Lecture 1

Design of Software Architectures

Dr. Georgiana Caltais, Dr. Vadim Zaytsev

UNIVERSITY OF TWENTE.

Design of Software Architectures

- Goals:
  - understand what software architecture is
  - understand when software architecture is needed
  - understand why software architecture is needed
  - have a conceptual framework for software architects
  - be able to explain the above to others
  - (know the rest)

the rest in https://osiris.utwente.nl/student/OnderwijsCatalogusZoekCursus.do

Team

Fernando Castor
f.castor@utwente.nl

Georgiana Caltais
g.g.c.caltais@utwente.nl

Vadim Zaytsev
v.zaytsev@utwente.nl

Lecture 1: Overview

- This course
- Software Architecture in a Nutshell
- Architecture Description
  - Scope. Environment. Software. Statements.

In DoSA, you will:

- Simulate architecting a real system
- Apply knowledge from sessions
- Realise it is not exact science
- Focus on understanding concepts and principles
- Establish consistency throughout the document
- Provide rationale/justification of choices
- Apply the knowledge that you already have
- Evaluate architecture of another group
- Suggest improvements

Cases

- Work in teams!
- 6–7 people
- Deadline TODAY
  - Team leader sends
    - preferred case
    - student names
- First come first served

Boldly

Boldly is a comprehensive software system designed to revolutionise urban transportation by leveraging advanced technologies for seamless traffic flow, optimised route planning, and enhanced commuter experiences. This system integrates various components, including real-time traffic monitoring, data analytics, user interfaces, and communication networks, to create a holistic solution for modern urban mobility challenges.

Users can access **Boldly** through web and mobile applications. The system provides real-time traffic updates, suggests optimal routes based on current conditions, and dynamically adjusts directions to avoid traffic jams. **Boldly** can predict traffic patterns, peak hours, and potential bottlenecks. This enables proactive traffic management and helps users plan their journeys more efficiently. Users can ask **Boldly** about travel options that combine buses, trains, and shared mobility services. **Boldly** supports effective communication and user-friendly interfaces that play a crucial role in increasing public engagement and user adoption.

Boldly is integrated with existing transportation infrastructure, third-party data sources, and various communication protocols to ensure uninterrupted service even during peak loads or technical failures. In case of accidents or emergencies, **Boldly** triggers automated alerts to emergency services, reroutes traffic, and provides real-time updates to commuters and relevant authorities. The system controls traffic signals dynamically, based on real-time traffic flow, reducing congestion and improving overall traffic management.

A comprehensive dashboard provides city officials with insights into traffic patterns, congestion hotspots, and transportation trends. This data informs urban planning decisions and infrastructure improvements. The urban mobility landscape is continuously evolving with the introduction of new transportation technologies (e.g., autonomous vehicles) and changing urban dynamics. **Boldly** must remain adaptable and flexible to accommodate these changes.

Processing large volumes of real-time data from diverse sources requires robust and efficient data processing and analytics infrastructure. **Boldly** implements sophisticated dynamic and adaptive traffic signal control algorithms that respond to changing traffic conditions in real time, thus ensuring low latency and high throughput while maintaining accurate insights. This grants **Boldly** a definite competitive advantage over their peers in the industry.

WiseWorld

WiseWorld is a comprehensive software system designed to optimise and streamline various aspects of urban living through advanced technological solutions. This platform acts as a central hub that connects and manages a wide range of interconnected subsystems, all working in harmony to enhance the quality of life for city residents and improve overall city operation.

WiseWorld integrates real-time data from sensors embedded in various city infrastructure elements such as roads, bridges, public transportation systems and utilities. It provides predictive maintenance insights, enabling timely repairs and minimising disruptions. It also offers intelligent traffic flow analysis, congestion detection, and dynamic traffic signal optimization to reduce commute times and enhance road safety. Emergency services can access real-time information about incidents, accidents, and potential hazards. **WiseWorld** aids in dispatching resources effectively and communicates critical updates to residents in times of emergencies.

The platform enables efficient management of energy consumption and distribution across the city. It integrates with smart grids, monitors energy usage patterns, and implements demand-response strategies to ensure sustainable resource utilisation. By collecting data from air and water quality sensors, **WiseWorld** provides insights into environmental health. It supports eco-friendly initiatives and aids in achieving sustainability goals. City planners can utilise **WiseWorld** to simulate various urban development scenarios, assess their impact on traffic, environment, and resources, and make informed decisions for sustainable city growth.

The platform aggregates and analyses massive amounts of data from different sources, providing actionable insights through intuitive visualisations and reports to city administrators and decision-makers. **WiseWorld** offers a mobile app for residents to access city services, report issues, and participate in community activities. It also facilitates e-governance, enabling citizens to engage with local authorities directly. **WiseWorld** comes equipped with intuitive user interfaces and mobile applications that provide citizens with easy access to services and promote engagement.

WiseWorld is an evolving solution that needs regular updates, maintenance, and potentially even hardware upgrades, while ensuring minimal downtime and disruptions to city services. The system implements sophisticated algorithms exploiting data from various sources, such as IoT sensors, municipal databases, social media, and external APIs, that ensure compatibility, accuracy, and real-time synchronisation across disparate data streams. This provides **WiseWorld** with a clear and significant competitive edge over their industry competitors.

Boldly

Boldly is a comprehensive software system designed to revolutionise urban transportation by leveraging advanced technologies for seamless traffic flow, optimised route planning, and enhanced commuter experiences. This system integrates various components, including real-time traffic monitoring, data analytics, user interfaces, and communication networks, to create a holistic solution for modern urban mobility challenges.

Users can access **Boldly** through web and mobile applications. The system provides real-time traffic updates, suggests optimal routes based on current conditions, and dynamically adjusts directions to avoid traffic jams. **Boldly** can predict traffic patterns, peak hours, and potential bottlenecks. This enables proactive traffic management and helps users plan their journeys more efficiently. Users can ask **Boldly** about travel options that combine buses, trains, and shared mobility services. **Boldly** supports effective communication and user-friendly interfaces that play a crucial role in increasing public engagement and user adoption.

Boldly is integrated with existing transportation infrastructure, third-party data sources, and various communication protocols to ensure uninterrupted service even during peak loads or technical failures. In case of accidents or emergencies, **Boldly** triggers automated alerts to emergency services, reroutes traffic, and provides real-time updates to commuters and relevant authorities. The system controls traffic signals dynamically, based on real-time traffic flow, reducing congestion and improving overall traffic management.

A comprehensive dashboard provides city officials with insights into traffic patterns, congestion hotspots, and transportation trends. This data informs urban planning decisions and infrastructure improvements. The urban mobility landscape is continuously evolving with the introduction of new transportation technologies (e.g., autonomous vehicles) and changing urban dynamics. **Boldly** must remain adaptable and flexible to accommodate these changes.

Processing large volumes of real-time data from diverse sources requires robust and efficient data processing and analytics infrastructure. **Boldly** implements sophisticated dynamic and adaptive traffic signal control algorithms that respond to changing traffic conditions in real time, thus ensuring low latency and high throughput while maintaining accurate insights. This grants **Boldly** a definite competitive advantage over their peers in the industry.

WiseWorld

WiseWorld is a comprehensive software system designed to optimise and streamline various aspects of urban living through advanced technological solutions. This platform acts as a central hub that connects and manages a wide range of interconnected subsystems, all working in harmony to enhance the quality of life for city residents and improve overall city operation.

WiseWorld integrates real-time data from sensors embedded in various city infrastructure elements such as roads, bridges, public transportation systems and utilities. It provides predictive maintenance insights, enabling timely repairs and minimising disruptions. It also offers intelligent traffic flow analysis, congestion detection, and dynamic traffic signal optimization to reduce commute times and enhance road safety. Emergency services can access real-time information about incidents, accidents, and potential hazards. **WiseWorld** aids in dispatching resources effectively and communicates critical updates to residents in times of emergencies.

The platform enables efficient management of energy consumption and distribution across the city. It integrates with smart grids, monitors energy usage patterns, and implements demand-response strategies to ensure sustainable resource utilisation. By collecting data from air and water quality sensors, **WiseWorld** provides insights into environmental health. It supports eco-friendly initiatives and aids in achieving sustainability goals. City planners can utilise **WiseWorld** to simulate various urban development scenarios, assess their impact on traffic, environment, and resources, and make informed decisions for sustainable city growth.

The platform aggregates and analyses massive amounts of data from different sources, providing actionable insights through intuitive visualisations and reports to city administrators and decision-makers. **WiseWorld** offers a mobile app for residents to access city services, report issues, and participate in community activities. It also facilitates e-governance, enabling citizens to engage with local authorities directly. **WiseWorld** comes equipped with intuitive user interfaces and mobile applications that provide citizens with easy access to services and promote engagement.

WiseWorld is an evolving solution that needs regular updates, maintenance, and potentially even hardware upgrades, while ensuring minimal downtime and disruptions to city services. The system implements sophisticated algorithms exploiting data from various sources, such as IoT sensors, municipal databases, social media, and external APIs, that ensure compatibility, accuracy, and real-time synchronisation across disparate data streams. This provides **WiseWorld** with a clear and significant competitive edge over their industry competitors.

Strategy

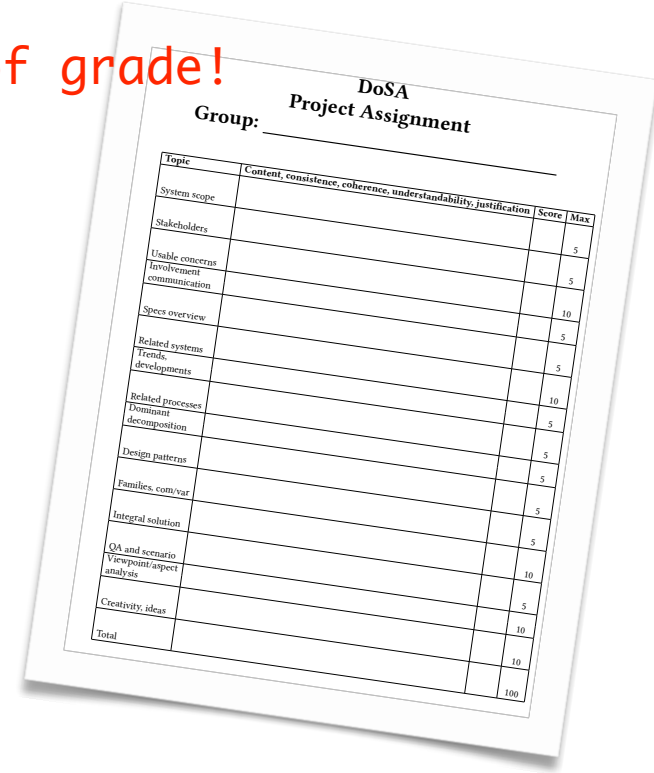
- Choose a case, form a group
- Each group writes an architecture description doc
- Lectures end in homework
  - add a chapter to the architecture description
  - add a cross-cutting concern
  - refine an existing piece of the architecture
- Can use us as a project coach and/or stakeholder

https://canvas.utwente.nl/courses/12664

Group Assignment: SA Description
Suggested Structure

- Introduction
- System scope
- Environment
  - Stakeholders and their concerns
  - Related systems
  - Related processes, organisations, stakeholders
  - Main QA scenarios and use cases
  - Trends and developments
- Design
  - Dominant decomposition/style
  - Design decisions: Justification of choices related to environment
  - View 1..N
    - Per view: Used design patterns (show examples; if you cannot, generalise)
  - Consistency: relation between views, known gaps/inconsistencies.
- Appendix
  - Explanation of viewpoints, references to standards

50% of grade!



DoSA
Project Assignment

Group: _____

Topic	Content, consistence, coherence, understandability, justification	Score	Max
System scope			5
Stakeholders			5
Usable concerns			10
Involvement communication			5
Specs overview			5
Related systems			10
Trends, developments			5
Related processes			5
Dominant decomposition			5
Design patterns			5
Families, com/var			5
Integral solution			10
QA and scenario			5
Viewpoint/aspect analysis			10
Creativity, ideas			10
Total			100

cf. ISO 42010!

Tips for the Group Assignment

- Start with the case description
  - be explicit about changes/additions
- Distribute responsibilities
  - document it in an appendix
- Make sure to apply each lecture
- Don’t get stuck
  - make assumptions
  - ask your coach/stakeholder

Individual Assignment

50% of grade!

- Evaluate the architecture of the counterpart team
- Tasks like
  - Select 2 questions that should have a better answer
  - Select 1 question that has a satisfying answer
  - Apply one QA scenario from your architecture
- Checks if you can
  - handle architecture documents
  - suggest improvements constructively

Deadlines

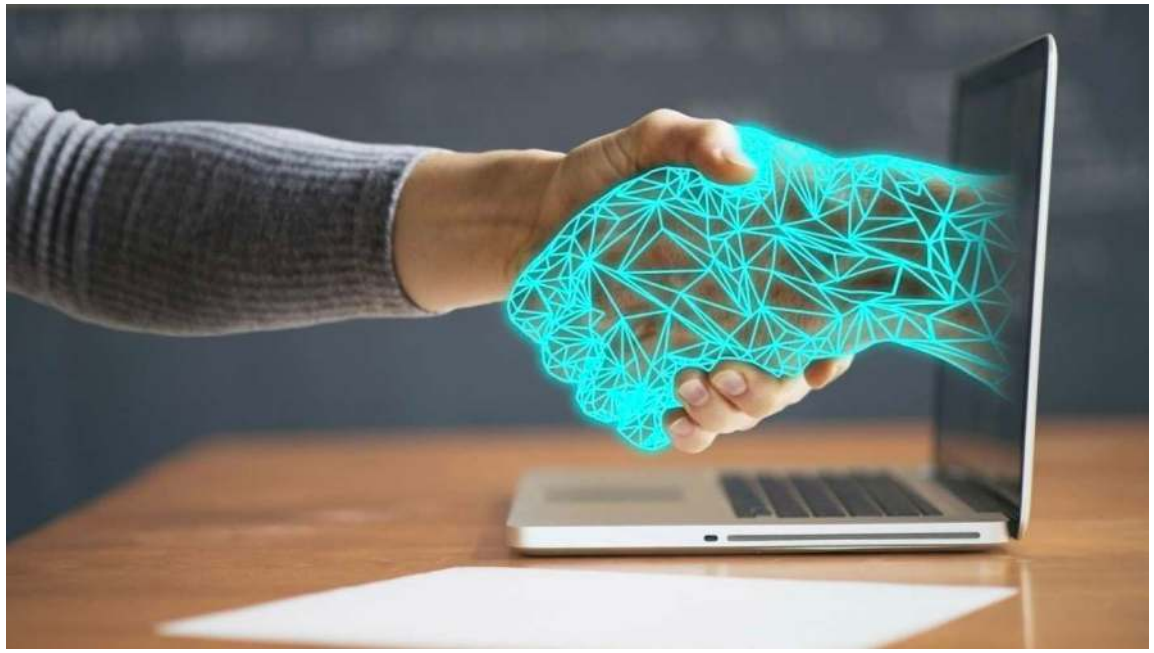
- Intermediate feedback
  - 18 September 18:00
  - 9 October 13:00
- Final document
  - 27 October 17:00
- Individual
  - 10 November 17:00

Week number	36	37	38	39	40	41	42	43	44	45
Week type	L	IM	L		IM	L	IM			
Quartile week	01-01	01-02	01-03	01-04	01-05	01-06	01-07	01-08	01-09	1-10
Monday	4	11	(D) 18	25	2	(D) 9	16	23	30	6
Tuesday	5	12	19	26	3	10	17	24	31	7
Wednesday	(GC) 6	13	(VZ) 20	27	4	(GC) 11	18	25	1	8
Thursday	7	(PM) 14	21	28	(PM) 5	12	(PM) 19	26	2	9
Friday	(GC) 8	(PM) 15	(VZ) 22	29	(PM) 6	(GC) 13	(PM) 20	(D) 27	3	(D) 10
Saturday	09-09	16-09	23-09	30-09	7-10	14-10	21-10	28-10	4-11	11-11
Sunday	10-09	17-09	24-09	1-10	8-10	15-10	22-10	29-10	5-11	12-11

NEW!

- Progress meetings
  - ~30min / group
  - Mandatory
  - 14/15 September
  - 5/6 October
  - 19/20 October
- Team
  - Georgiana Caltais
  - Fernando Castor
  - Vadim Zaytsev

Week number	36	37	38	39	40	41	42	43	44	45
Week type	L	IM	L		IM	L	IM			
Quartile week	01-01	01-02	01-03	01-04	01-05	01-06	01-07	01-08	01-09	1-10
Monday	4	11	(D) 18	25	2	(D) 9	16	23	30	6
Tuesday	5	12	19	26	3	10	17	24	31	7
Wednesday	(GC) 6	13	(VZ) 20	27	4	(GC) 11	18	25	1	8
Thursday	7	(PM) 14	21	28	(PM) 5	12	(PM) 19	26	2	9
Friday	(GC) 8	(PM) 15	(VZ) 22	29	(PM) 6	(GC) 13	(PM) 20	(D) 27	3	(D) 10
Saturday	09-09	16-09	23-09	30-09	7-10	14-10	21-10	28-10	4-11	11-11
Sunday	10-09	17-09	24-09	1-10	8-10	15-10	22-10	29-10	5-11	12-11



Credits: Brian Butler

Lecture 1: Overview

- This course
- Software Architecture in a Nutshell
- Architecture Description
  - Scope. Environment. Software. Statements.

Software Architecture

What is it about?

Why do you need it?

Software Architecture 2022

What is it about?

- various components and relations among them
- architecture patterns
- data flow management
- use cases
- requirements
- business logic
- non-functional reqs

Why do you need it?

- to ease the development
- maintainability, ensure it
- to connect CEOs to devs
- communication is easier
- ensure robustness
- cost evaluation
- risk evaluation
- to solve a real life problem correctly

ISO/IEC/IEEE 42010

- *Conceptualisation of a system's architecture, as expressed in an architecture description, assists the understanding of the system's essence and key properties pertaining to its behaviour, composition and evolution, which in turn affect concerns such as the feasibility, utility and maintainability of the system.*

INTERNATIONAL STANDARD

ISO/IEC/ IEEE 42010

First edition
2011-12-01

Systems and software engineering — Architecture description

Ingénierie des systèmes et des logiciels — Description de l'architecture

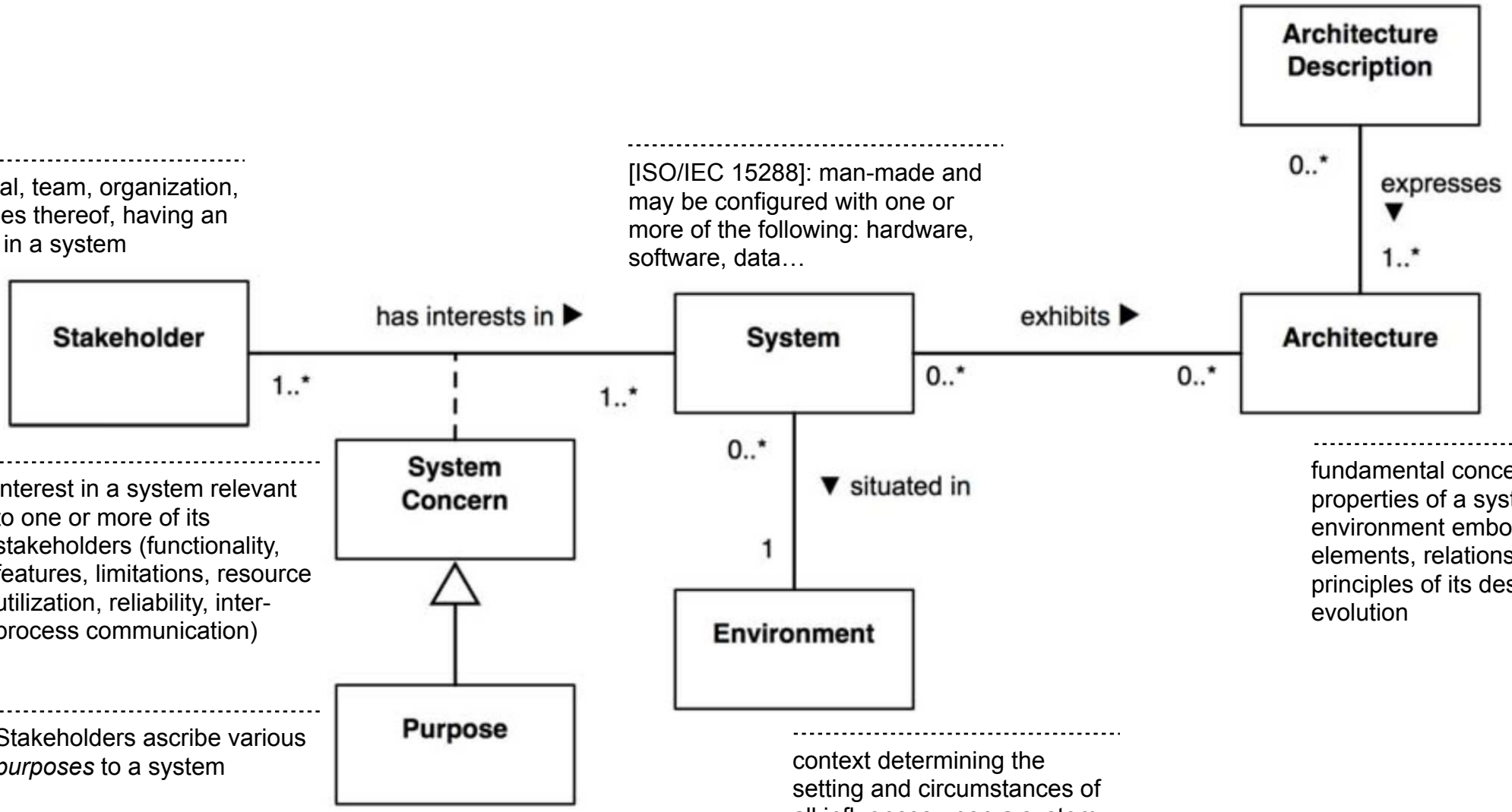
ISO IEC

IEEE

Reference number
ISO/IEC/IEEE 42010:2011(E)

ISO/IEC/IEEE 42010

(AD) work product used to express an architecture



individual, team, organization, or classes thereof, having an interest in a system

[ISO/IEC 15288]: man-made and may be configured with one or more of the following: hardware, software, data…

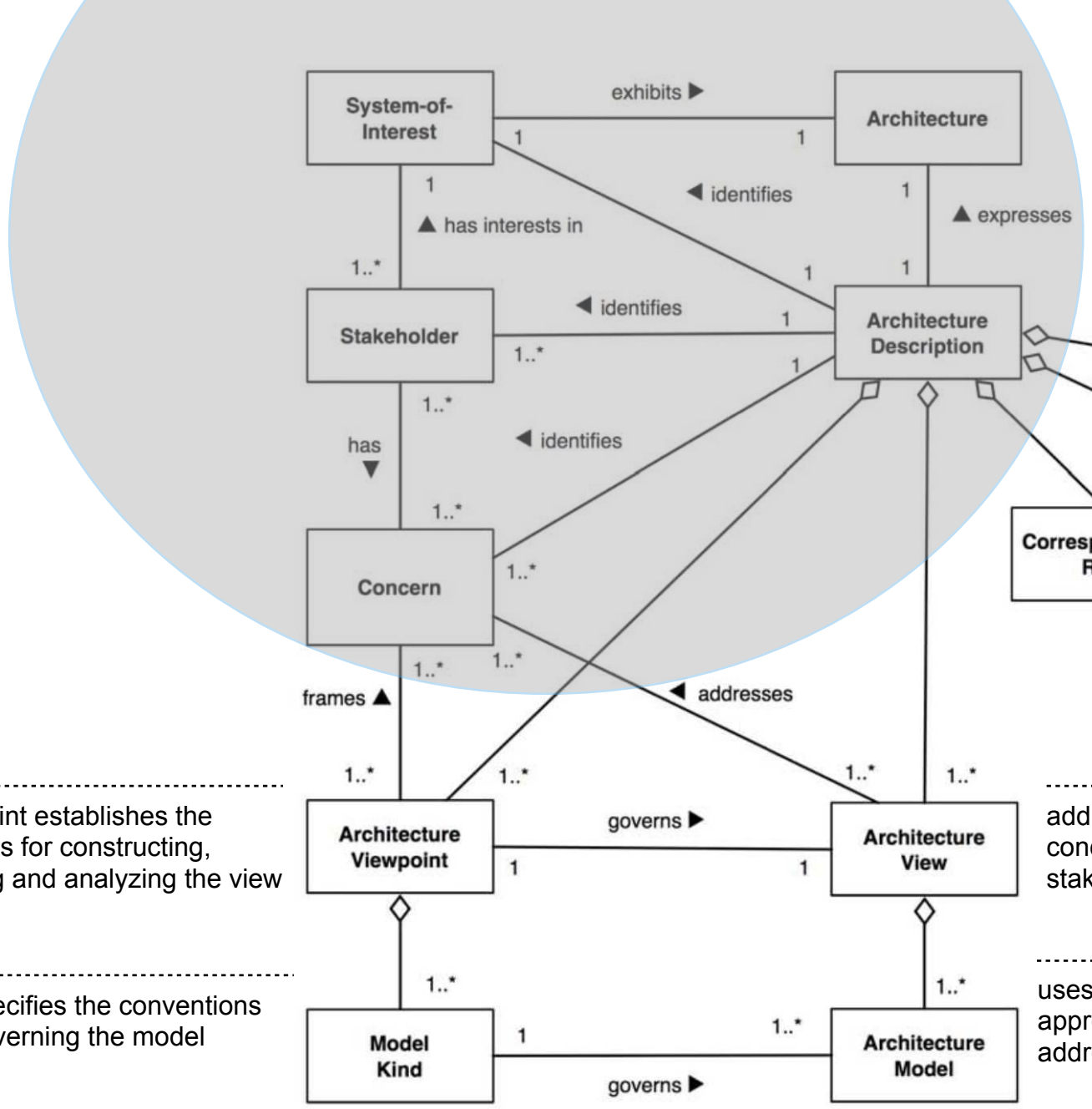
interest in a system relevant to one or more of its stakeholders (functionality, features, limitations, resource utilization, reliability, inter-process communication)

fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution

Stakeholders ascribe various *purposes* to a system

context determining the setting and circumstances of all influences upon a system

Architecture Description



records explanation, justification or reasoning about architecture decisions that have been made

defines a relation between AD elements (composition, dependency, constraint, obligation…)

the viewpoint establishes the conventions for constructing, interpreting and analyzing the view

addresses one or more of the concerns held by the system's stakeholders

specifies the conventions governing the model

uses modelling conventions appropriate to the concerns to be addressed

ISO/IEC/IEEE 42010:2011(E), §4.2.2

ISO/IEC/IEEE 42010

- Read the standard
  - know your way around
- [req] UML class diagram knowledge
- Be able to (intuitively) explain §3–5 concepts
  - stakeholder
  - viewpoint
- Ask questions if needed
- Read your case, record questions

INTERNATIONAL STANDARD

ISO/IEC/ IEEE 42010

First edition
2011-12-01

Systems and software engineering — Architecture description

Ingénierie des systèmes et des logiciels — Description de l'architecture

Reference number
ISO/IEC/IEEE 42010:2011(E)

Lecture 1: Overview

- This course
- Software Architecture in a Nutshell
- Architecture Description
  - Scope. Environment. Software. Statements.

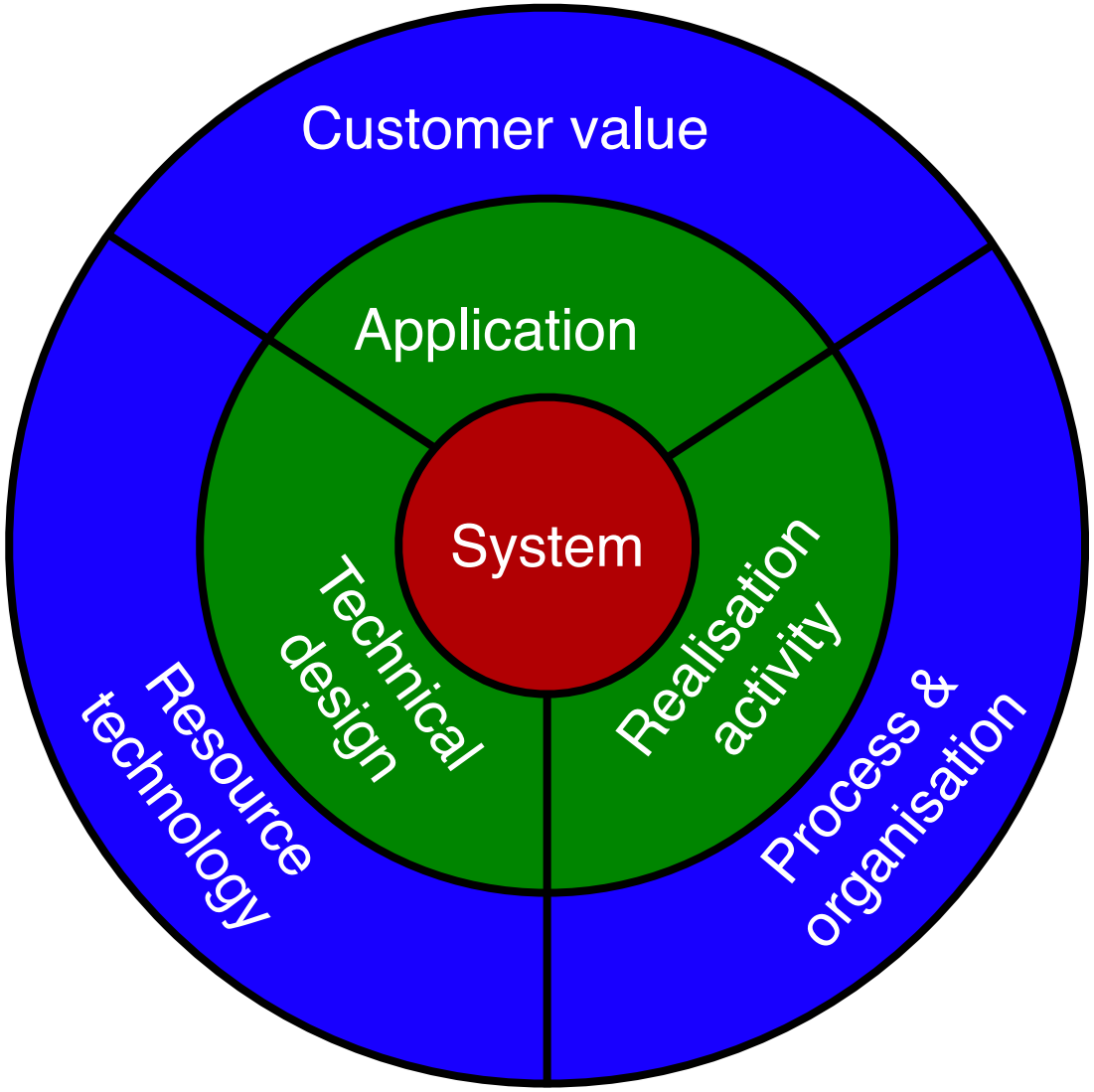
Software architecture

- The software architecture of a system is
  - a collection of statements
  - that gives direction to
    - the application
    - the design
    - the realisation
  - of the software in its environment
- Statements are a model of:
  - structure of the system elements and their relations, or
  - guidelines for creating structure, elements and their relationships

Architectural differences

Define your scope

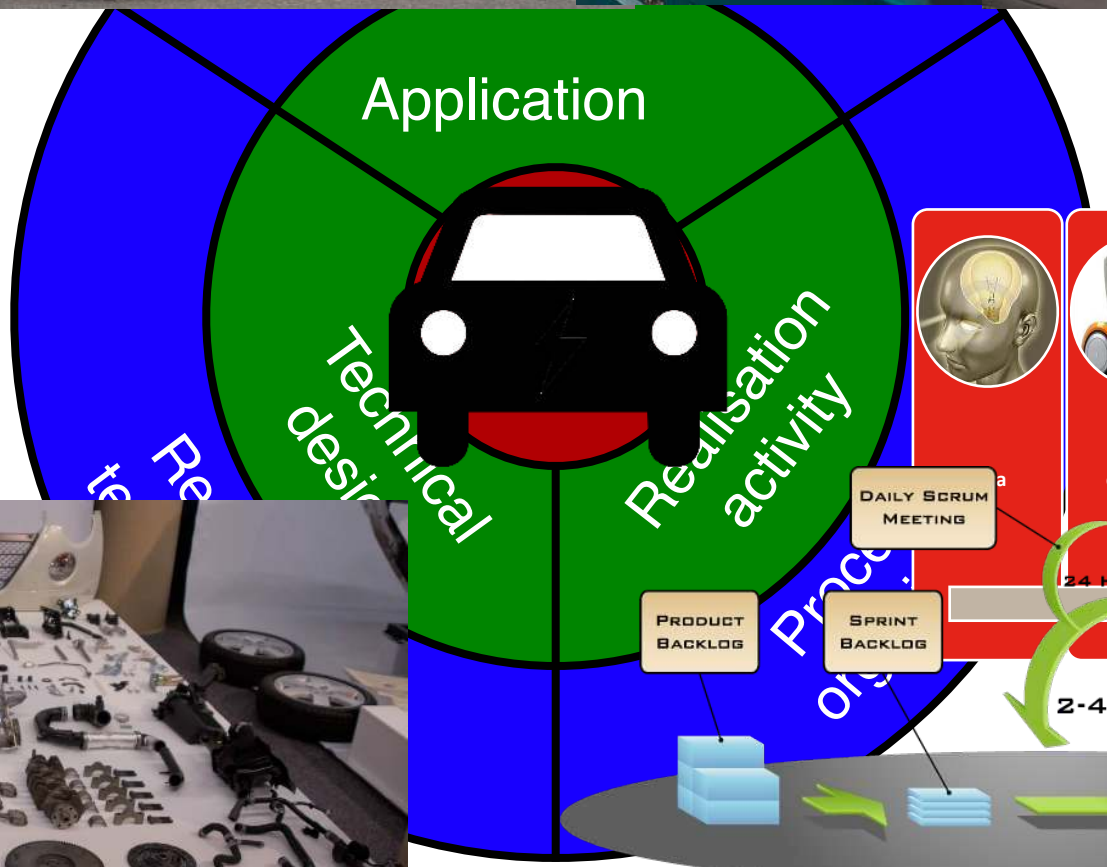
- Define the scope of your system
  - the essence in one or two sentences
- Add details about
  - application
  - design
  - realisation
- The system boundary is up to you
- Make explicit what your architecture will cover and what is out of your scope
- (This may change during your project)



"Architecture Reasoning Model" by Robert Deckers


BIKE EXCHANGE


LKH-98B

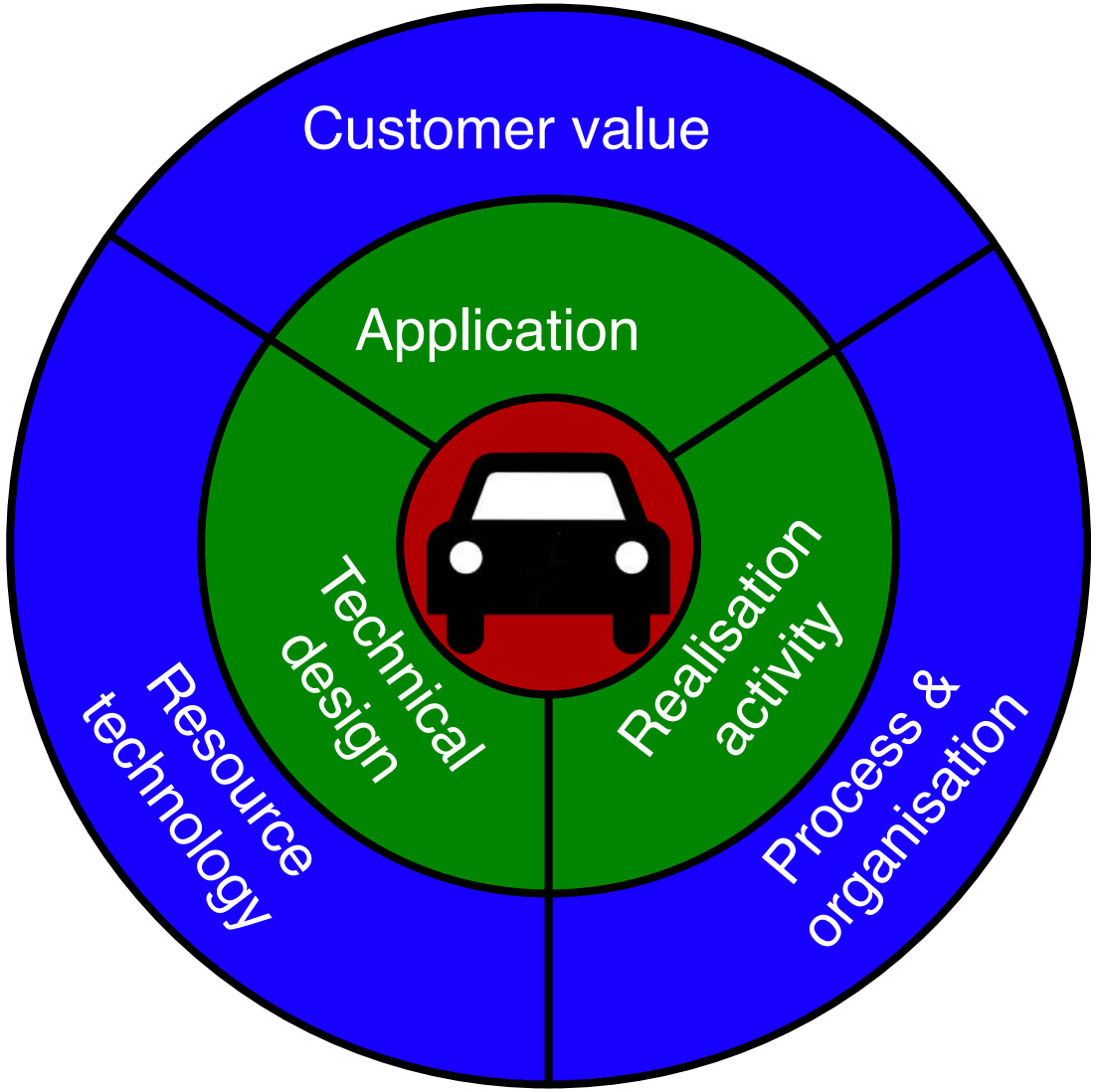

Application
Realisation activity


concept
development
Use
maintenance
End of life
Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
Potentially Shippable Product Increment
2-4 Weeks

Software architecture

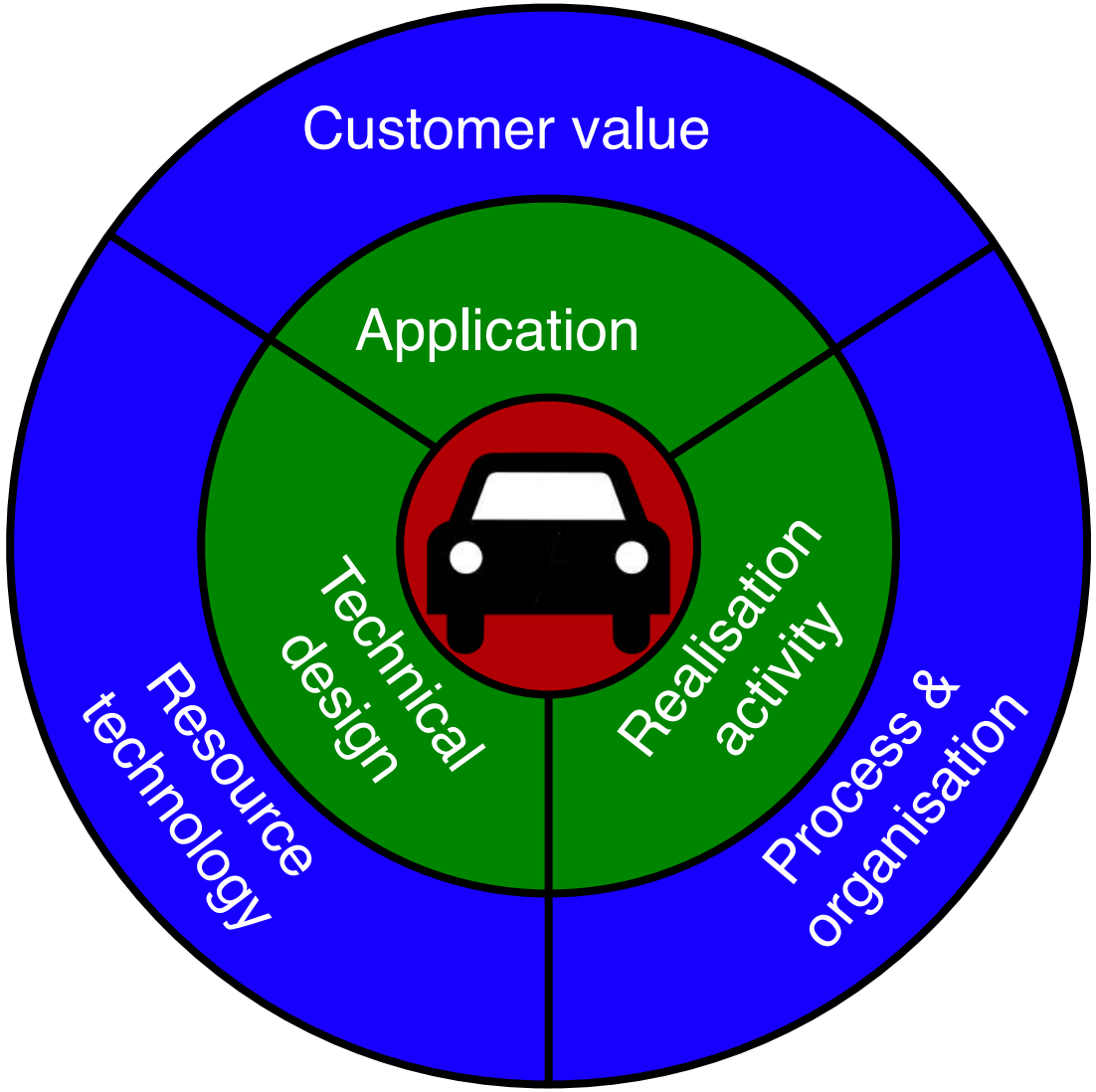
- The software architecture of a system is
  - a collection of statements
  - that gives direction to
    - the application
    - the design
    - the realisation
  - of the software in its environment
- Statements are a model of:
  - structure of the system elements and their relations, or
  - guidelines for creating structure, elements and their relationships

Environment elements?



Environment elements 2022

- physical location
- laws
- people
- device
- time
- systems to interact
- configuration
- resources
- topology
- communication
- guidelines
- API
- constraints

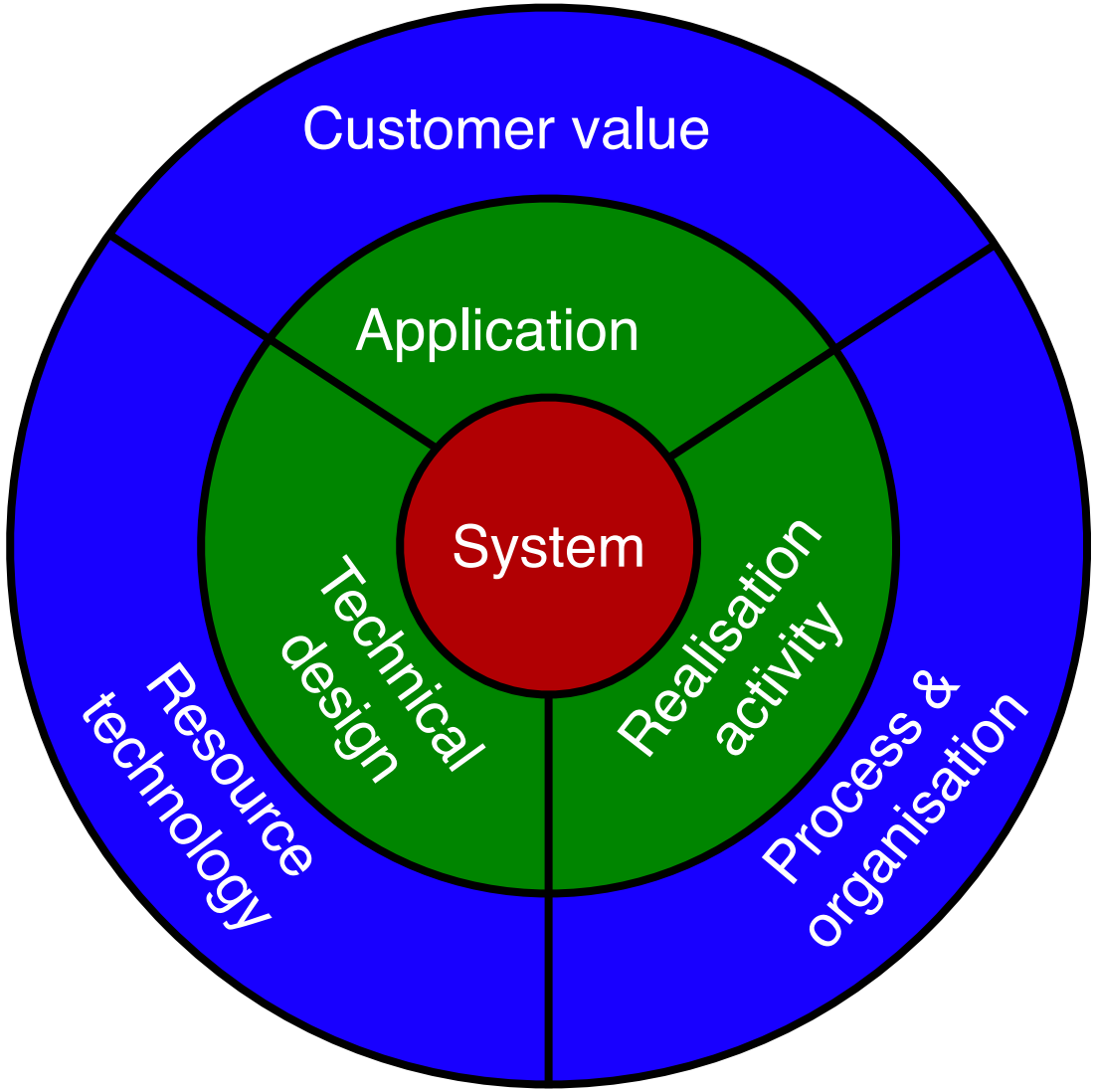


Ways to interact with the env?

Explore the environment

- What is in the environment of your system?

- Add details about
  - application
    - who? for what? in which context?
  - design
    - technologies? platforms? others?
  - realisation
    - process? organisations? activities?



"Architecture Reasoning Model" by Robert Deckers

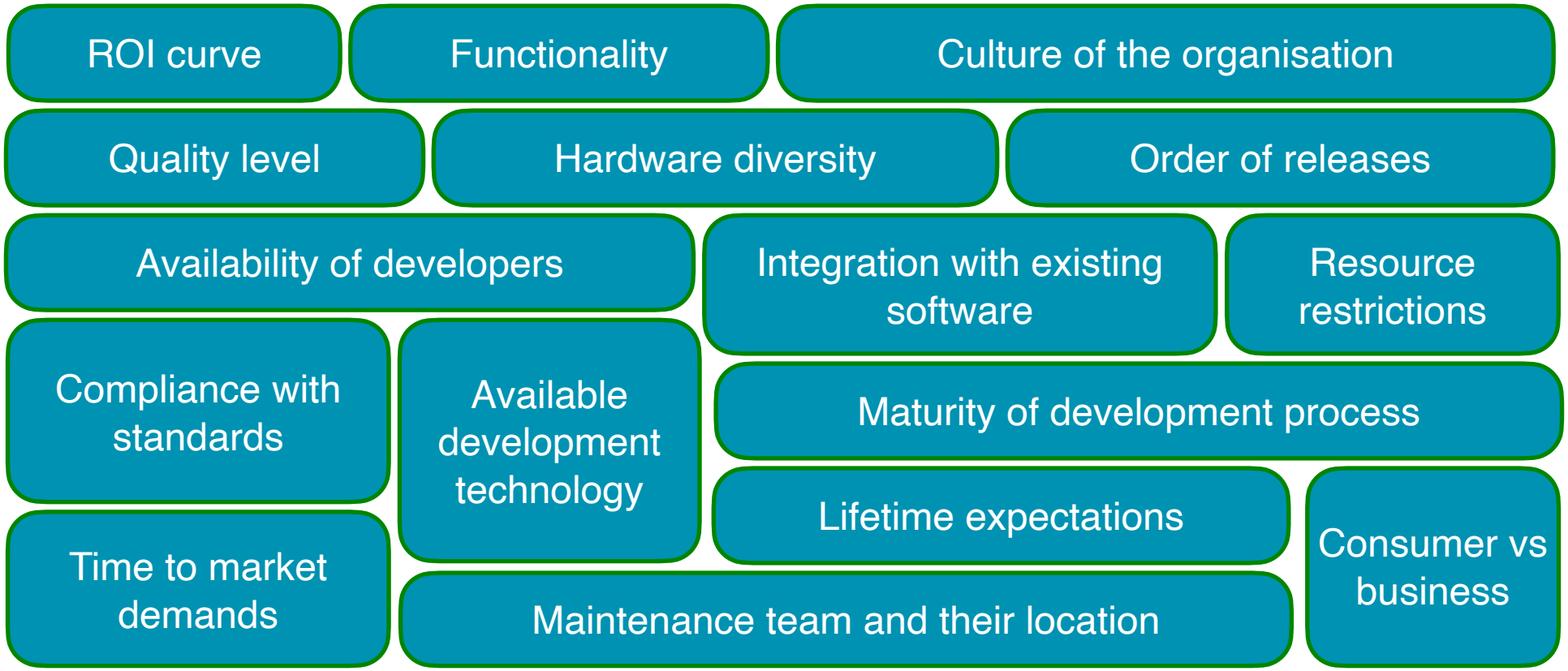
Software architecture

- The software architecture of a system is
  - a collection of statements
  - that gives direction to
    - the application
    - the design
    - the realisation
  - of the software in its environment
- Statements are a model of:
  - structure of the system elements and their relations, or
  - guidelines for creating structure, elements and their relationships

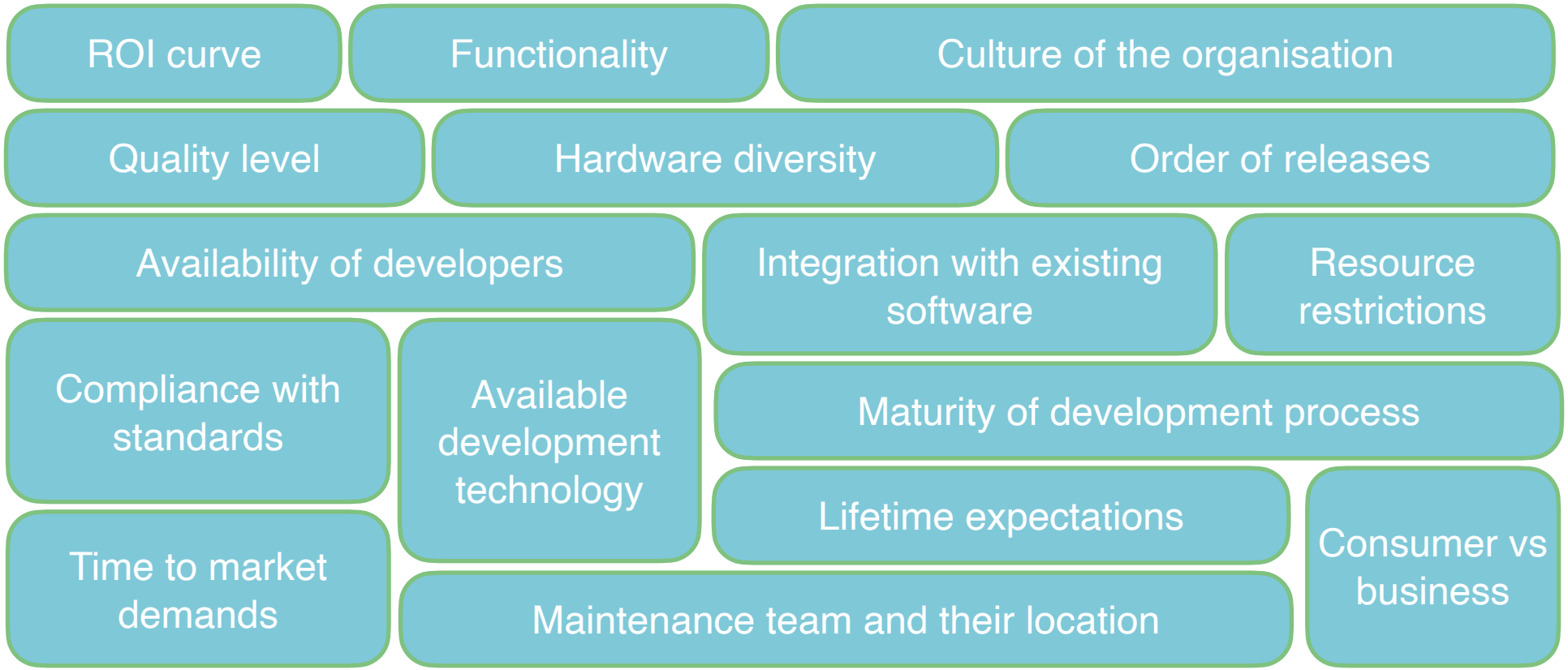
What can be software?

- Think of things that could be software in your system
  - How does it look like? Language? Protocol? Terminology?
  - What are its applications? Where do you use it for?
  - What is the *"machine"* that processes it?
- Softwarisation is:
  - automation of activities
  - offering a specification environment
  - replicability and testability

What is important in software?



What is important for software?



Software architecture

- The software architecture of a system is
  - a collection of statements
  - that gives direction to
    - the application
    - the design
    - the realisation
  - of the software in its environment
- Statements are a model of:
  - structure of the system elements and their relations, or
  - guidelines for creating structure, elements and their relationships

Statements in architecture

- A **statement** is:
  - something that someone has
    - said, written, or outed in another way
- Statements are related through terminology
- Statements are coherent and not contradictory

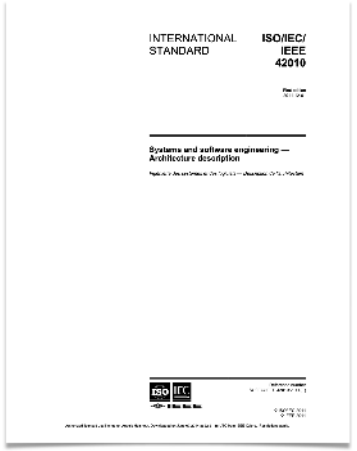
Where's the language?

- Domains?
- Experts?
- Terminology? (concepts)
- Syntax? (form)
- Who speaks the language?
- Who must read it?

Homework

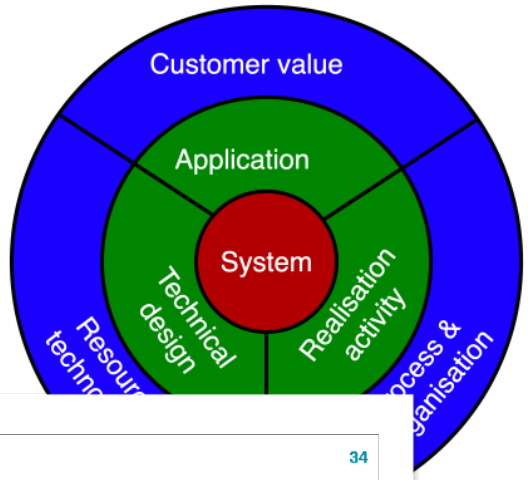
ISO/IEC/IEEE 42010

- Read the standard
  - know your way around
- [req] UML class diagram knowledge
- Be able to (intuitively) explain §3–5 concepts
  - stakeholder
  - viewpoint
- Ask questions if needed
- Read your case, record questions



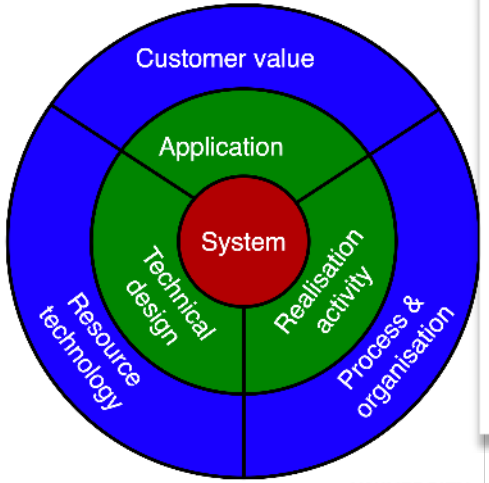
Define your scope

- Define the scope of your system
  - the essence in one or two sentences
- Add details about
  - application
  - design
  - realisation
- The system boundary is up to you
- Make explicit what your architecture



Explore the environment

- What is in the environment of your system?
- Add details about
  - application
    - who? for what? in which context?
  - design
    - technologies? platforms? others?
  - realisation
    - process? organisations? activities?



"Architecture Reasoning Model" by Robert Deckers

What can be software?

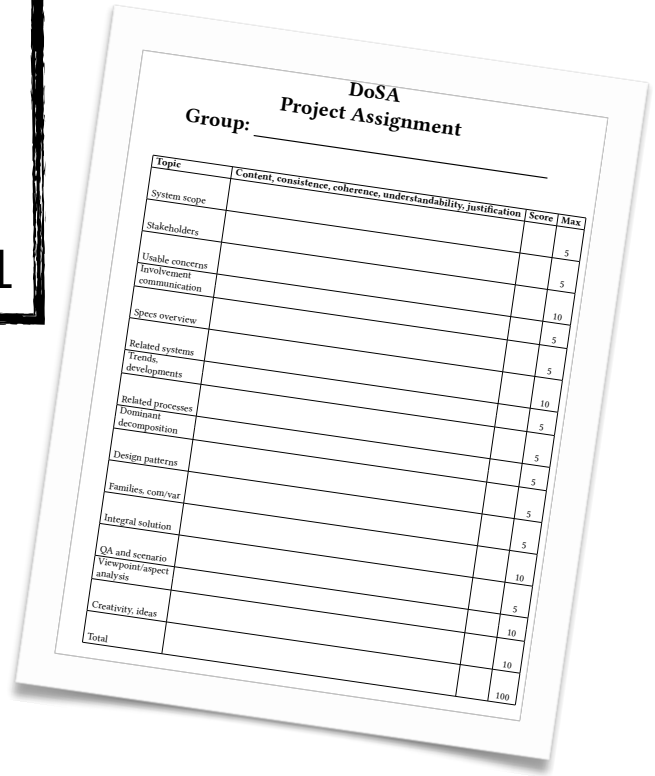
- Think of things that could be software in your system
  - How does it look like? Language? Protocol? Terminology?
  - What are its applications? Where do you use it for?
  - What is the *"machine"* that processes it?
- Softwarisation is:
  - automation of activities
  - offering a specification environment
  - replicability and testability

UNIVERSITY OF TWENTE.

Group Assignment: SA Description

Suggested Structure

TODAY!
Create a team (6-7)
Choose your case
Email to:
g.g.c.caltais@utwente.nl



DoSA
Project Assignment
Group: _____

Topic	Content, consistence, coherence, understandability, justification	Score	Max
System scope			5
Stakeholders			5
Usable concerns			10
Involvement communication			5
Specs overview			5
Related systems			10
Trends, developments			5
Related processes			5
Dominant decomposition			5
Design patterns			5
Families, com/var			5
Integral solution			10
QA and scenario			5
Viewpoint/aspect analysis			10
Creativity, ideas			10
Total			100

- Introduction
- System scope
- Environment
  - Stakeholders and their concerns
  - Related systems
  - Related processes, organisations, stakeholders
  - Main QA scenarios and use cases
  - Trends and developments
- Design
  - Dominant decomposition/style
  - Design decisions: Justification of choices related to environment
  - View 1..N
    - Per view: Used design patterns (show examples; if you cannot, generalise)
  - Consistency: relation between views, known gaps/inconsistencies.
- Appendix
  - Explanation of viewpoints, references to standards

cf. ISO 42010!