional, and non-functional aspects of an AI system (Bernadri et al., 2019).
- New quality aspects arise, such as “explainability”, or “data privacy”
  - Depending on the use case certain a wide set of quality aspects can be relevant (Habibullah and Horkoff, 2019)
  - New stakeholders need to be included with their own views on the system (Vogelsang and Borg, 2019)

Bosch, J., Olsson, H. H., & Crnkovic, I. (2021). Engineering ai systems: A research agenda. In Artificial Intelligence Paradigms for Smart Cyber-Physical Systems (pp. 1-19). IGI global.

Muccini, H., & Vaidhyanathan, K. (2021, May). Software architecture for ml-based systems: what exists and what lies ahead. In 2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN) (pp. 121-128). IEEE.

Bernardi, L., Mavridis, T., & Estevez, P. (2019, July). 150 successful machine learning models: 6 lessons learned at booking. com. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 1743-1751).

Habibullah, K. M., & Horkoff, J. (2021, September). Non-functional requirements for machine learning: understanding current use and challenges in industry. In 2021 IEEE 29th International Requirements Engineering Conference (RE) (pp. 13-23). IEEE.

Vogelsang, A., & Borg, M. (2019, September). Requirements engineering for machine learning: Perspectives from data scientists. In 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW) (pp. 245-251). IEEE.

Challenges for VEDLIoT system design

- We conducted two workshops with industry and academic partners involved in VEDLIoT in 2021
- 11 participants
- Additionally, we conducted a literature search

No	Area of Expertise	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	Secure conc. for IoT and AI		✓
10	AI Hardware Research		✓
11	Systems Safety Concepts		✓

ID	Description	L	W	Sources
#1	Additional views needed for describing the AI model		✓	
#2	Data requirements for ensuring the desired AI's behaviour must be considered	✓	✓	Kondermann (2013); Ries et al. (2021); Woods (2016)
#3	Description of context and design domain		✓	
#4	Describing the learning setting environment	✓		Bosch et al. (2020); Muccini and Vaidhyanathan (2021); Woods (2016)
#5	Integration of additional stakeholders (e.g., data scientists, policy makers)	✓		Ahmad et al. (2021); Altarturi et al. (2017); Sculley et al. (2015); Vogelsang and Borg (2019)
#6	Management of dependencies and correspondences between views	✓		Nuseibeh (2001)
#7	New quality aspects (e.g., explainability, fairness)	✓	✓	Aydemir and Dalpiaz (2018); European Commission (2020); Habibullah and Horkoff (2021); Horkoff (2019)
#8	Run Time monitoring	✓	✓	Bernardi et al. (2019); Wan et al. (2020)
#9	Support of decomposition into different levels of system design	✓		Giaimo et al. (2010); NATO (2020); Nuseibeh (2001)
#10	Support of middle-out design	✓		Murugesan et al. (2019)

Mentimeter 1

Is it a challenge…

Other challenges?

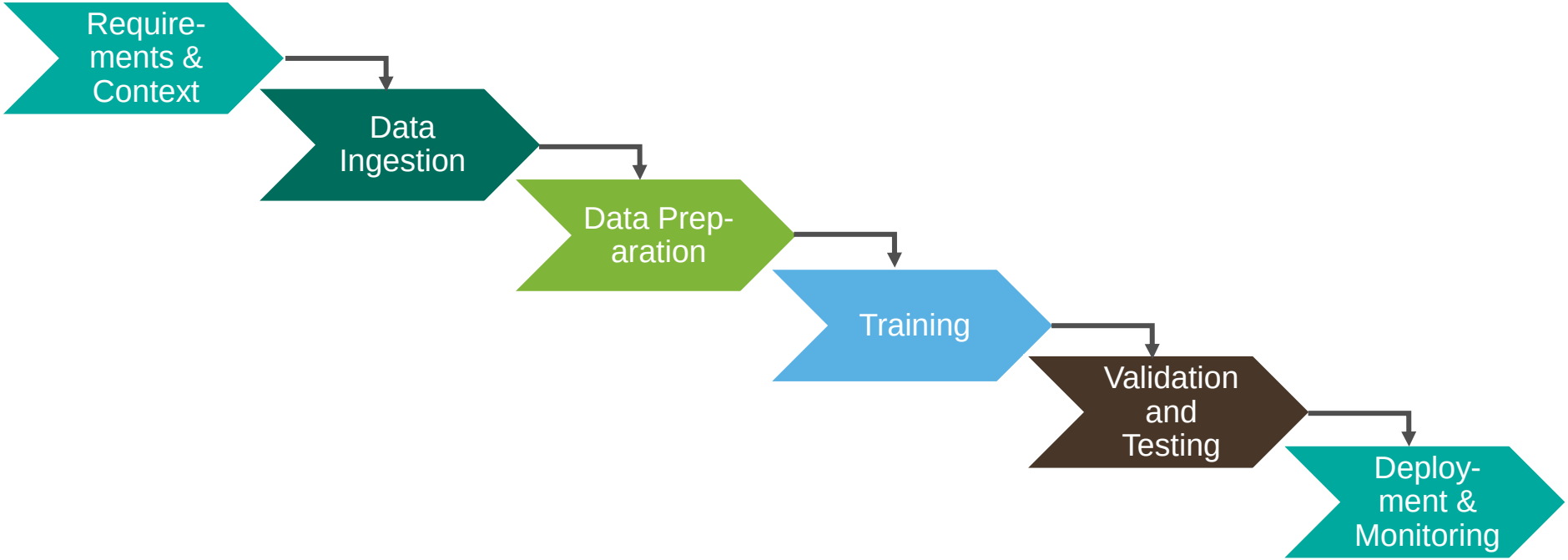
AI Dev. Pipelines and Agile?

- Developing AI systems in an truely agile software project is challenging due to strong dependencies between different steps in a typical ”AI/ML development pipeline”.

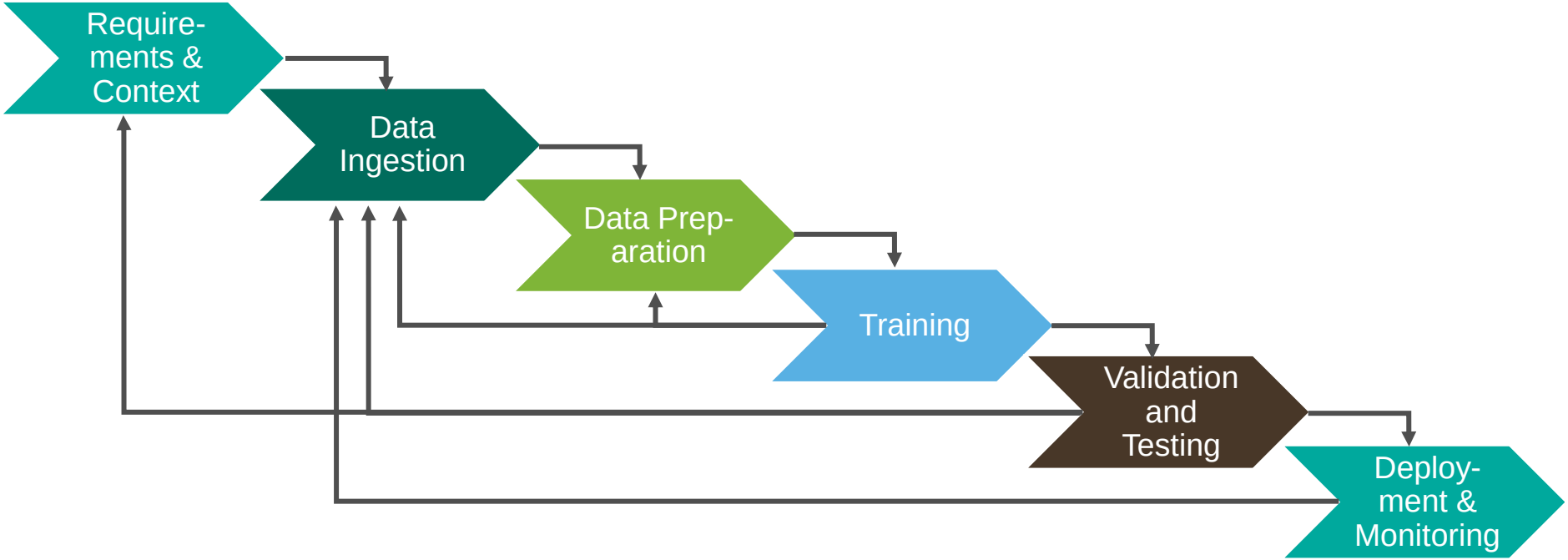
AI Dev. Pipelines and Agile?

- Developing AI systems in an truely agile software project is challenging due to strong dependencies between different steps in a typical ”AI/ML development pipeline”.

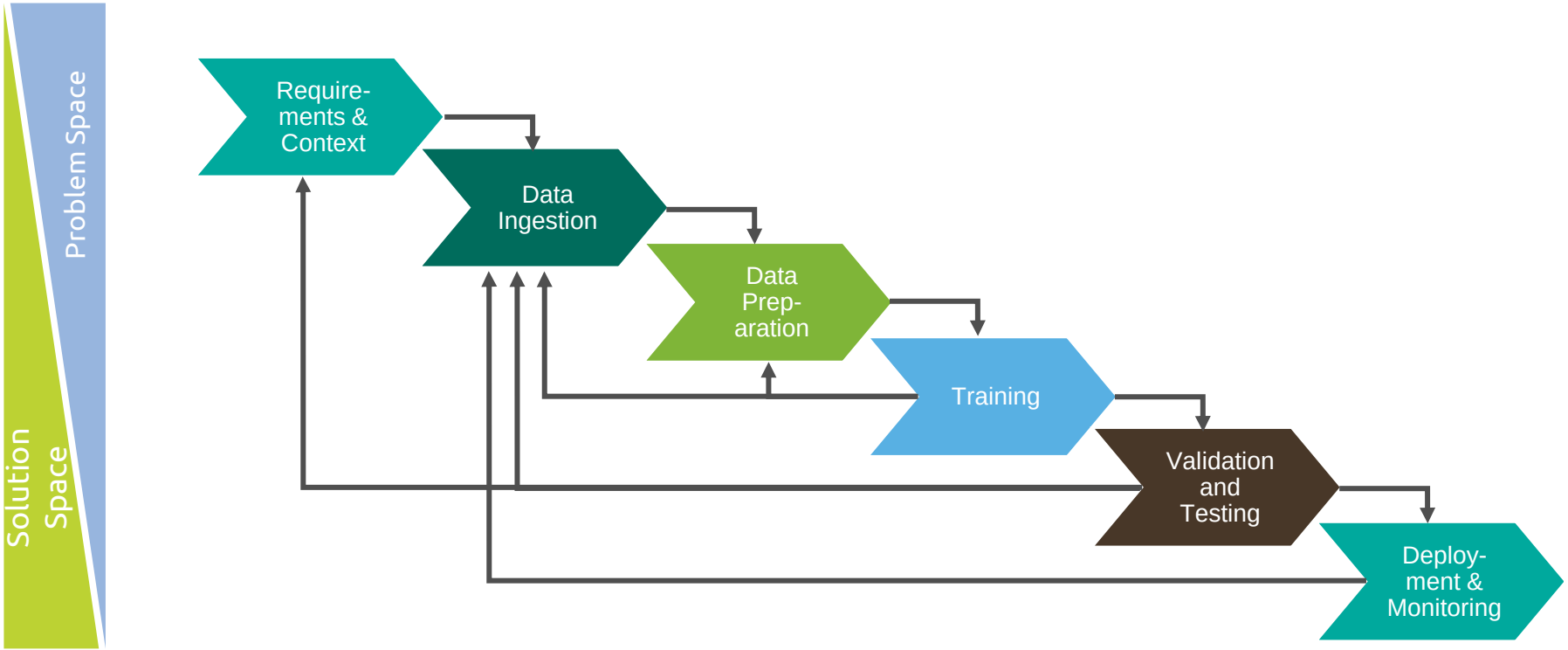
AI Dev. Pipelines and Agile?



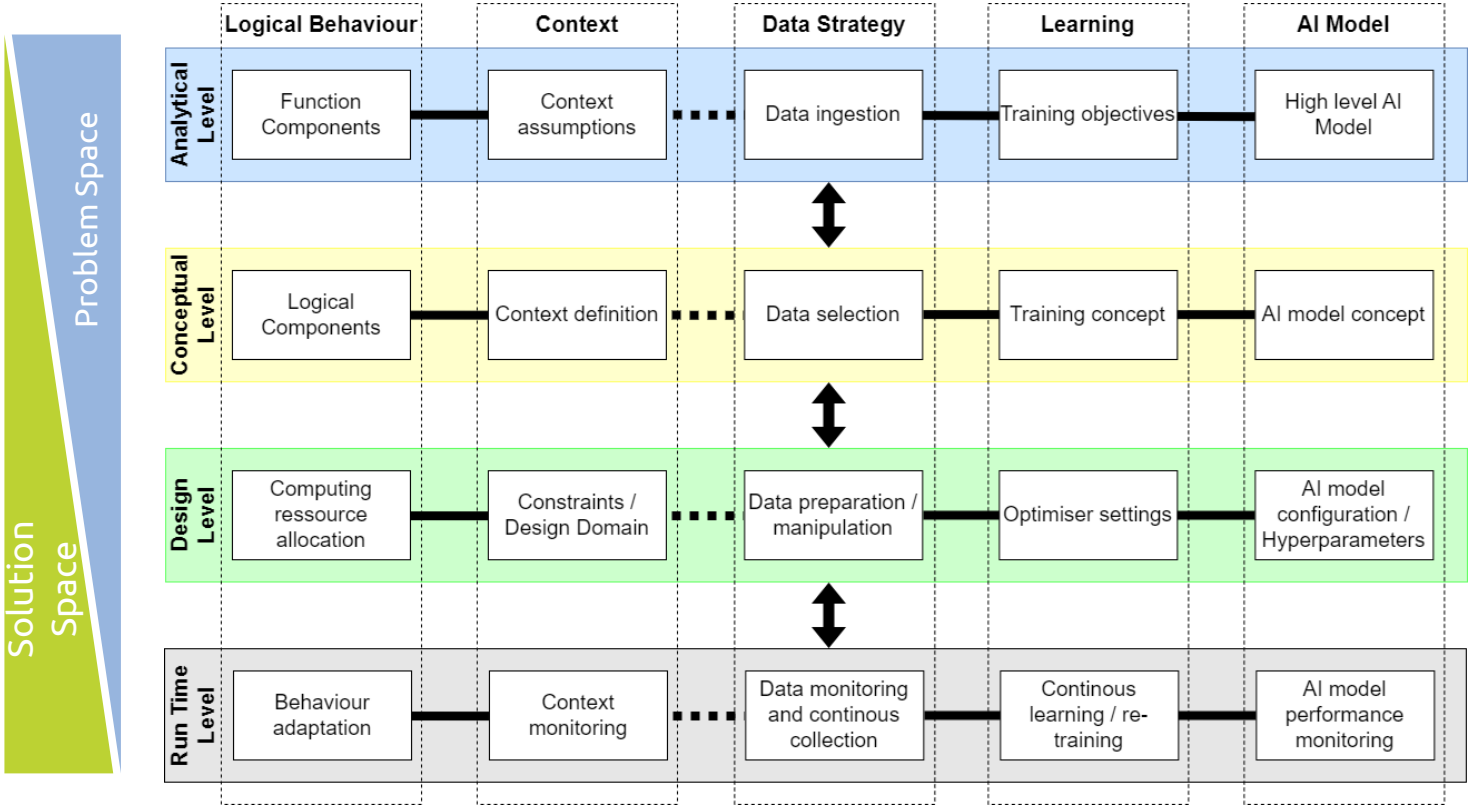
AI Dev. Pipelines and Agile?



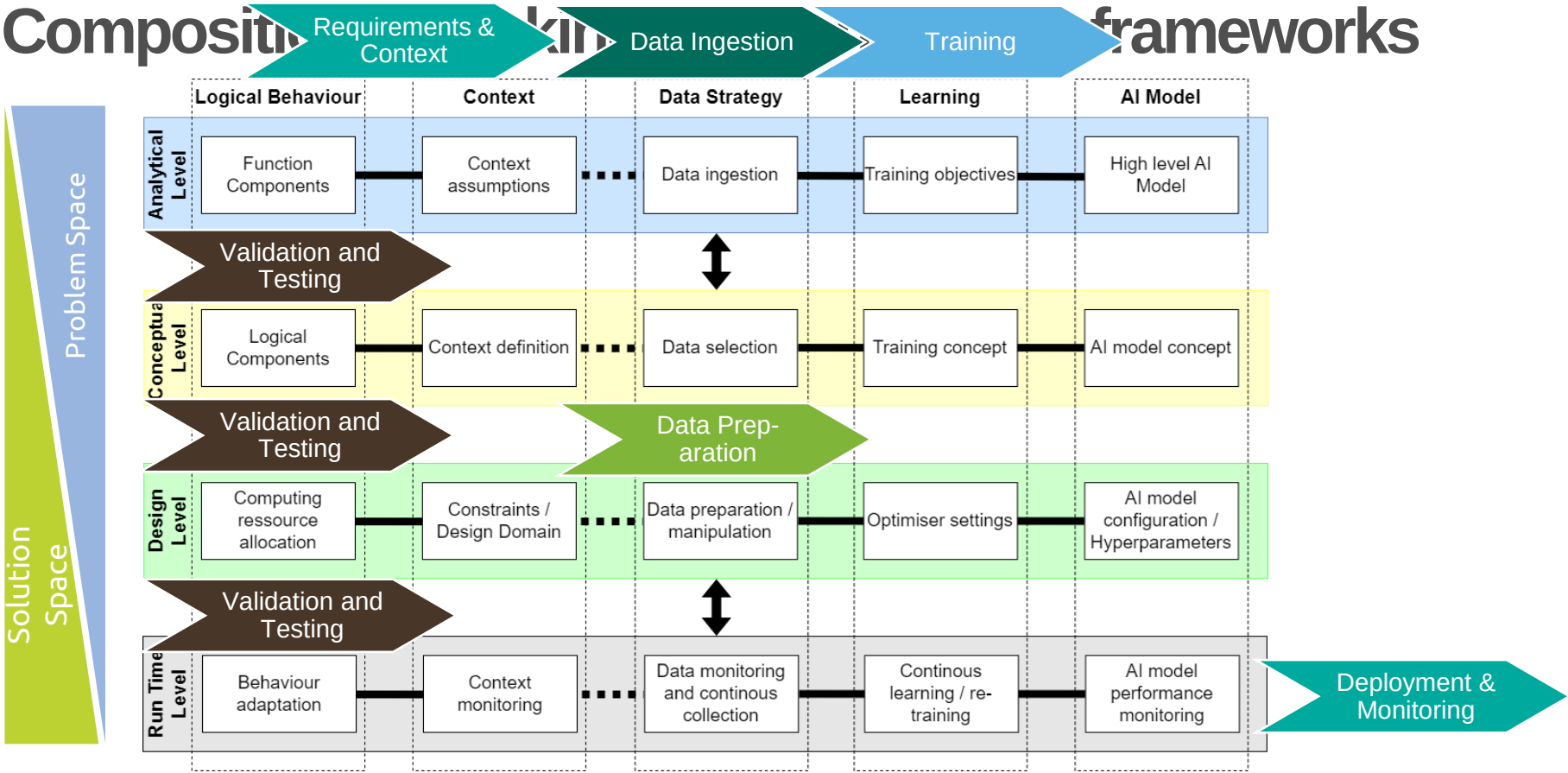
AI Dev. Pipelines and Agile?



Compositional thinking for architecture frameworks



Composit… frameworks



Cluster of concern

Cluster of concern

A cluster of concern is a partially ordered set of architectural views which represent a specific concern of the system at different levels of details.

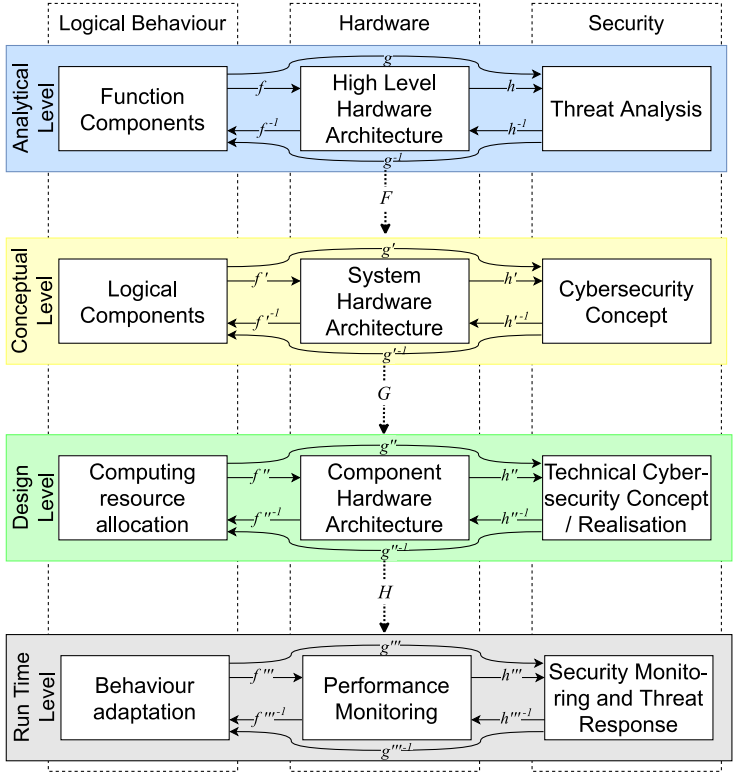
Example. The architectural view "computing resource allocation" contains more details about the final system than the architectural view "logical components". The architectural view "function components" contains the least amount of details. These three architectural views form an ordered set of architectural views considering the concern "logical behaviour". Therefore, they form the cluster of concern "Logical Behaviour".

Level of abstraction

Level of abstraction

The set of architectural views with equivalent level of details about the system-of-interest constitutes a category. A category is a collection of objects which are related to each other in a consistent way (Perrone, 2019). We call the category level of abstraction. *Architectural views on a level of abstraction are related to each other through morphisms. A morphism can exist only between architectural views on the same level of abstraction.*

Level of abstraction - Example



Knowledge creation (e.g. definition of security goals).

Concept design (e.g. introduction of safety mechanisms).

Final design (e.g. assigning functions to secure processor environments.

Monitoring concept definition (e.g. monitoring of secure processing environment).

Consistent architecture

Consistency of architectures

If the product over all architectural views on a level of abstraction is valid,[a] the architectural views consistently describe the system-of-interest.

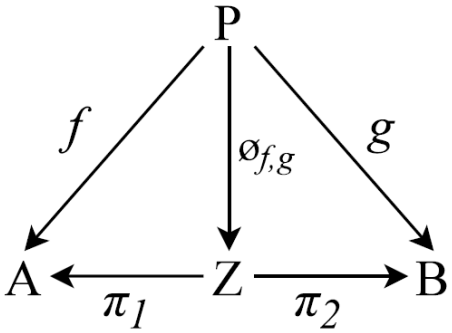
[a] Valid means, informally, that no matter which architectural view one "starts at", it is always possible to "transit" via the correspondence rules to another architectural view, and still see the same system (from a different perspective).

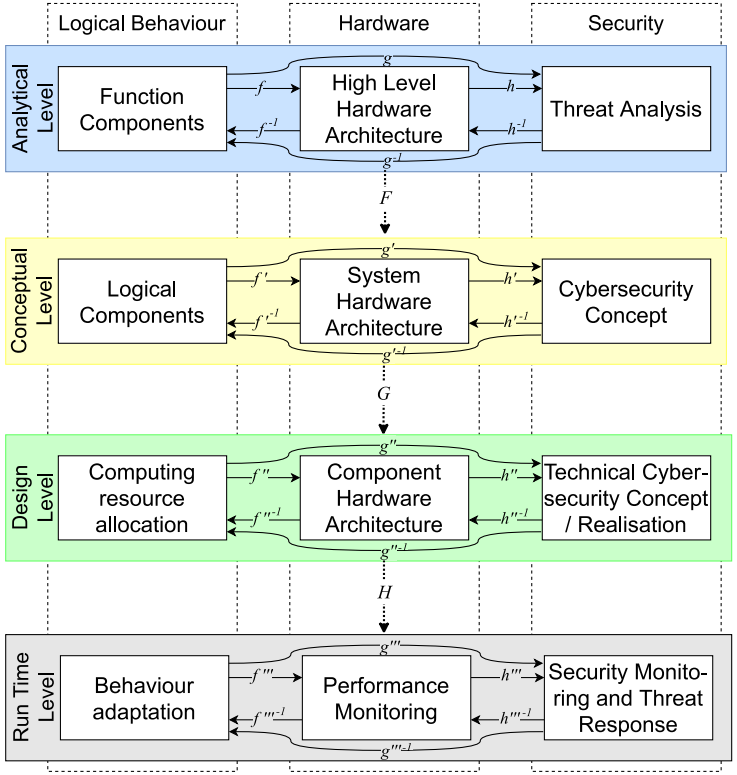
Consistent architecture - Example

Consistency of architectures

If the product over all architectural views on a level of abstraction is valid,[a] the architectural views consistently describe the system-of-interest.

[a] Valid means, informally, that no matter which architectural view one "starts at", it is always possible to "transit" via the correspondence rules to another architectural view, and still see the same system (from a different perspective).





Mapping of relations (and dependencies)

Mapping of relations

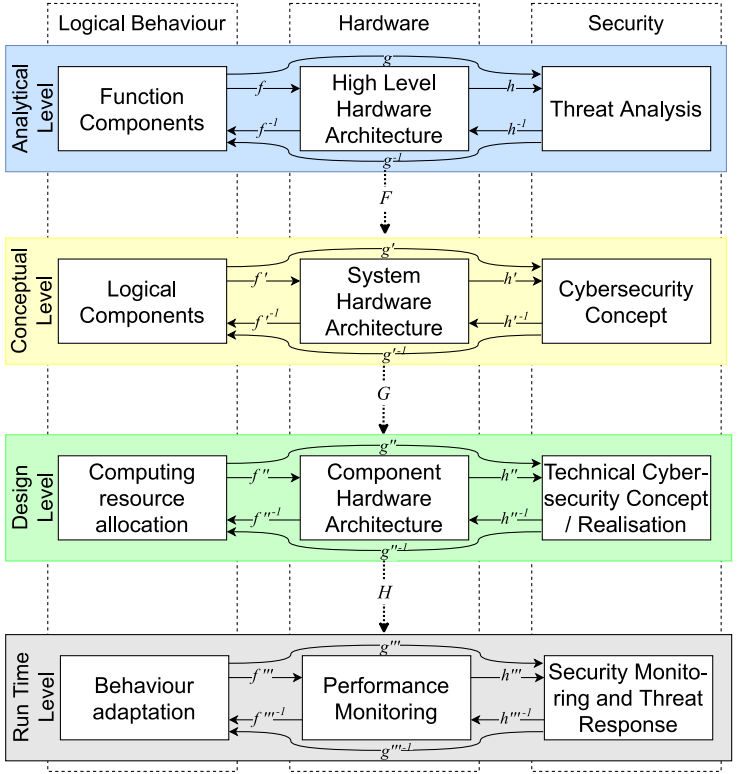
Functors map all views of one level of abstraction to corresponding views of the next lower level of abstraction, and all relations between views to corresponding relations of the next lower level of abstraction.

Mapping of relations - Example

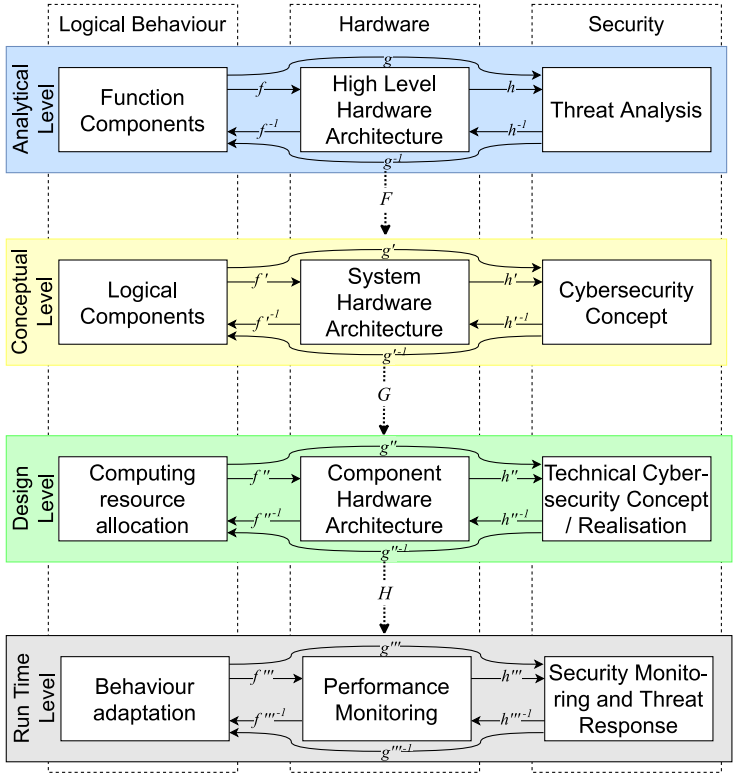
Mapping of relations

Functors map all views of one level of abstraction to corresponding views of the next lower level of abstraction, and all relations between views to corresponding relations of the next lower level of abstraction.

- Do not loose dependencies along the way.



Compositional thinking for architecture frameworks



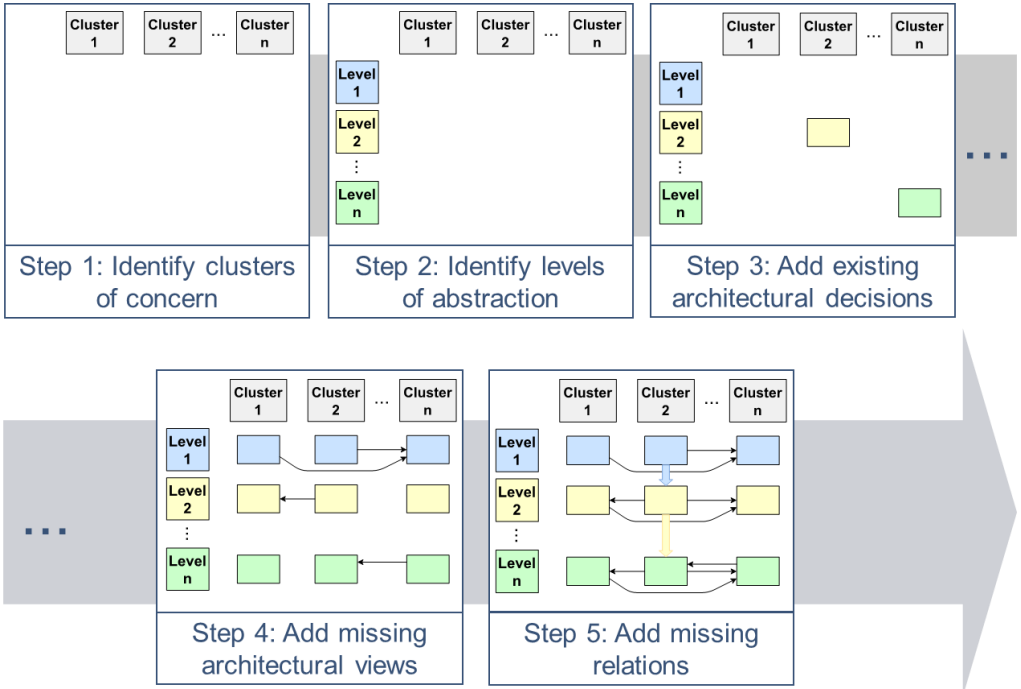
Rule 1: Clusters of concern shall contain architectural views with different levels of details of a certain aspect of the system under development.

Rule 2: Architectural views shall be sorted into levels of abstractions, according to their level of details about the system under development.

Rule 3: By using correspondence rules, it shall be possible to arrive at different architectural views of the system without encountering inconsistencies.

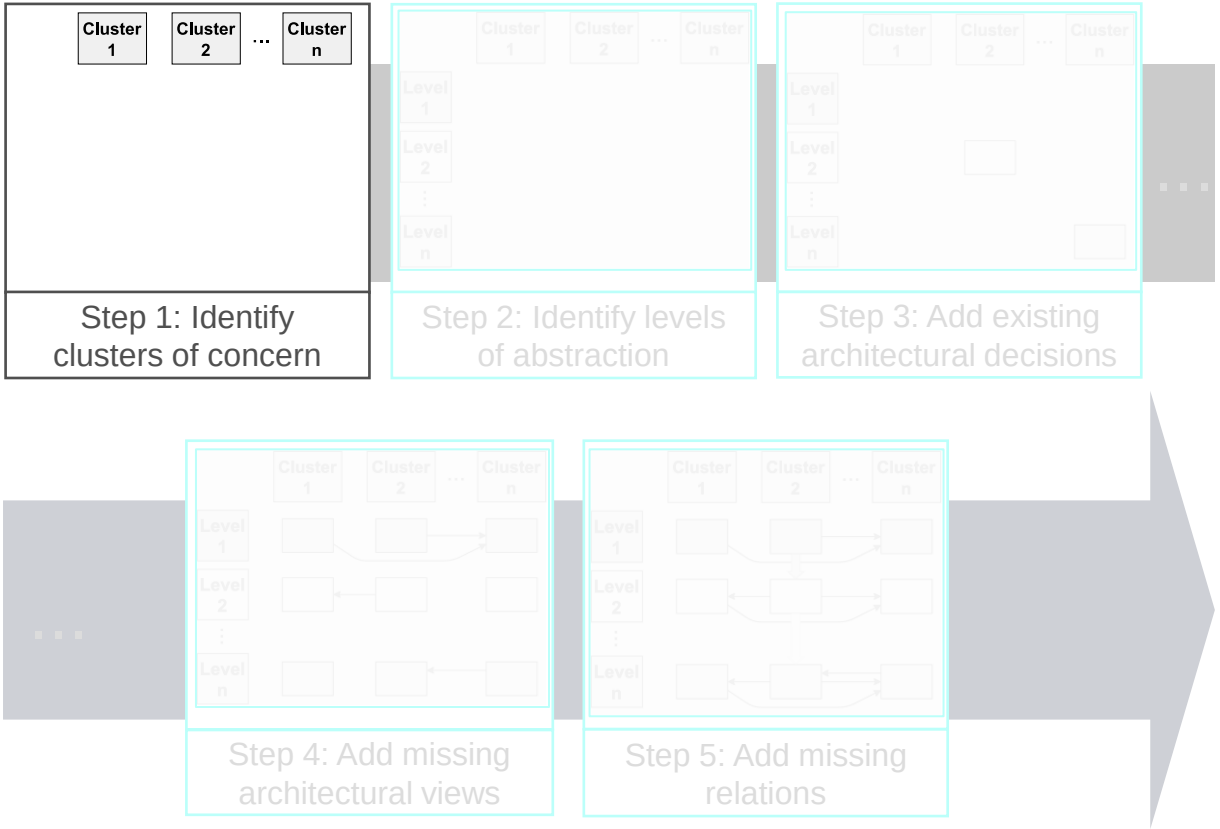
Rule 4: Architectural views, and relations between them, shall be mapped to the next lower level of abstraction.

Iterative and middle-out design of AI systems

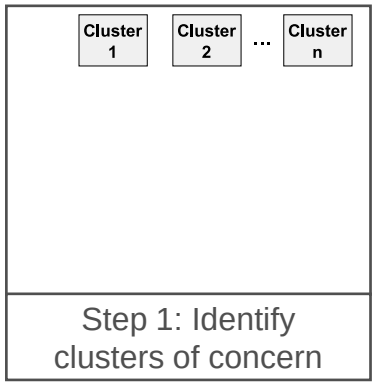


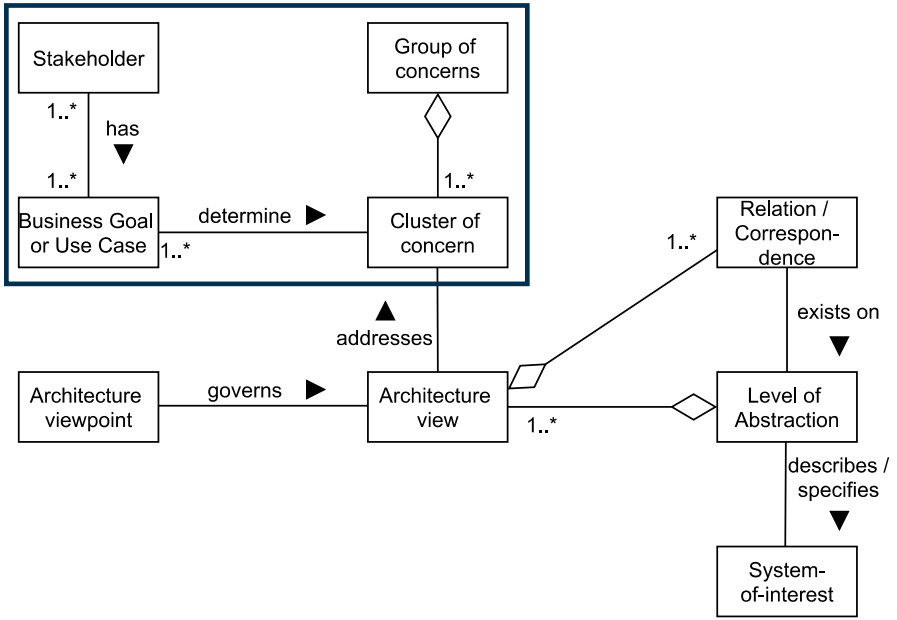
- Step 1: Identify clusters of concern
- Step 2: Identify levels of abstraction
- Step 3: Add existing architectural decisions.
- Step 4: Add missing architectural views.
- Step 5: Add missing relations.
- Step 6: Iterate if needed.

An example: Automatic emergency braking

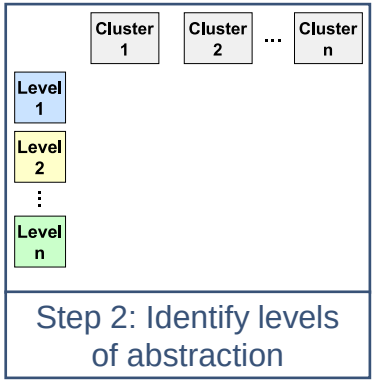


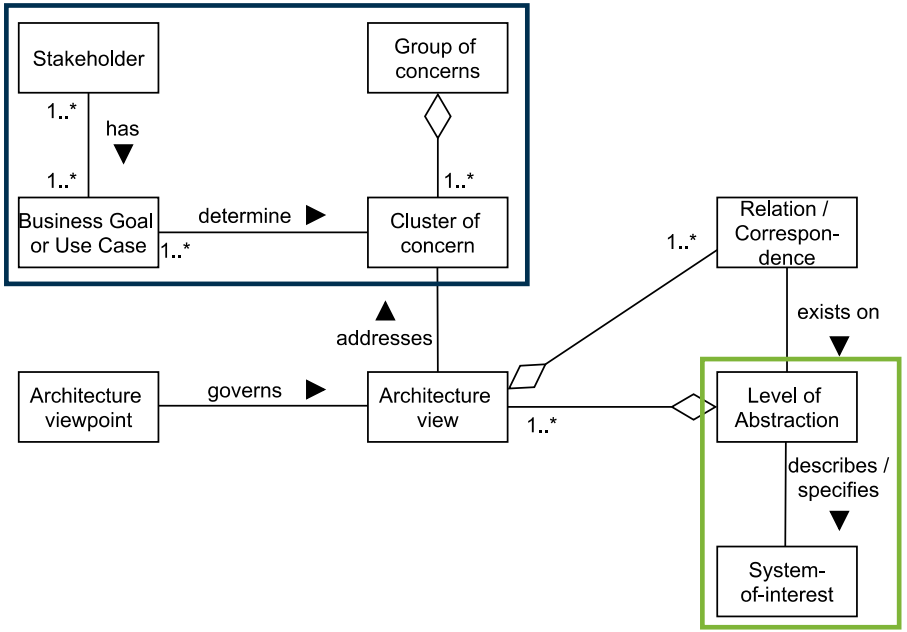
- Step 1: Identify clusters of concern
- Step 2: Identify levels of abstraction
- Step 3: Add existing architectural decisions.
- Step 4: Add missing architectural views.
- Step 5: Add missing relations.
- Step 6: Iterate if needed.

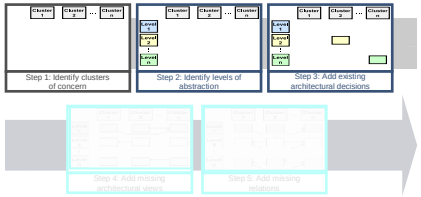

Cluster 1
Cluster 2
...
Cluster n
Step 1: Identify clusters of concern

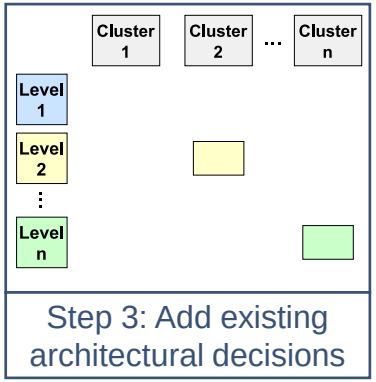

Stakeholder
Group of concerns
1..*
has
1..*
1..*
Business Goal or Use Case
determine
1..*
Cluster of concern
addresses
Relation / Correspon-dence
1..*
exists on
Architecture viewpoint
governs
Architecture view
1..*
Level of Abstraction
describes / specifies
System-of-interest

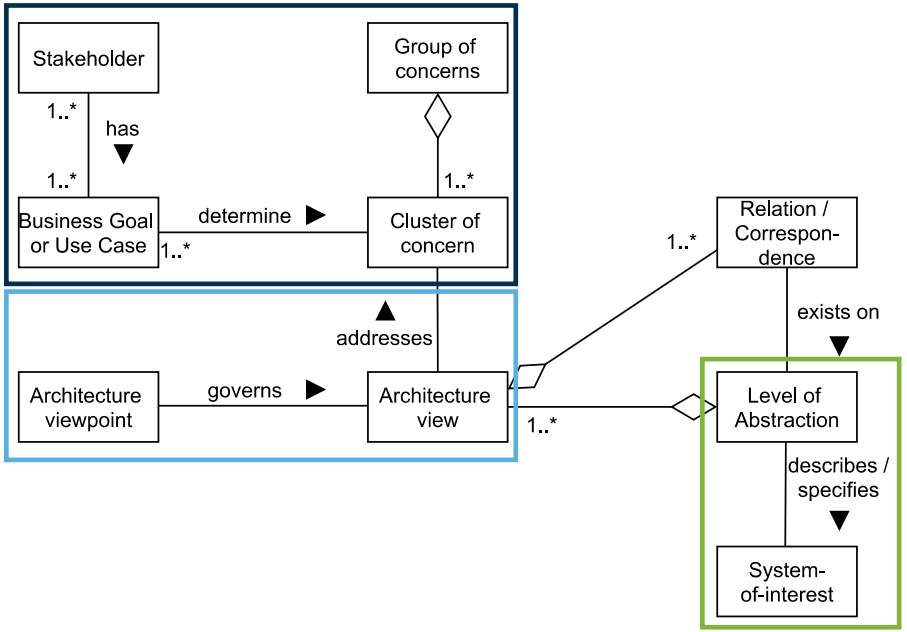

Step 1: Identify clusters of concern
Step 2: Identify levels of abstraction
Step 3: Add existing architectural decisions
Step 4: Add missing architectural views
Step 5: Add missing relations

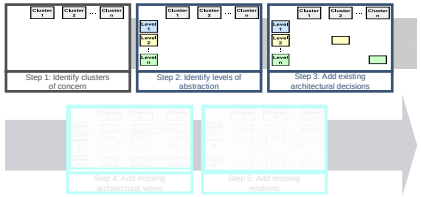

Cluster 1
Cluster 2
...
Cluster n
Level 1
Level 2
Level n
Step 2: Identify levels of abstraction

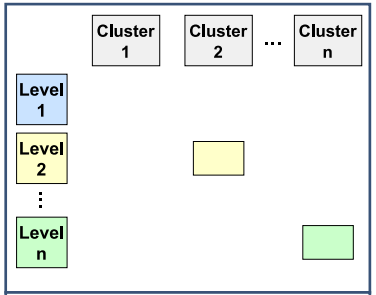

Stakeholder
Group of concerns
1..*
has
1..*
1..*
Business Goal or Use Case
determine
1..*
Cluster of concern
addresses
Architecture viewpoint
governs
Architecture view
1..*
1..*
Relation / Correspon-dence
exists on
Level of Abstraction
describes / specifies
System-of-interest


Step 1: Identify clusters of concern
Step 2: Identify levels of abstraction
Step 3: Add existing architectural decisions
Step 4: Add missing architectural views
Step 5: Add missing relations


Cluster 1
Cluster 2
...
Cluster n
Level 1
Level 2
Level n
Step 3: Add existing architectural decisions

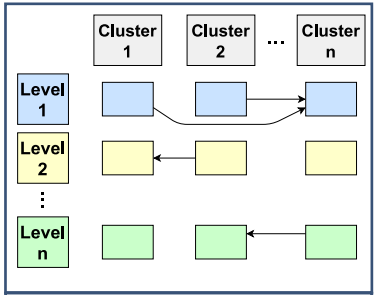

Stakeholder
Group of concerns
1..*
has
1..*
1..*
Business Goal or Use Case
determine
1..*
Cluster of concern
addresses
Architecture viewpoint
governs
Architecture view
1..*
1..*
Relation / Correspon-dence
exists on
Level of Abstraction
describes / specifies
System-of-interest



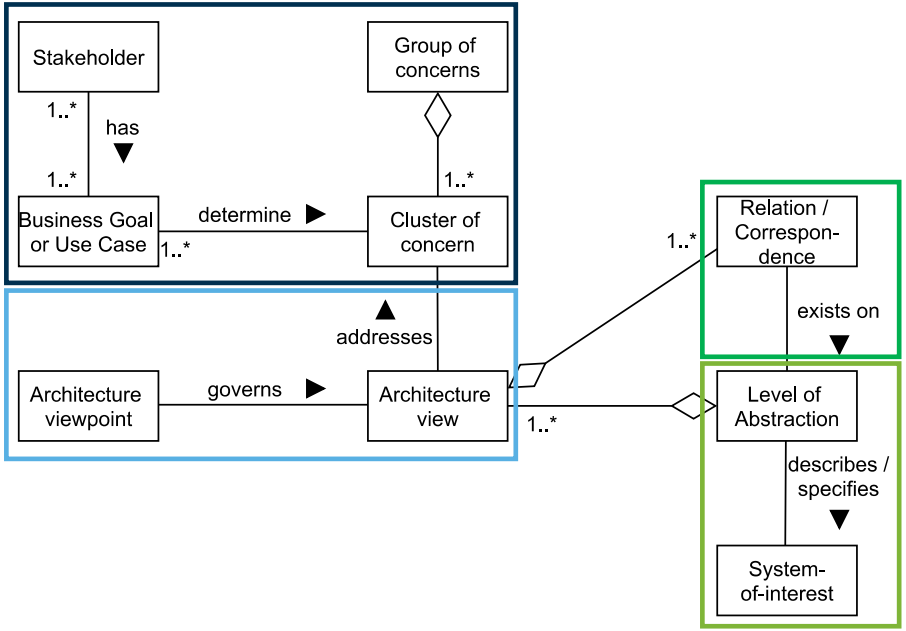


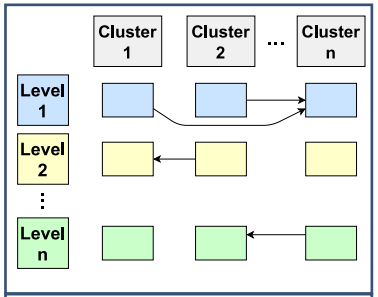
Step 3: Add existing architectural decisions

	Context and Constraints	Data Strategy	Learning	AI Model	Hardware
Analytical Level	**Context Assumption** - Obstacles consist of Pedestrians. - Pedestrians can either be in the lane, or on the road but not in the lane, or the road is empty. - [...]		Classification Categories Obstacle Detection Pedestrian in the lane Pedestrian on the road, but not in the lane Empty Road	Model for Obstacle Detection. Deep Learning Network. Input: Sensor data. Output: 3 classes Timing: real-time	Vehicle Platform: Sensors, Act-uators, Processing and decision unit, Comm; Edge Unit; OEM Cloud Unit
Conceptual Level		**Training data quality attributes** - Resolution at 1920x1080 - Object is in focus - Pedestrians standing, walking, running, and in wheel chair. **Feature attributes** - Variation of pedestrian clothes - Variation of pedestrian look - Background variations			
Design Level	**Design Domain** - System can only be used on motorways or highways. - Headlights are used at night time. - Camera mountaing can vary by +/- 5 degrees. - [...]				Vehicle Platform: Camera, Ethernet Bridge, ML Accelerator, Ethernet Bridge, Gas pedal, Warning lights, CAN FD gateway, Jetson NX, USB 3.1, 5G modem, CAN2.0B gateway, Brake system, Brake ECU; Edge Unit: 5G Unit > 1 TFLOPS > 1 GB storage; OEM Cloud Unit: > 10 TFLOPS > 100 GB storage
Run Time Level	**Context Monitoring** - System is out of context if vehicle speed exceeds 130 km/h. - [...]				



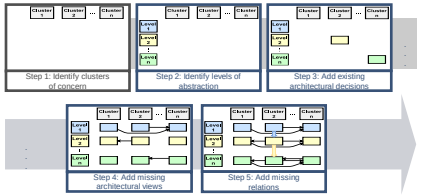
Step 4: Add missing architectural views

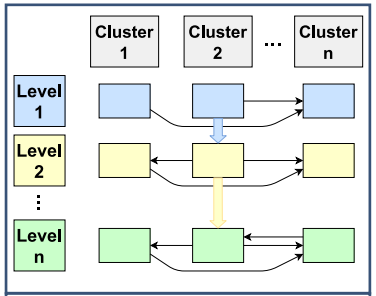




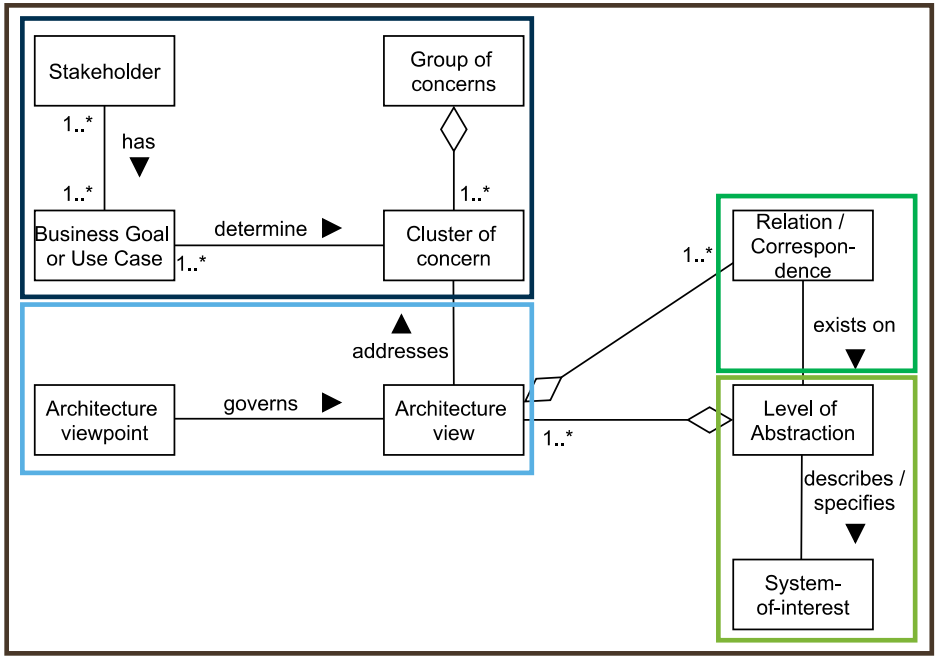
Step 4: Add missing architectural views

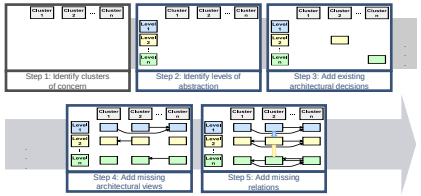
	Context and Constraints	Data Strategy	Learning	AI Model	Hardware
Analytical Level	**Context Assumption** - Obstacles consist of Pedestrians. - Pedestrians can either be in the lane, or on the road but not in the lane, or the road is empty. - [...]	Data Server at OEM Data labels Data ingestion	Classification Categories Obstacle Detection Pedestrian in the lane Pedestrian on the road, but not in the lane Empty Road	Model for Obstacle Detection. Deep Learning Network. Input: Sensor data. Output: 3 classes Timing: real-time	Vehicle Platform Sensors Act-uators Processing and decision unit Comm Edge Unit OEM Cloud Unit
Conceptual Level	**Context Definition** - Pedestrians are between 1.00 metre and 2.10 metre. - Pedestrians do not move faster than 20 km/h. - Pedestrians can also be in a wheel chair. - [...]	**Training data quality attributes** - Resolution at 1920x1080 - Object is in focus - Pedestrians standing, walking, running, and in wheel chair. **Feature attributes** - Variation of pedestrian clothes - Variation of pedestrian look - Background variations	Pedestrian in the lane Pedestrian on the road, but not in the lane Empty Road 2500 objects 2500 objects 2500 objects Training and Testing Data	Input: Visual Data (1920 x 1080 x 3) Feature extraction (Convolution & Max-pooling) Flattening Classification (Fully-Connected Layers) Output (3 x 1 x 1)	Vehicle Platform Camera Vision ECU Gas pedal Warning lights Processing and decision unit 5G modem Brake system Brake ECU Edge Unit OEM Cloud Unit
Design Level	**Design Domain** - System can only be used on motorways or highways. - Headlights are used at night time. - Camera mountaing can vary by +/- 5 degrees. - [...]	**Data manipulations** - Allow for up to 10% of pixels to be randomly disturbed. - Tilt imagine at random with a mean of 5 degrees.	**Optimiser Settings** - Loss function: Categorical-Crossentropy - Optimiser: RMSprop - Decay Rate: 6exp(-8) - Clipvalue: 1.0 **Learning Settings** - Epochs: 2000 - Batch Size: 256 - Dropout after Dense Layers: 40% - [...]	256 Filters, Kernel Size (3x3), Activation Function "ReLU" Pool Size 2x2, Strides 3x3 384 Filters, Kernel Size (3x3), Activation Function "ReLU" Pool Size 2x2, Strides 3x3 384 Filters, Kernel Size (3x3), Activation Function "ReLU" 4096 Neurons, Activation Function "tanh". 4096 Neurons, Activation Function "tanh" 3 Neurons, Activation Function "softmax"	Vehicle Platform Camera Ethernet Bridge ML Accelerator Ethernet Bridge Gas pedal Warning lights CAN FD gateway Jetson NX USB 3.1 5G modem CAN2.0B gateway Brake system Brake ECU Edge Unit 5G Unit > 1 TFLOPS > 1 GB storage OEM Cloud Unit > 10 TFLOPS > 100 GB storage
Run Time Level	**Context Monitoring** - System is out of context if vehicle speed exceeds 130 km/h. - [...]	**Data testing and validation** Input data Check resolution and brightness Check percentage of broken pixels	**Learning validation** Random Generator Capture Image and Classification Result Data Server at OEM	**Test and validation strategy** Feature map activity monitoring Uncertainty Monitoring Feature extraction Classification Output Object recognition network	**Test and validation strategy** Camera Hardware Watchdog Jetson NX Hardware Watchdog Power monitoring ML Accelerator Hardware Watchdog 5G modem Latency check OEM Cloud Unit > 10 TFLOPS > 100 GB storage Service monitoring

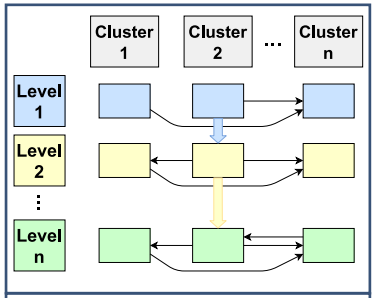




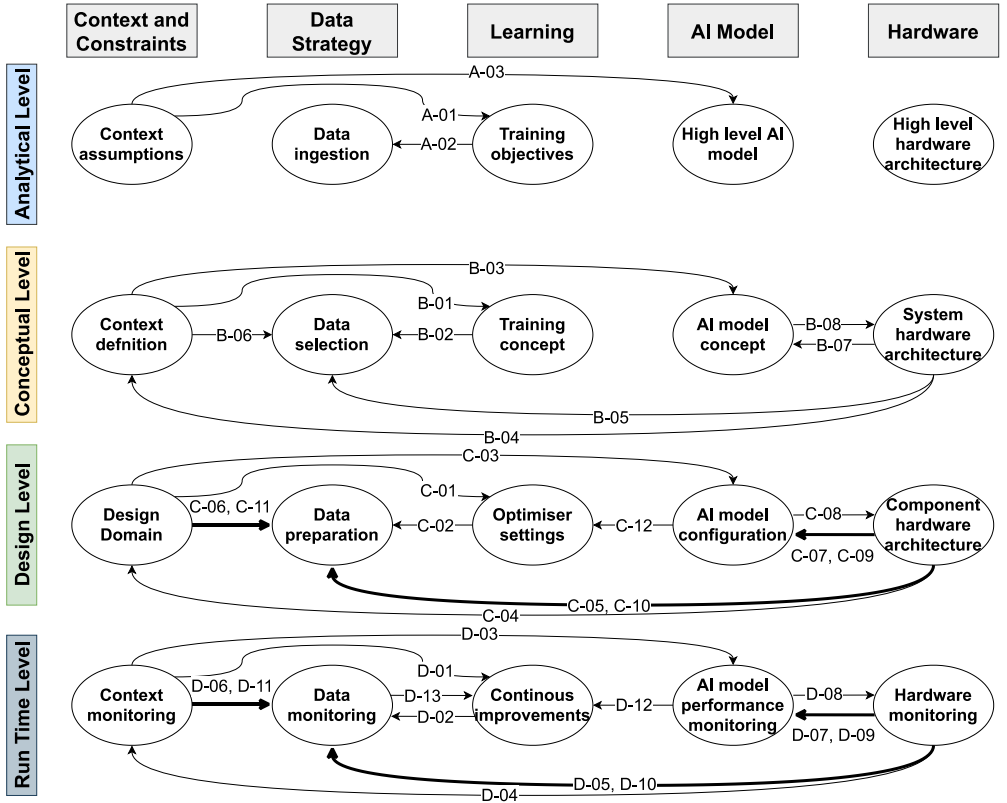
Step 5: Add missing relations



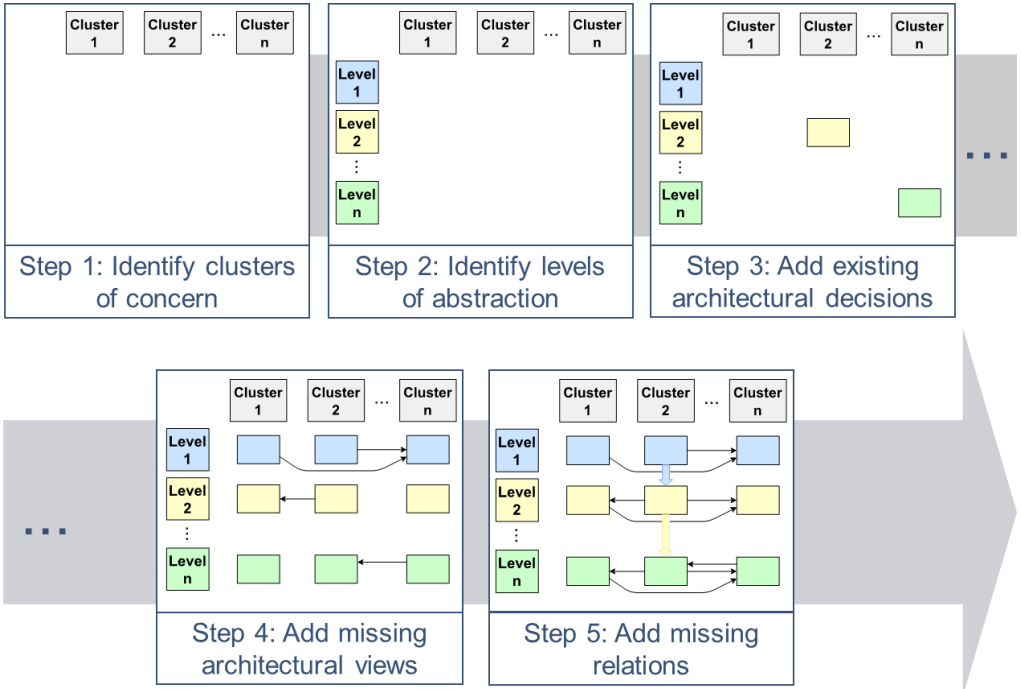




Step 5: Add missing relations



Iterative and middle-out design of AI systems



- Step 1: Identify clusters of concern
- Step 2: Identify levels of abstraction
- Step 3: Add existing architectural decisions.
- Step 4: Add missing architectural views.
- Step 5: Add missing relations.
- Step 6: Iterate if needed.

Traceability

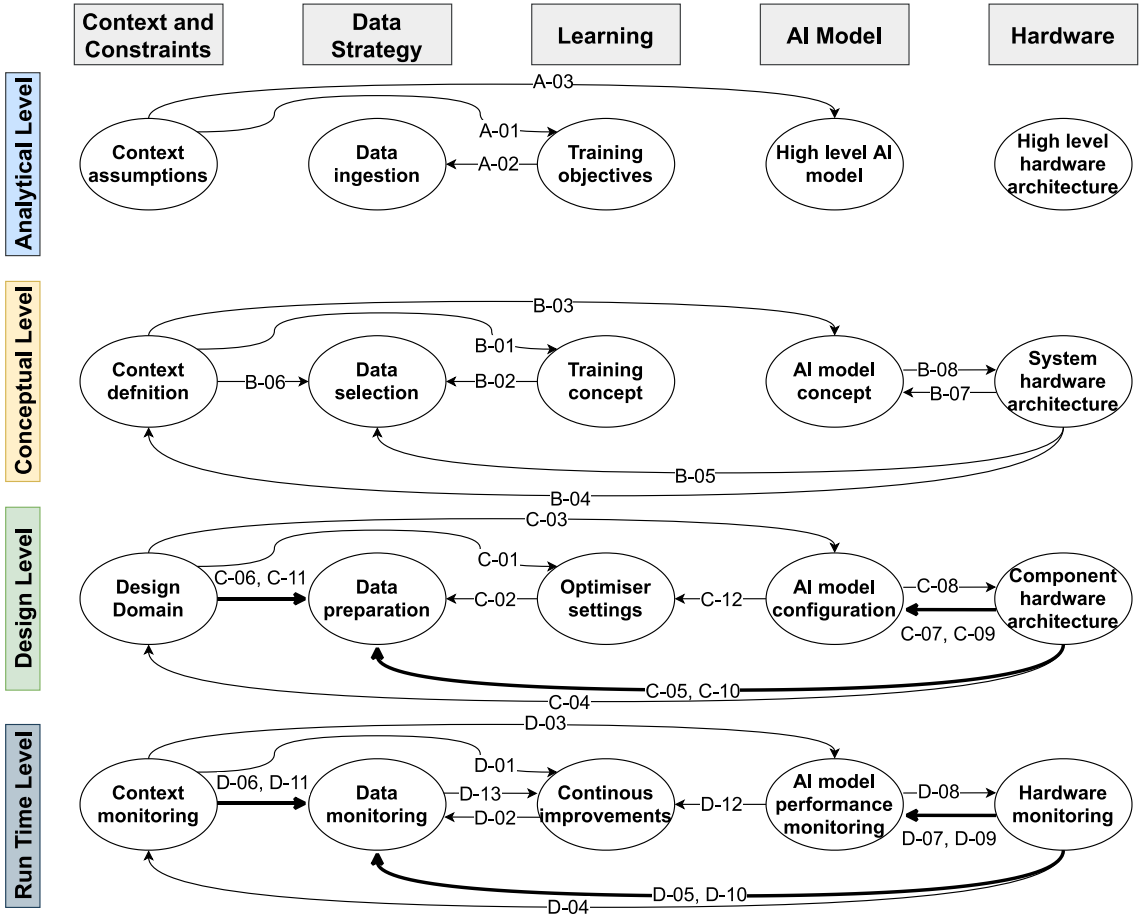
EUROPEAN COMMISSION

Proposal for a

REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL

LAYING DOWN HARMONISED RULES ON ARTIFICIAL INTELLIGENCE (ARTIFICIAL INTELLIGENCE ACT) AND AMENDING CERTAIN UNION LEGISLATIVE ACTS

{SEC(2021) 167 final} - {SWD(2021) 84 final} - {SWD(2021) 85 final}



Very efficient deep learning in the IoT

Requirements

Modelling & Verification

Safety & Robustness

Applications

Smart Home

Industrial IoT

Automotive AI

Security & Safety

Middleware

Toolchain

embedl

Emulation

RENODE

Benchmarking & Deployment

Kenning

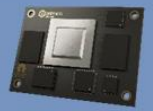
Microserver & Accelerators

Xilinx
Kria

Coral SoM

COM-HPC
Xilinx Zynq
UltraScale+

Jetson AGX
NVIDIA Xavier



RPi CM4
ARVSOM

SMARC
Xilinx Zynq
UltraScale+

Hardware Platforms

Embedded/
Far Edge

u.RECS

Near Edge

t.RECS

Cloud

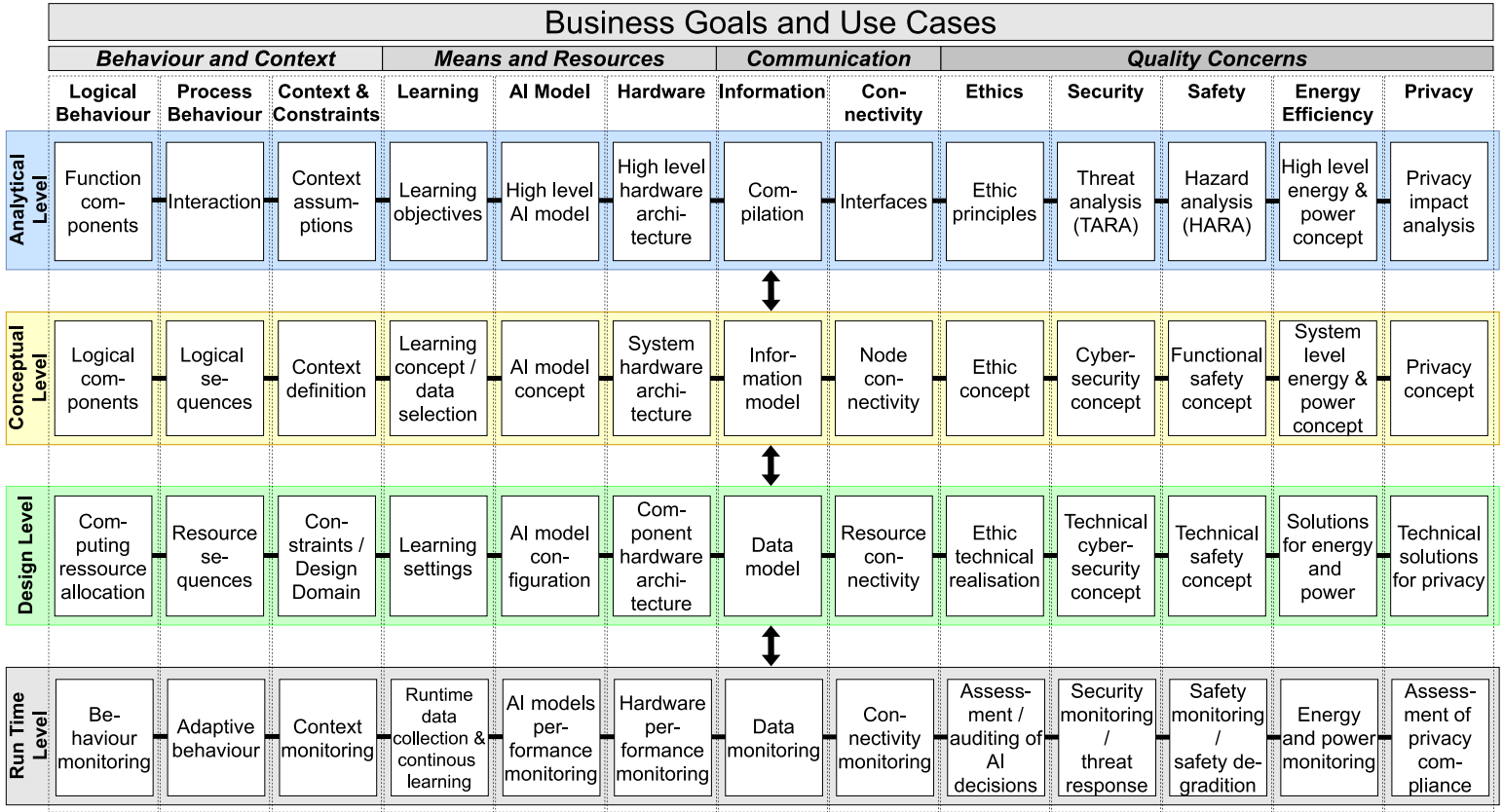
RECS|Box

Trusted Execution & Communication

Monitoring

RISC-V extensions

A compositional architecture framework for VEDLIoT



Summary

- The architectural framework helps connecting different aspects of a (distributed) AI system together.
- It is based on mathematical ideas from category theory and compositional thinking.
- It allows for “middle-out” development, i.e., existing design decisions are explicitly considered.
- It allows to keep an overview over the necessary quality aspects, such as safety, security, fairness, or privacy aspects of the systems.
- The framework enforces a runtime concept for the system.
- The traceability of design decisions allows for compliance with upcoming AI regulations.

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY