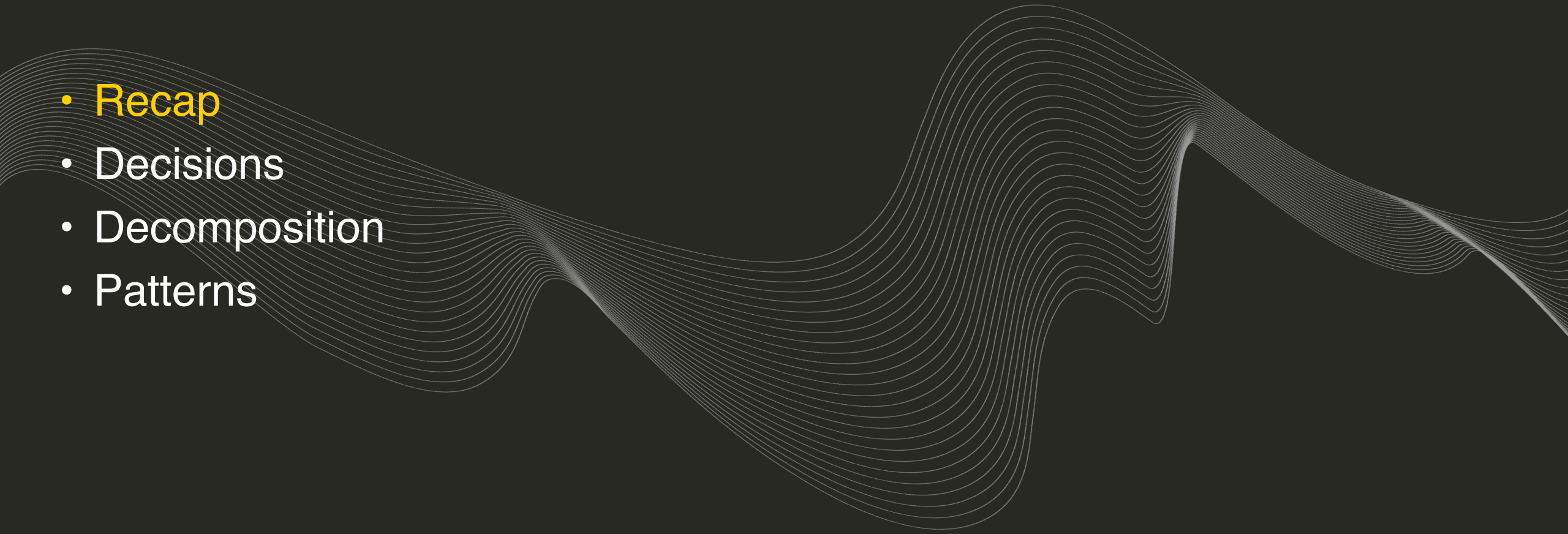
Lecture 7

Design of Software Architectures

Dr. Georgiana Caltais, Dr. Vadim Zaytsev

UNIVERSITY OF TWENTE.

Lecture 7: Overview

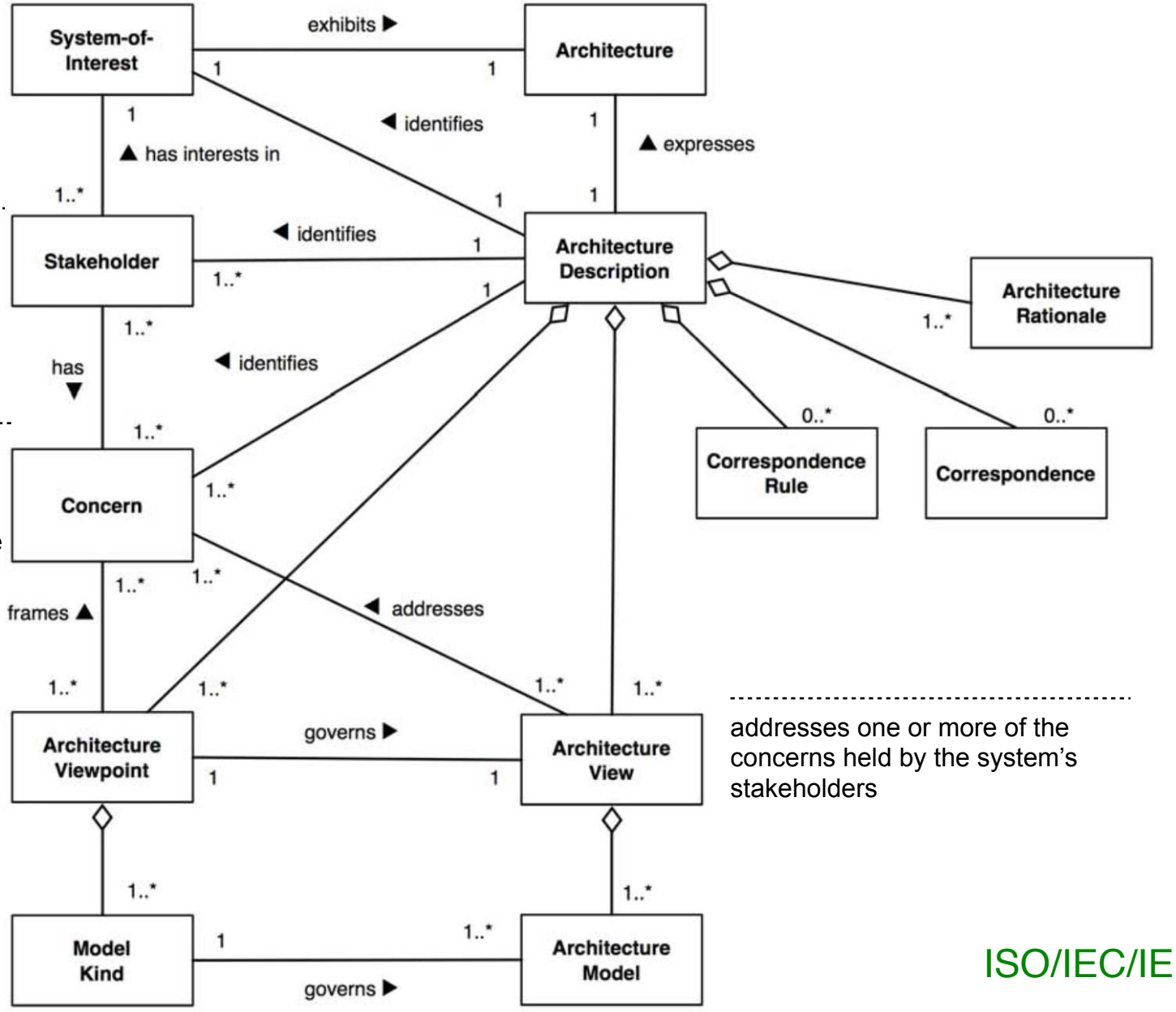


- Recap
- Decisions
- Decomposition
- Patterns

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

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

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

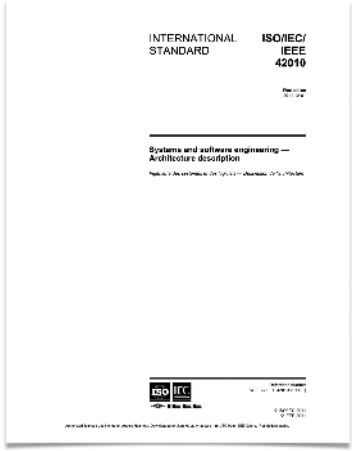
ISO/IEC/IEEE 42010:2011(E), §4.2.2

Homework - L1

H 22

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

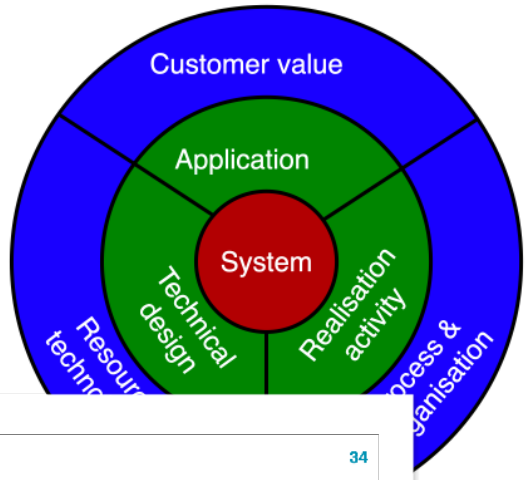


UNIVERSITY OF TWENTE.

H 26

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



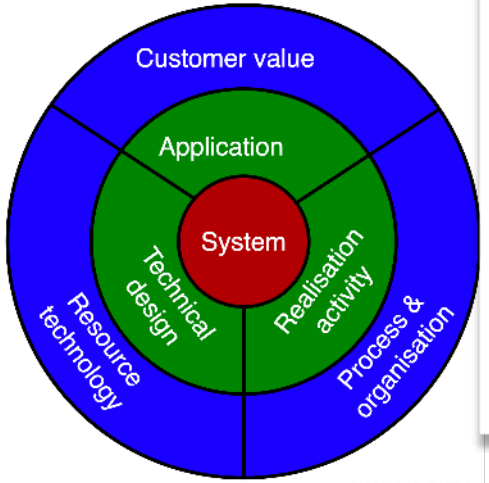
UNIVERSITY OF TWENTE.

H 32

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

UNIVERSITY OF TWENTE.

H 34

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.

UNIVERSITY OF TWENTE.

Homework - L2

Make it good

Good architecture is correct

- The architecture is based on
  - validated statements about the environment
  - (the stakeholder concerns in particular)
- Concerns are prioritised
- Architecture balances the concerns
- Achieved by:
  - Adequate environment analysis
  - Conscious balance of interests
  - Validation of environment statements

Good architecture is consistent

- The architecture forms a whole
- Architecture statements do not conflict
- The system can become reality with the architecture
- Achieved by:
  - Verification on contradictions
  - Demonstrable PoCs
  - Conscious management

Good architecture is communicated

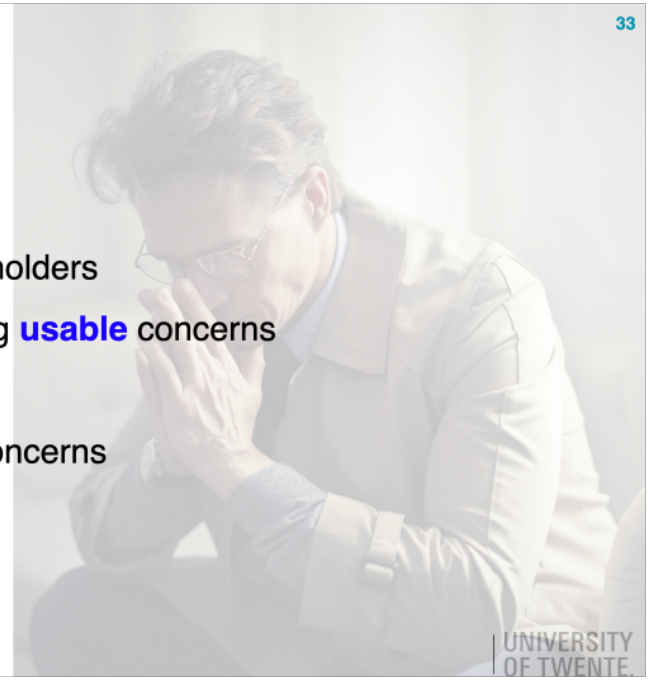
- Stakeholders know their relation with the architecture
- Stakeholders know what to do with it
- Communicating architecture starts with a consumable
- Achieved by:
  - Actionability by stakeholders
  - Sufficient anchoring
  - Conscious performance

- Concretise the abstract advice
- Formulate guidelines for your project
  - correctness
  - consistency
  - communication

UNIVERSITY OF TWENTE.

Usable concerns

- Define stakeholder concerns
  - for 10+ most important stakeholders
- Apply the guidelines for acquiring **usable** concerns
- Brainstorm ideas
  - how to address/solve top 5 concerns



UNIVERSITY OF TWENTE.

UNIVERSITY OF TWENTE.

Homework - L3

Specify domains, identify aspects

- What are your domains?
- What are aspects relevant for your project?
- Think of:
  - Application
  - Design
  - Realisation
- Just list them and give a short definition

Select viewpoints, clean up views

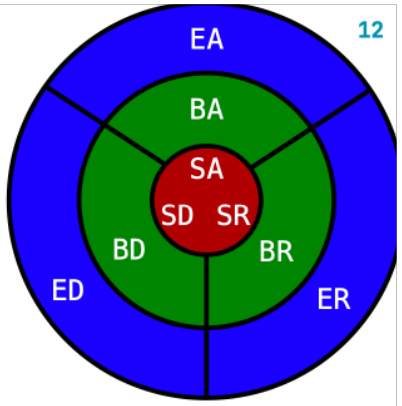
- Which stakeholder(s)? Which concerns?
- Decide on the language:
  - Concepts and notation
  - Visualisation (depends on communication context)
- Specify how to translate the model(s) into a view.
- Pitfalls:
  - Too little views (e.g., just the decomposition)
  - Overloaded views (everything in one picture)

Homework - L4

Identify related systems

- Define which type of relation each has with yours
- with respect to
  - Functionality
  - Design
  - Realisation
- Visualise relations between your syste and systems in the environment.

Requirements



- Distribute workload systematically
- Pick three main stakeholders, per stakeholder:
  - identify concerns (goal, task, needed system property)
  - write requirements/use cases, put on the circle
- Each req
  - fulfils a need, is verifiable, realisable, atomic, traceable
- All reqs
  - prioritised, grouped, unique, cover the domain, free of conflicts
- Prioritise according to sat/dissat level (per stakeholder)
- What was difficult? (be specific!)

Homework - L6

Map architecture onto the process

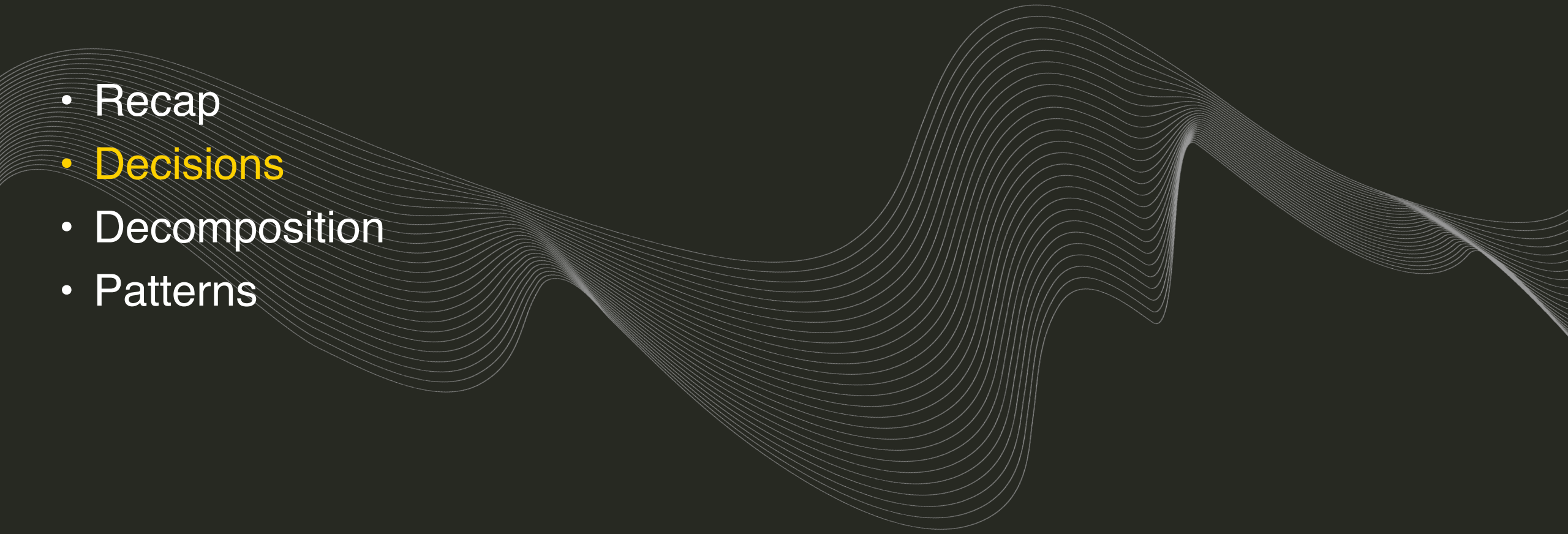
- Activities, artefacts, constraints, standards, systems, people
- What process comes after your architecture document:
  - main steps
  - involved roles (teams, stakeholders)
- How is the architecture used by the development process?
  - Which of your views is used by whom?
  - How is the architecture applied properly?
  - How can you verify if it is?

UNIVERSITY OF TWENTE.

View(points)

- Read the standard (Appendix A, B, C)
  - View addresses one or more of the concerns held by the system's stakeholders
  - Metaphors: program / programming language; map / legend
- Various other resources, e.g.:
  - http://www.iso-architecture.org/ieee-1471/faq.html#whvvps
  - Viewpoint documents the conventions for constructing, interpreting and analyzing a particular kind of view (languages, notations, model types, modeling methods, analysis techniques, design rules and any associated methods)

Lecture 7: Overview



- Recap
- Decisions
- Decomposition
- Patterns

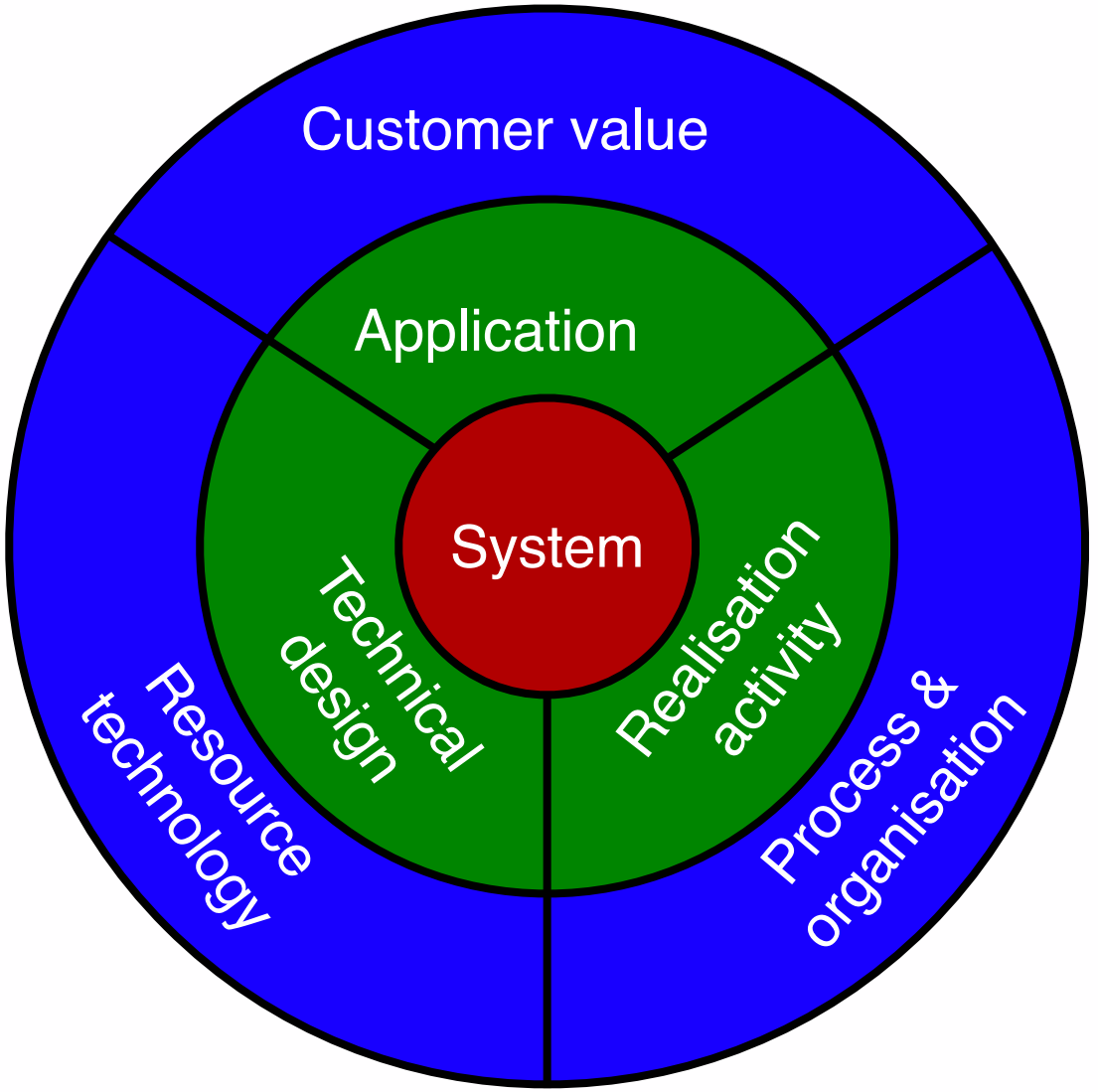
Role of the software architecture

ensure to have sufficient decisions

about the right things

to allow stakeholders
to validate, use, apply them

Realisation domain



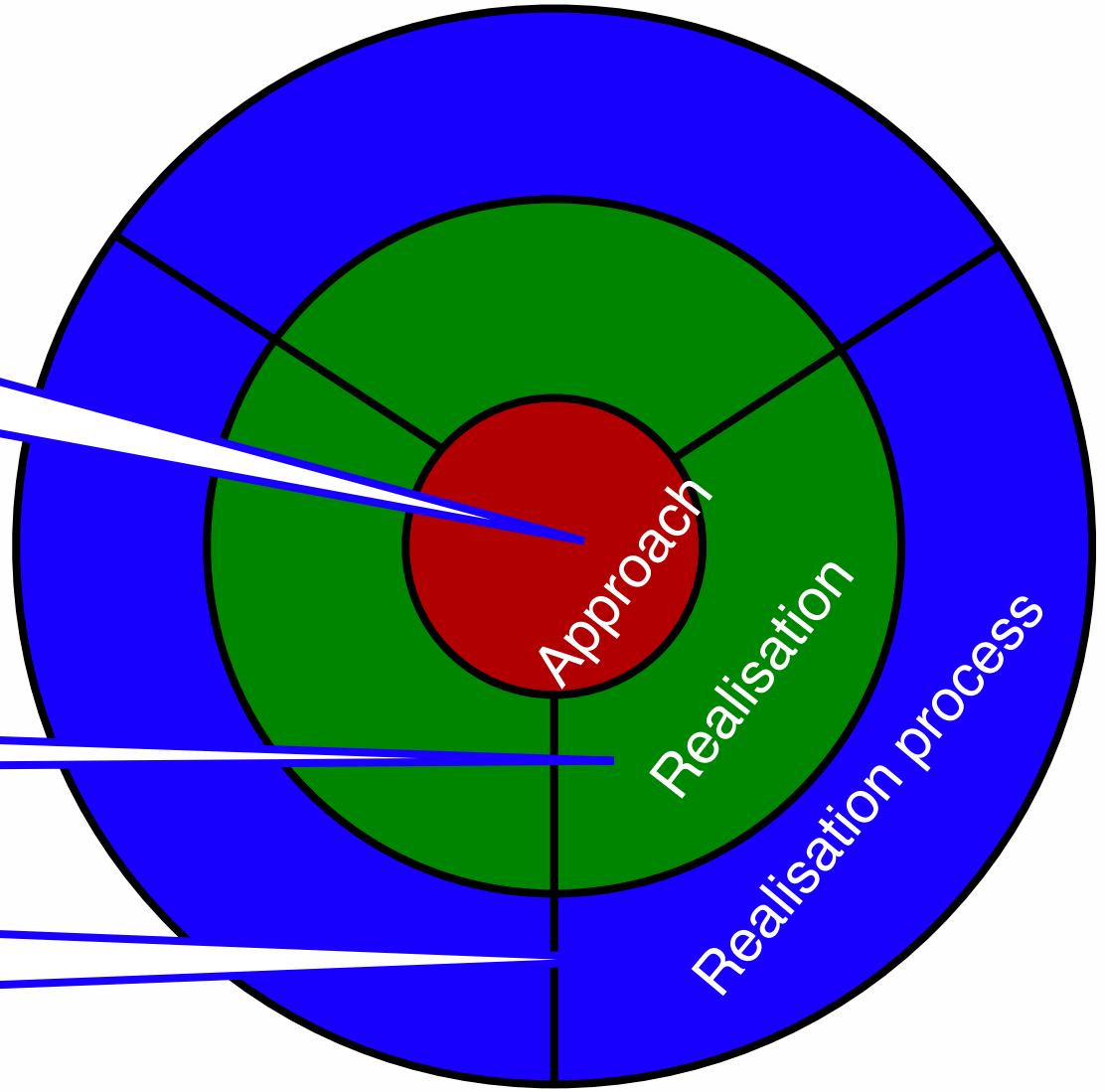
by Robert Deckers, based on ISO 42010 & DYA

Realisation domain

By which activities & what behaviour?
- main tasks
- phases
- milestones

Who does what when?
- planning, roadmap,
- responsibilities,
- work packages

Which organisation/people?
- team structure,
- knowledge/skills,
- education/training



by Robert Deckers, based on ISO 42010 & DYA

Decisions!

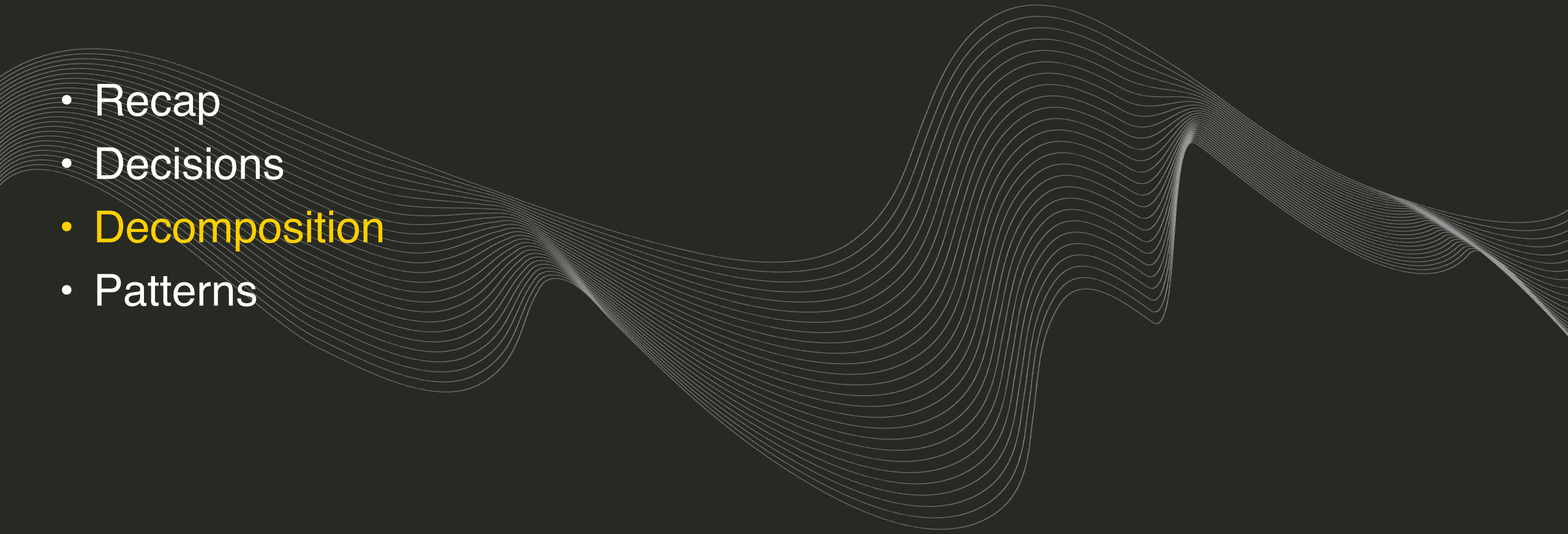
- The development activity is basically a group “actors” taking decisions that:
  - happen in certain context:
    - time: during a project, in maintenance, in a sprint…
    - space: specific persons belonging to the development activity
  - matter:
    - decision topic is relevant
    - remain (in specifications and heads)
  - finally result in a *formal* machine-readable specification
    - source code
    - configuration settings
    - rules, models

Who decides what and when

- Who are the deciding actors?
- What do they decide about?
- When are those decisions on the timeline?

- What happens when you swap an early & a late decision?

Lecture 7: Overview

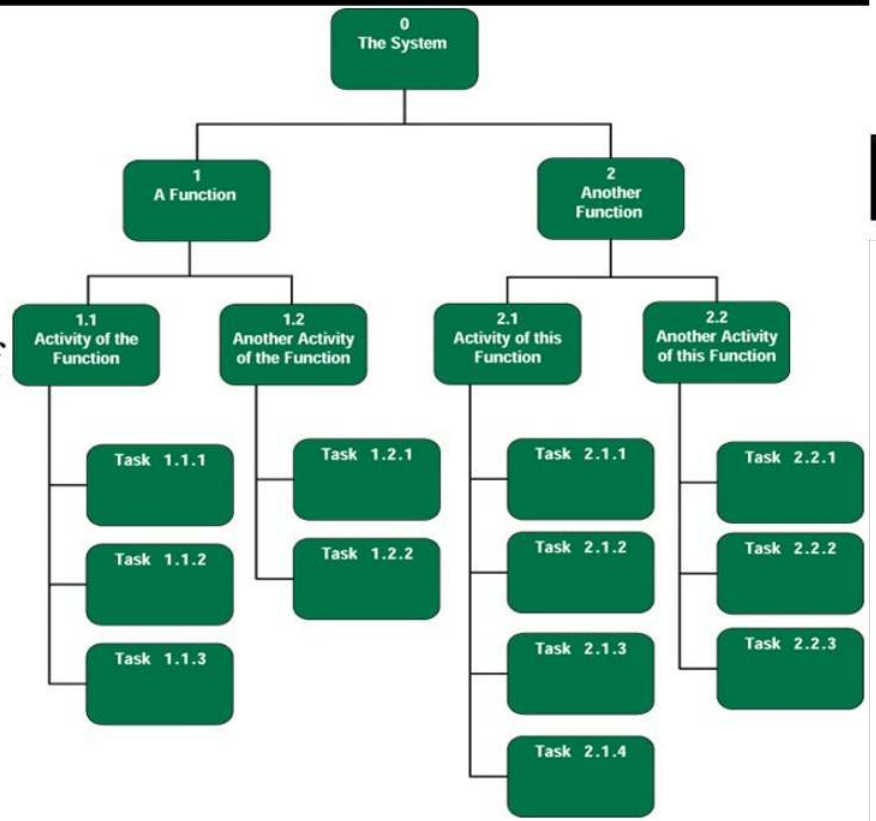


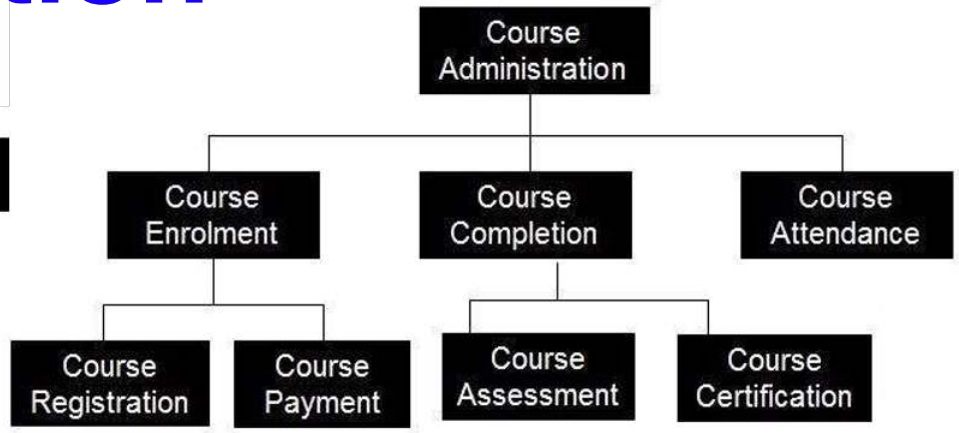
- Recap
- Decisions
- Decomposition
- Patterns

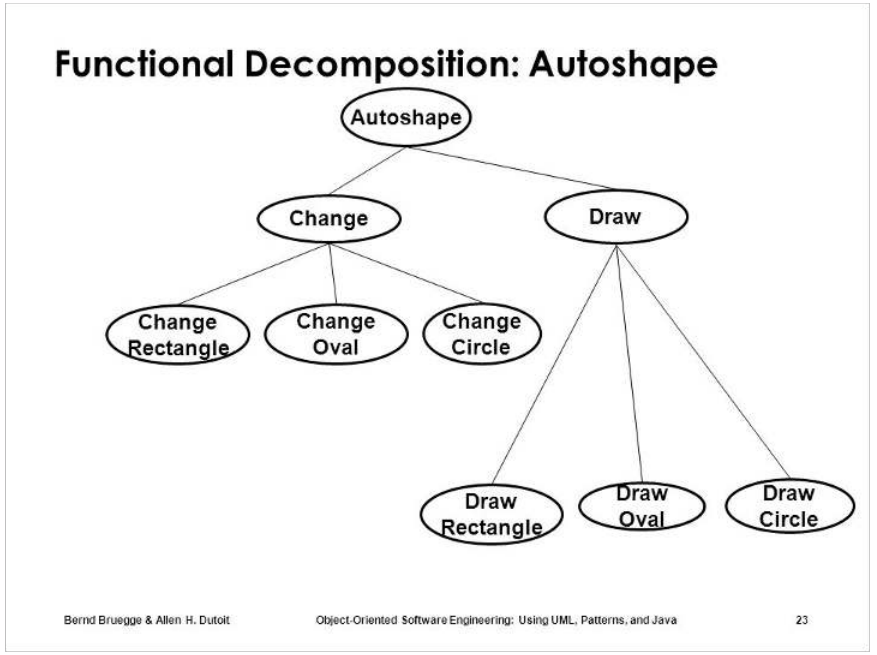
Functional decomposition

Decomposition Diagrams

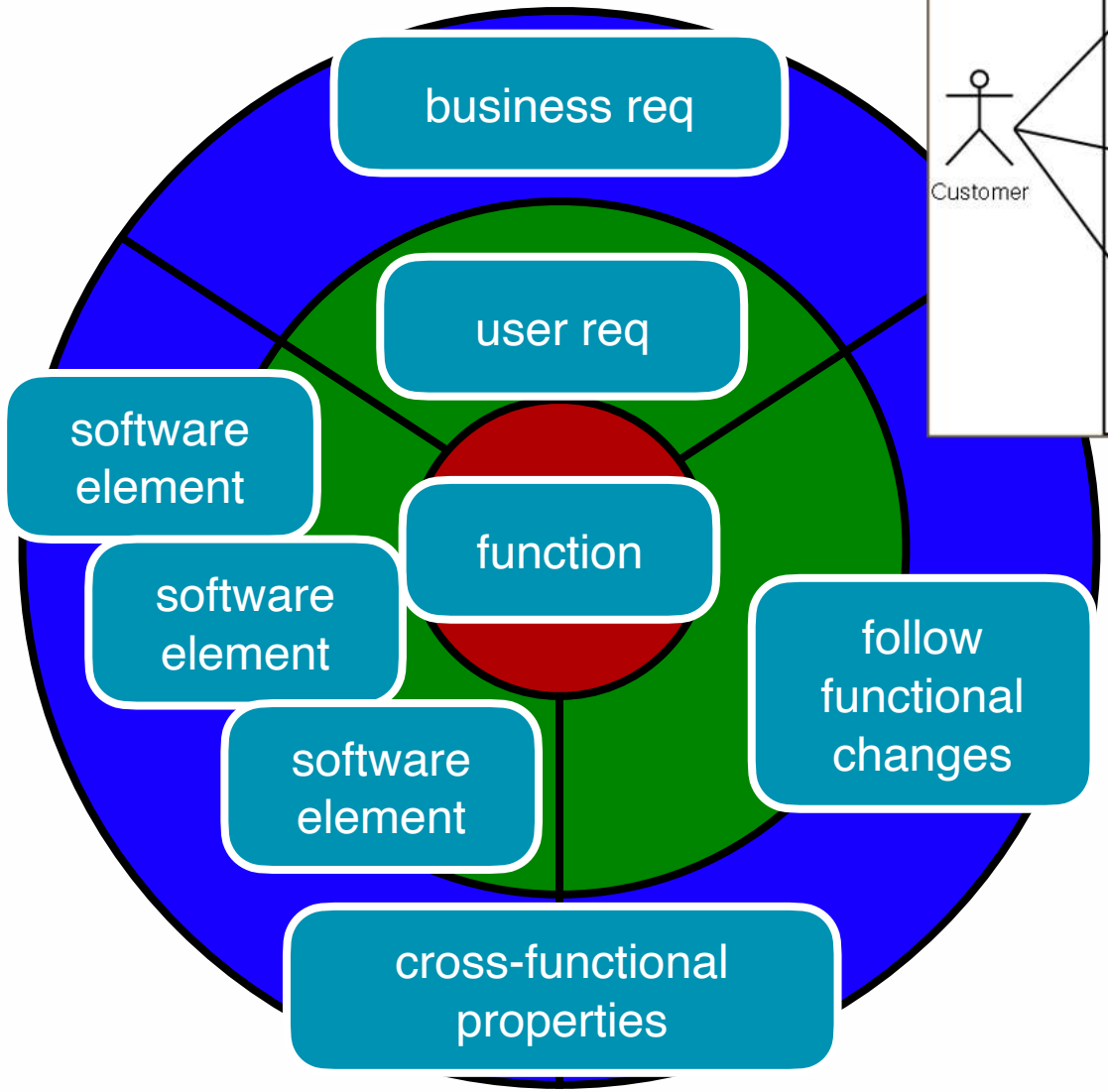
A decomposition diagram or hierarchy chart shows the top-down, functional decomposition of a system.

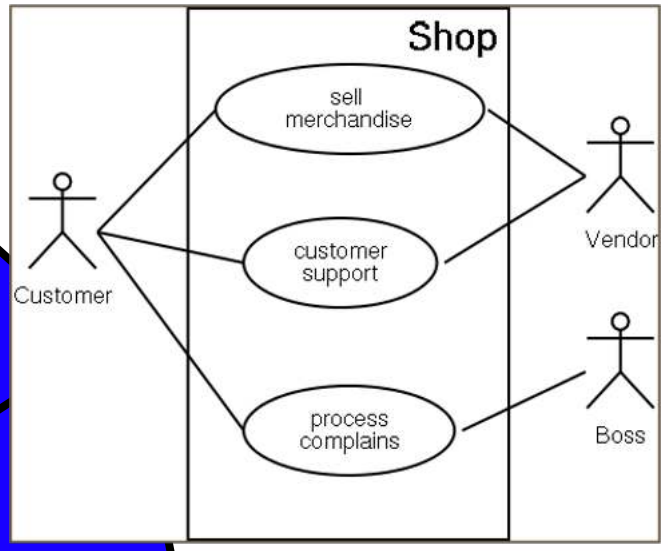




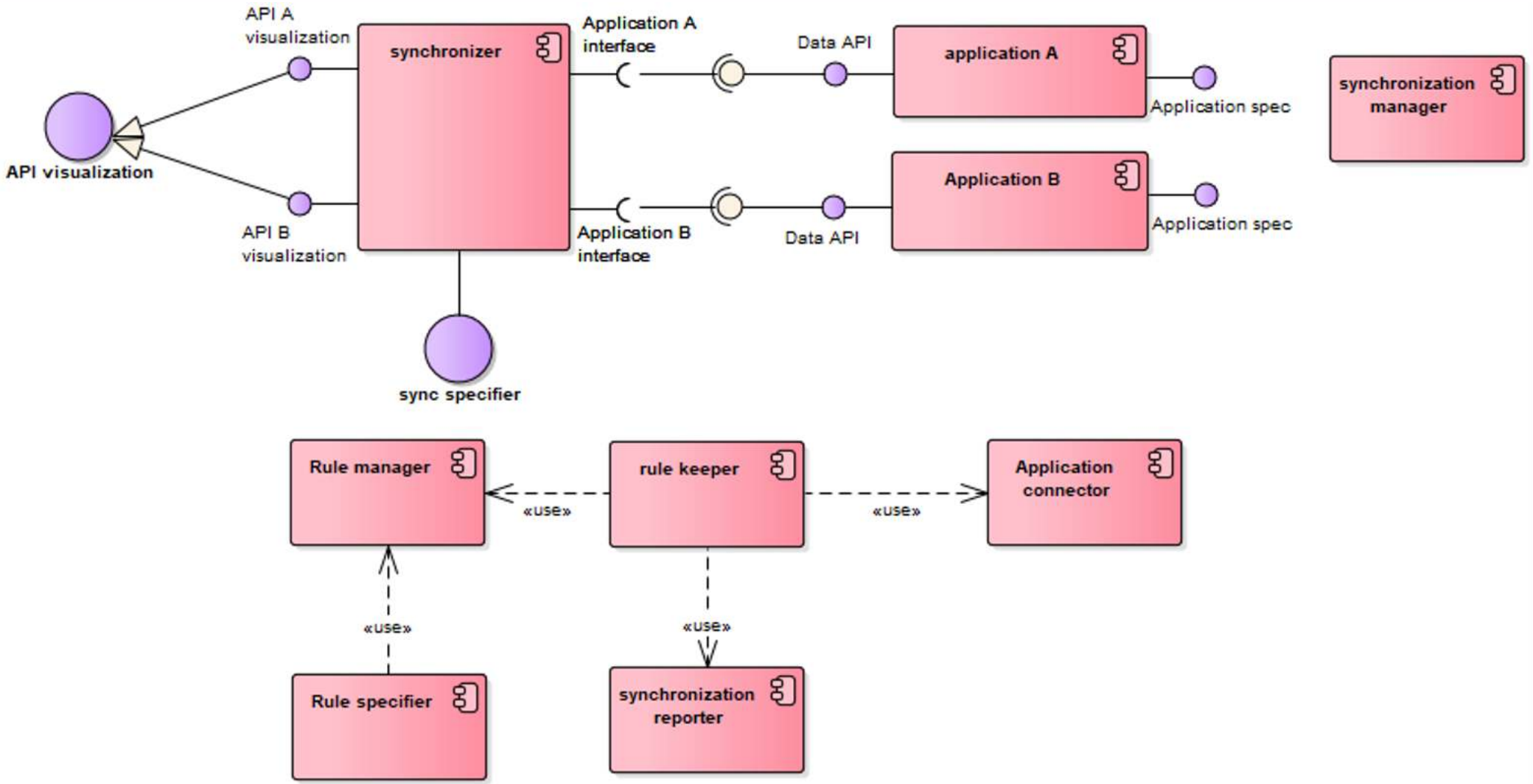


Functional decomposition

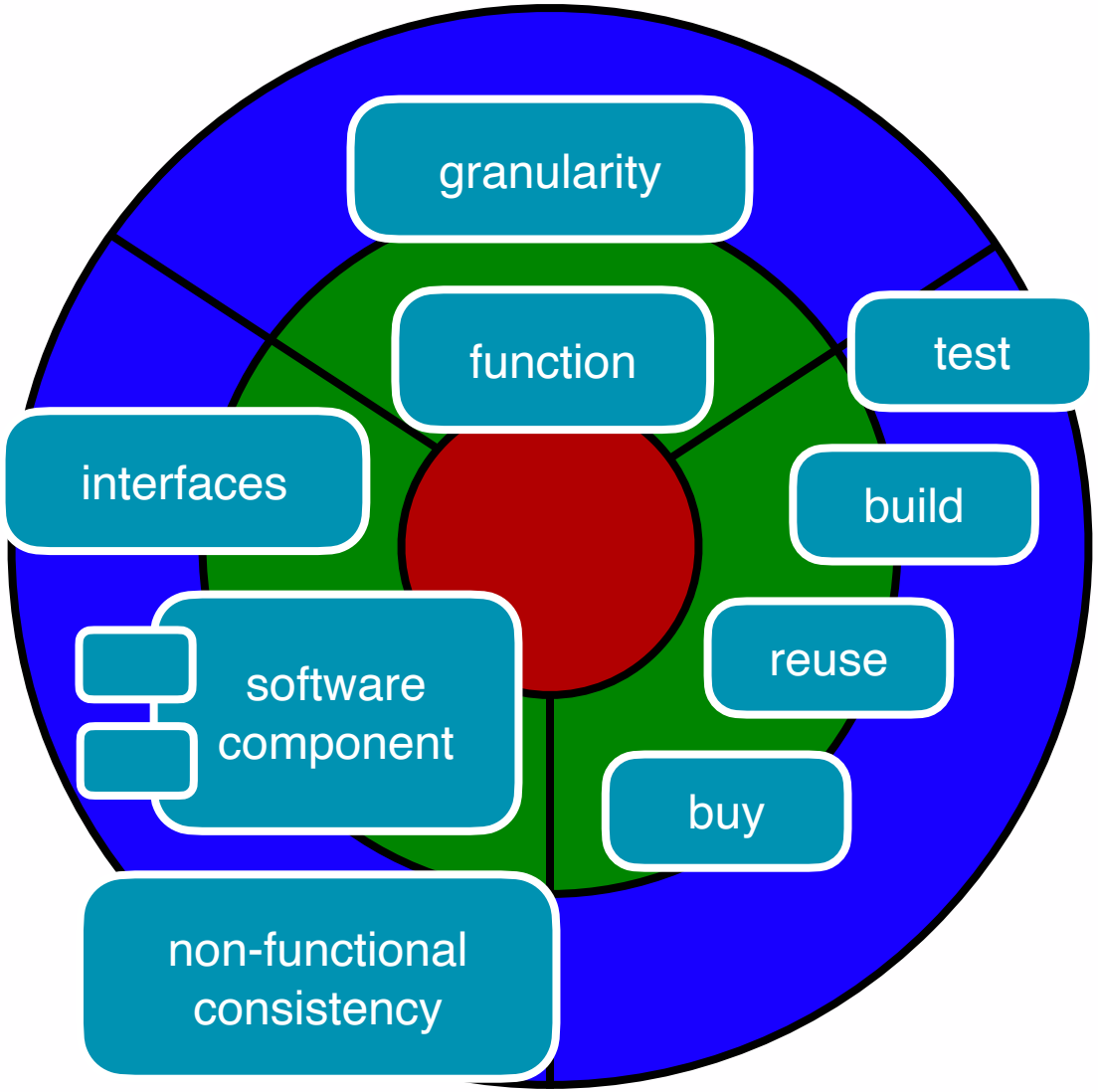




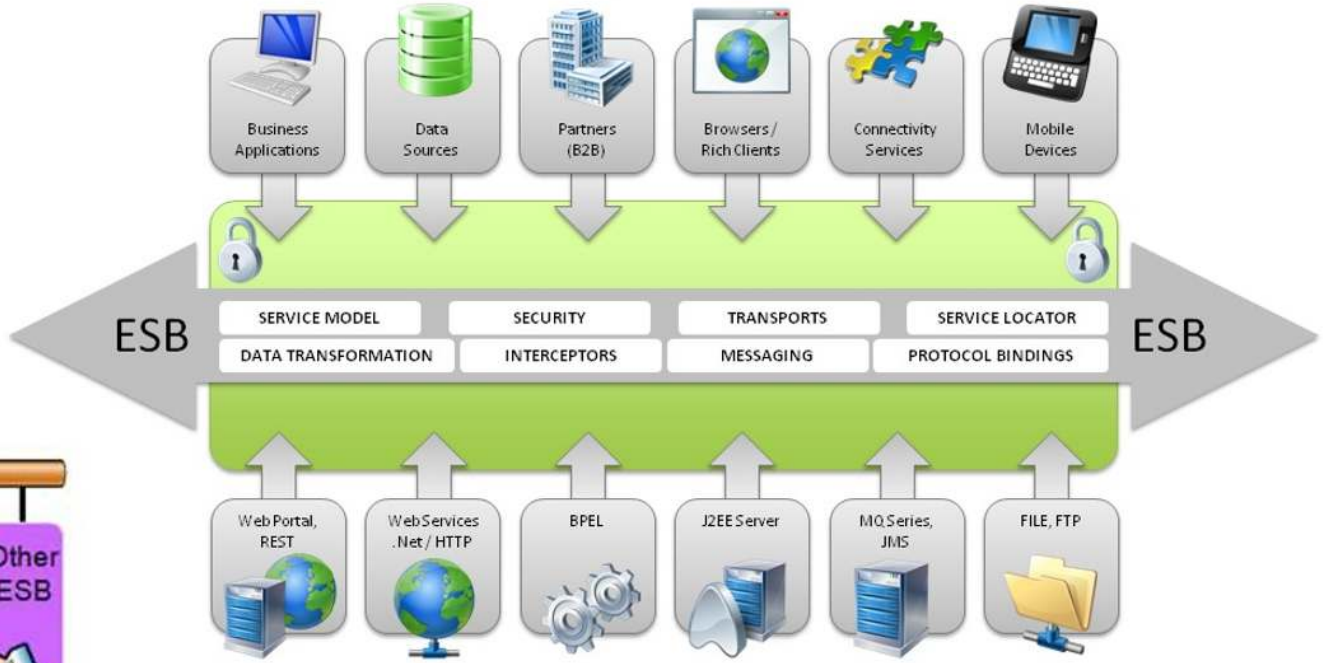
Components

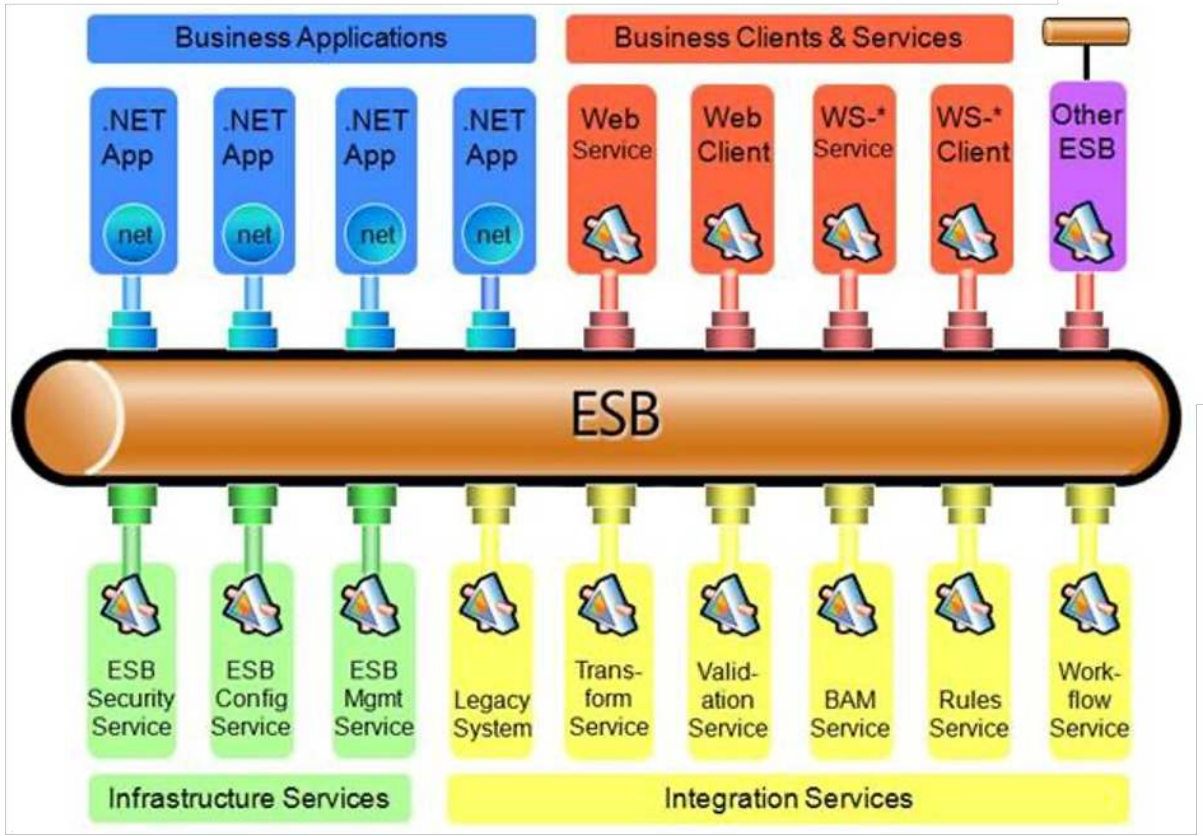

API A visualization
synchronizer
Application A interface
Data API
application A
Application spec
synchronization manager
API visualization
Application B
Application spec
API B visualization
Application B interface
Data API
sync specifier
Rule manager
«use»
rule keeper
«use»
Application connector
«use»
«use»
Rule specifier
synchronization reporter

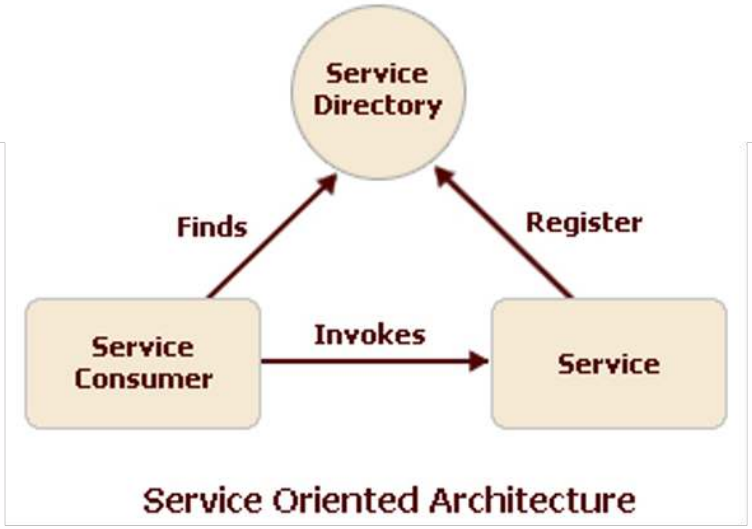
Component-based architecture



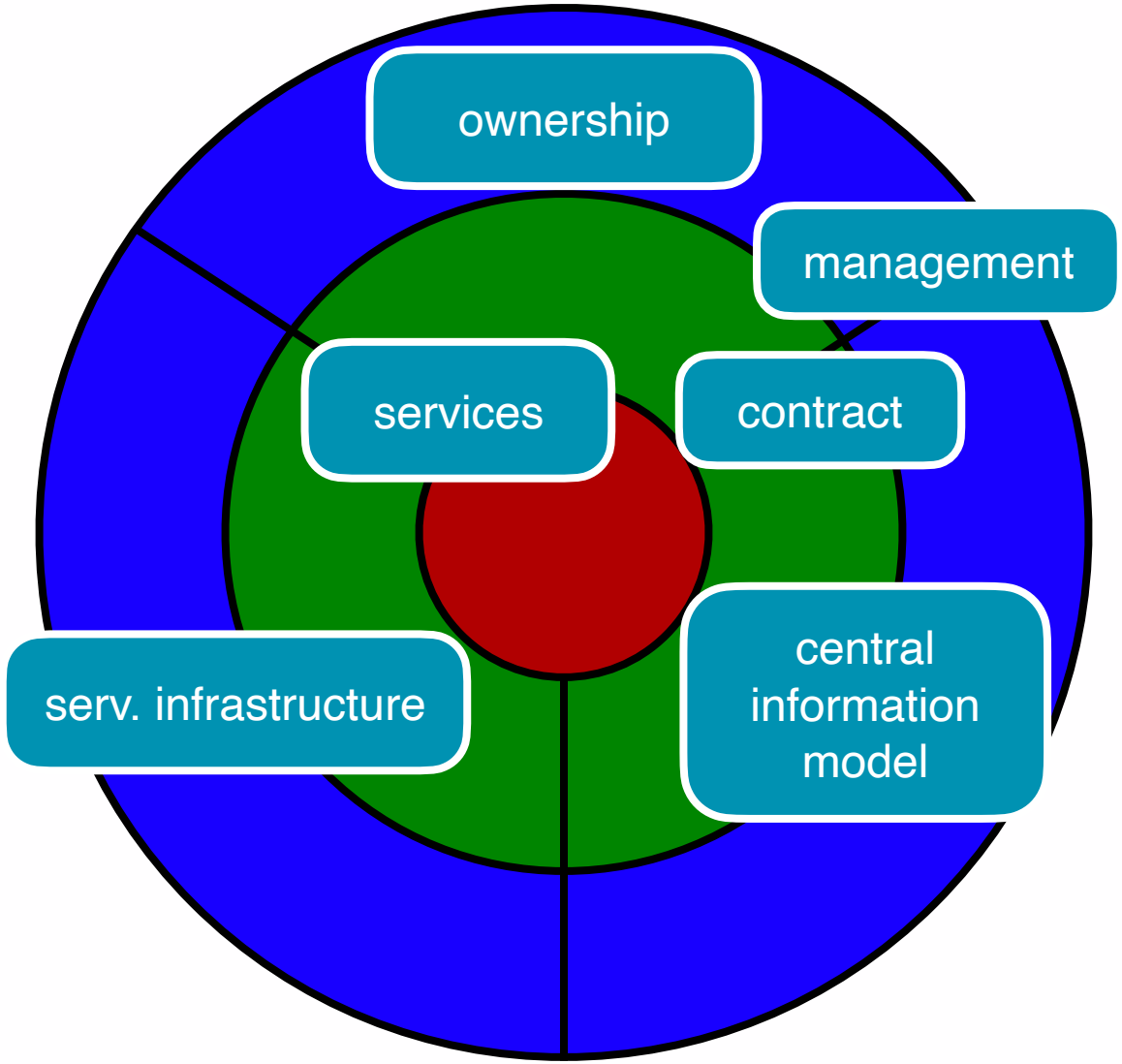
Service-oriented architecture







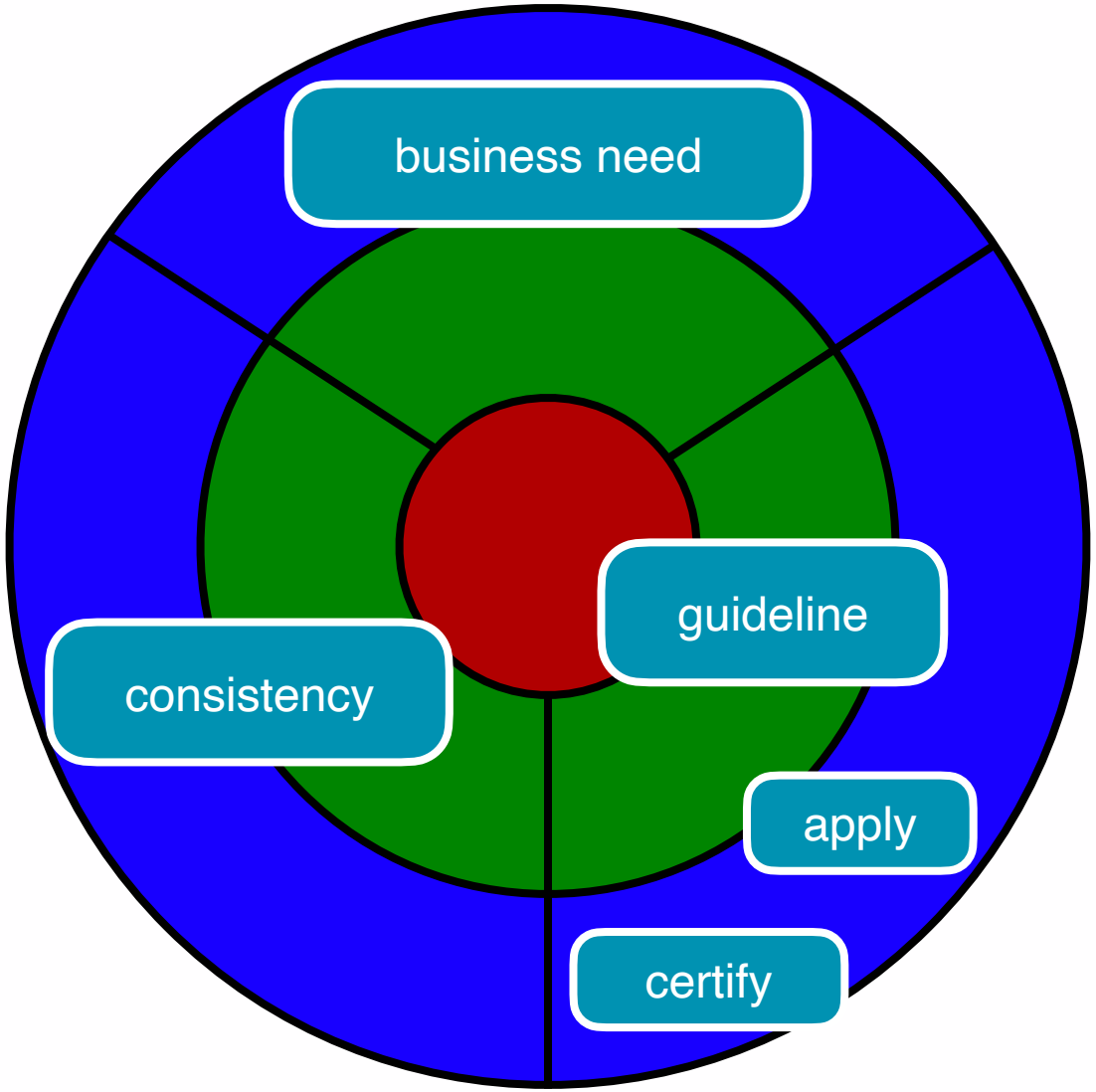
Service-oriented architecture



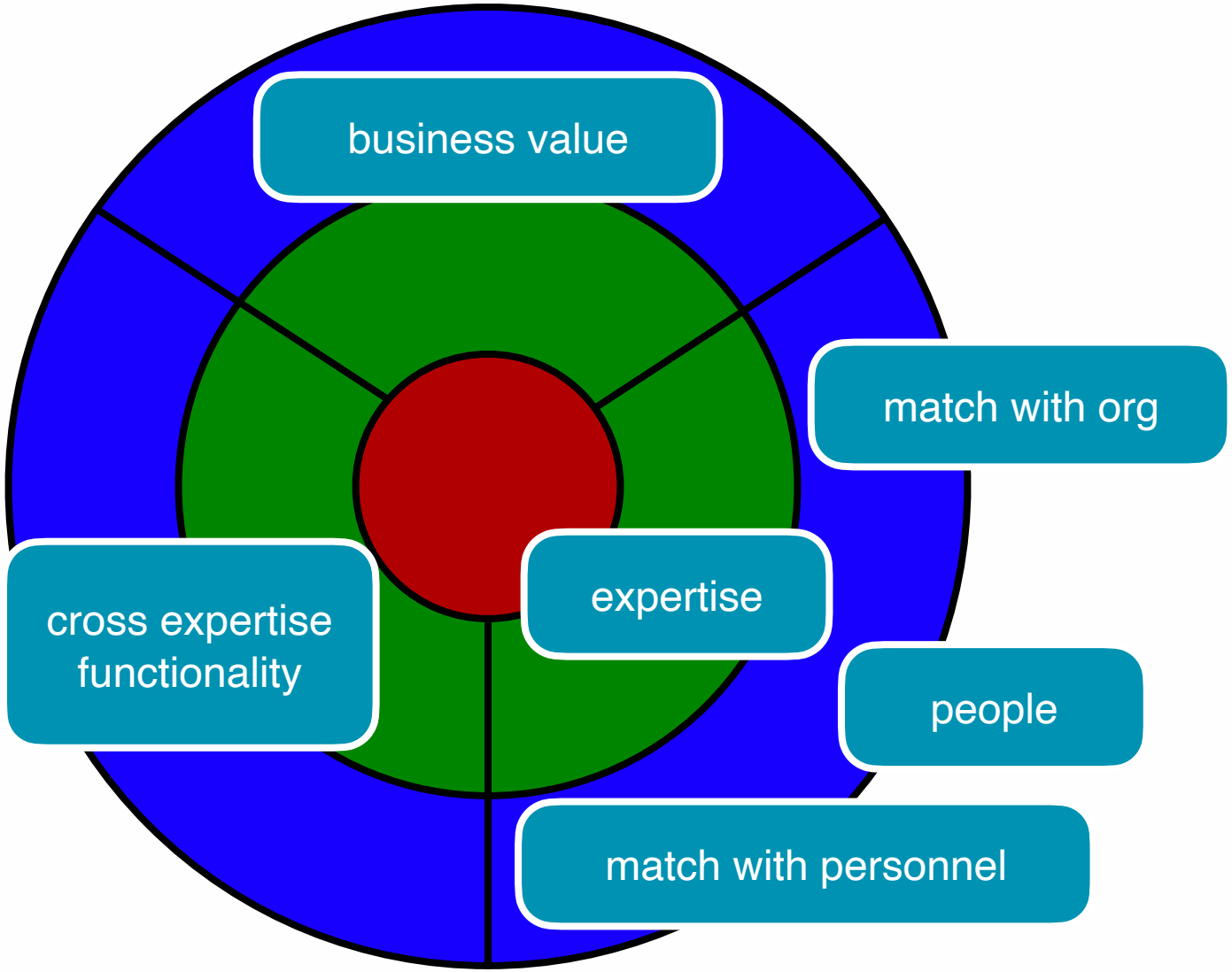
Guidelines/Principles

- https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/491871/EA-Principles-v2.1.pdf
- https://www.ibm.com/developerworks/rational/library/enterprise-architecture-financial-sector/index.html
- https://pdfs.semanticscholar.org/6a6c/29f9b443f704e825522c25581cbff48d162a.pdf
- https://ti.tuwien.ac.at/cps/people/obermaisser/papers/rr-32-2009.pdf
- https://www.iab.org/wp-content/IAB-uploads/2011/03/fred_carter.pdf
- https://scholar.google.nl/scholar?as_sdt=0%2C5&hl=nl&inst=4393003693960974403&q=intitle%3A%28architecture%20OR%20design%29%20intitle%3Aprinciples%20embedded%20system&start=20

Guidelines



Expertise-centric architecture



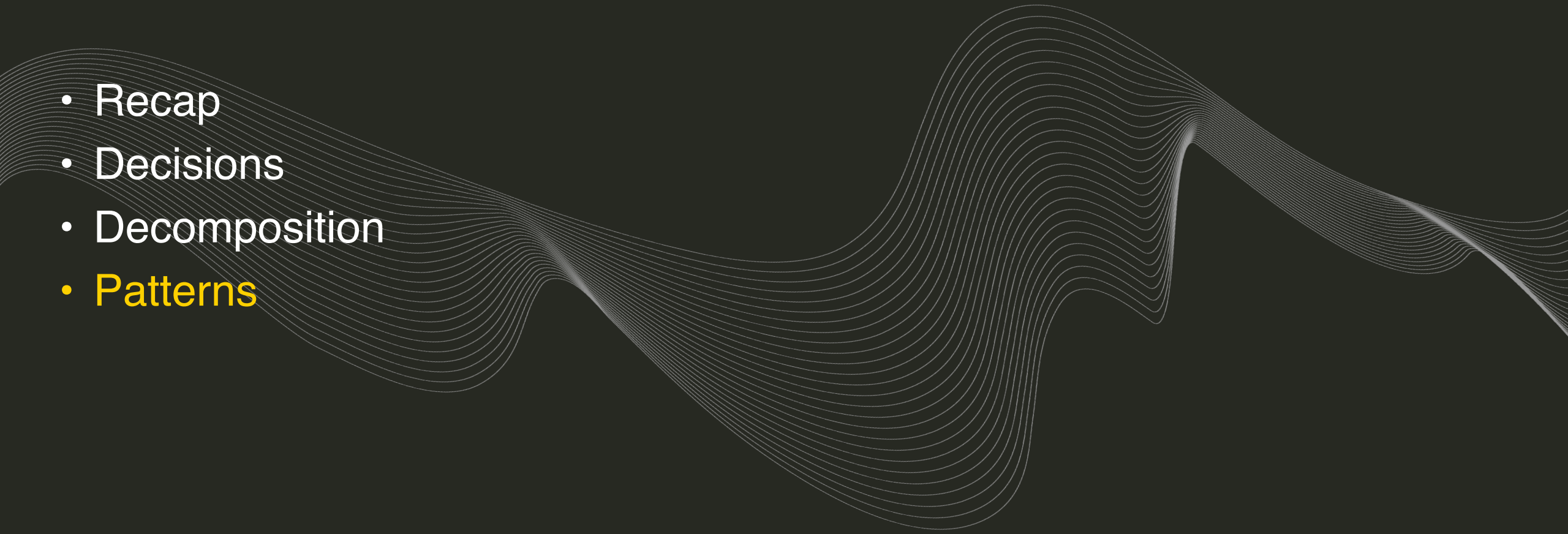
Any other architecture styles?

Dominant decomposition

- Choose one style — dominant
  - explain how it fits your domain!
- Choose another style — secondary
  - how is it related?
- How does it map
  - onto the structure of the dev org?
  - onto the tasks of team members?
- Which major stakeholder concerns:
  - will be served by your decomposition? (and why)
  - will be difficult to address well? (and why)

Lecture 7: Overview

- Recap
- Decisions
- Decomposition
- Patterns



Patterns

Patterns - general solution to general problem

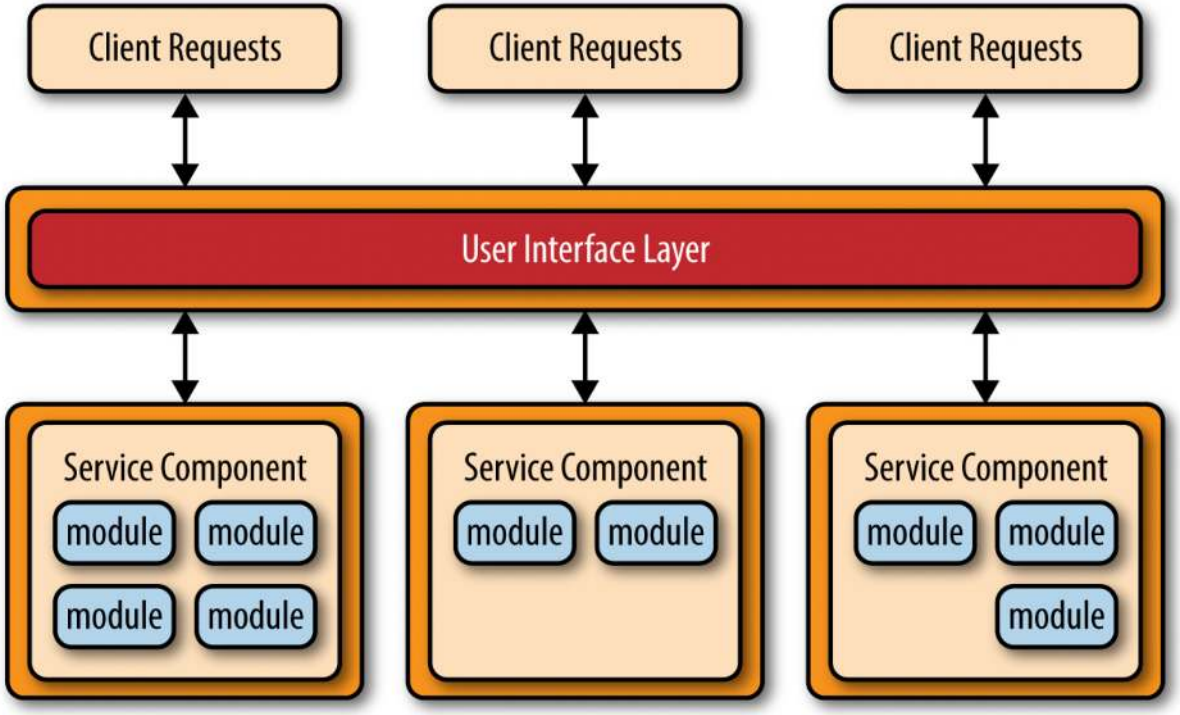
- **Design**
  - solutions to specific problems encountered during software development
  - guidelines for object relationships, responsibilities, and behaviours
  - Singleton, Factory, Observer, Decorator, Strategy, etc.
- **Architectural**
  - abstract solutions to problems related to the overall structure of a software system
  - address concerns such as system scalability, performance, maintainability, and security
  - Layered, Event-Driven, Microservices, Event-Bus, Pipe-Filter, Blackboard, MVC
  - Relation to: arch styles / decomposition?

Patterns - general solution to general problem

- **Design**
  - https://refactoring.guru/design-patterns
  - https://www.coengoedegebure.com/introduction-to-software-design-patterns/
  - For sleepless nights: https://www.youtube.com/watch?v=v9ejT8FO-7I
- **Architectural**
  - https://medium.com/@liams_o/fundamental-software-architectural-patterns-663440c5f9a5
  - https://www.youtube.com/watch?v=BrT3AO8bVQY
  - For sleepless nights: https://www.youtube.com/watch?v=SPS6P7orGSE

Microservices Architecture Pattern

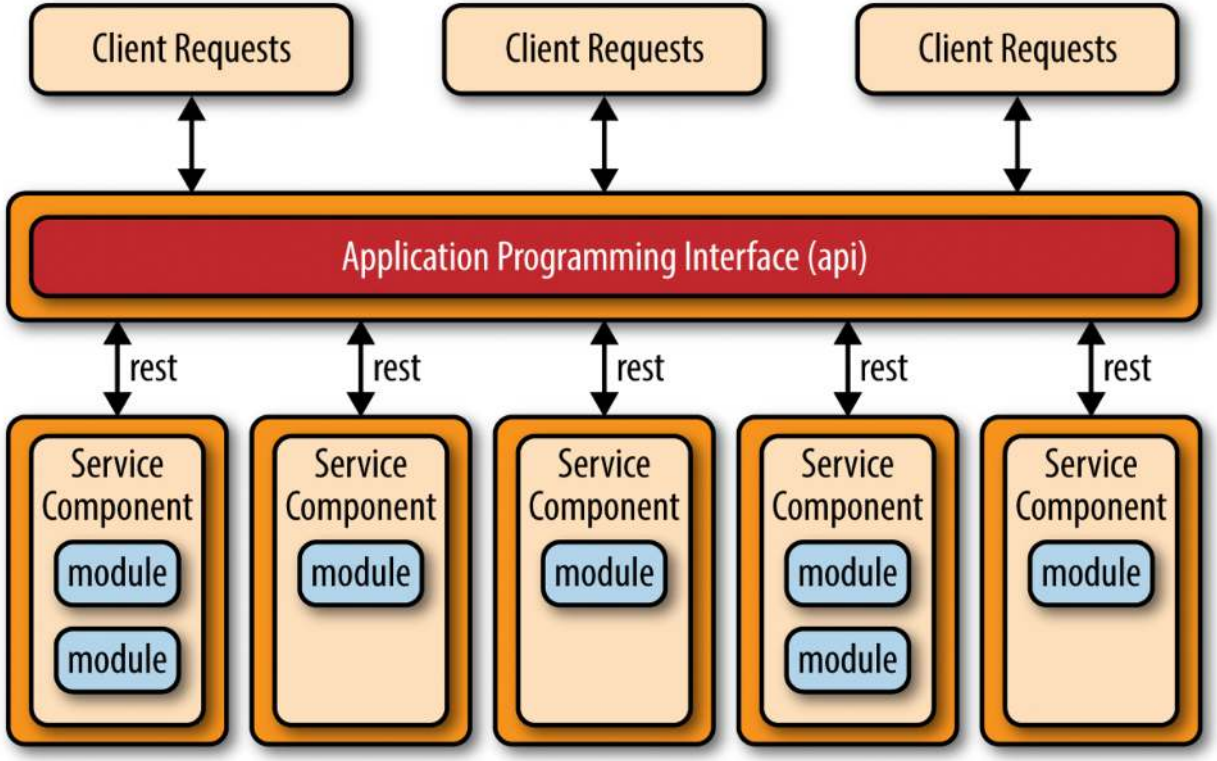
- Sources:
  - monolithic applications (layered)
  - SOA
- Eliminates orchestration needs, simplifies connectivity and access to service components
- Modules (e.g., Java classes):
  - single-purpose function
  - independent portion of a large business application
- Challenge: right level of service component granularity



Credits: O'REILLY https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch04.html#:~:text=While%20there%20are%20literally%20dozens,and%20the%20centralized%20messaging%20topology

Microservices - API-REST based

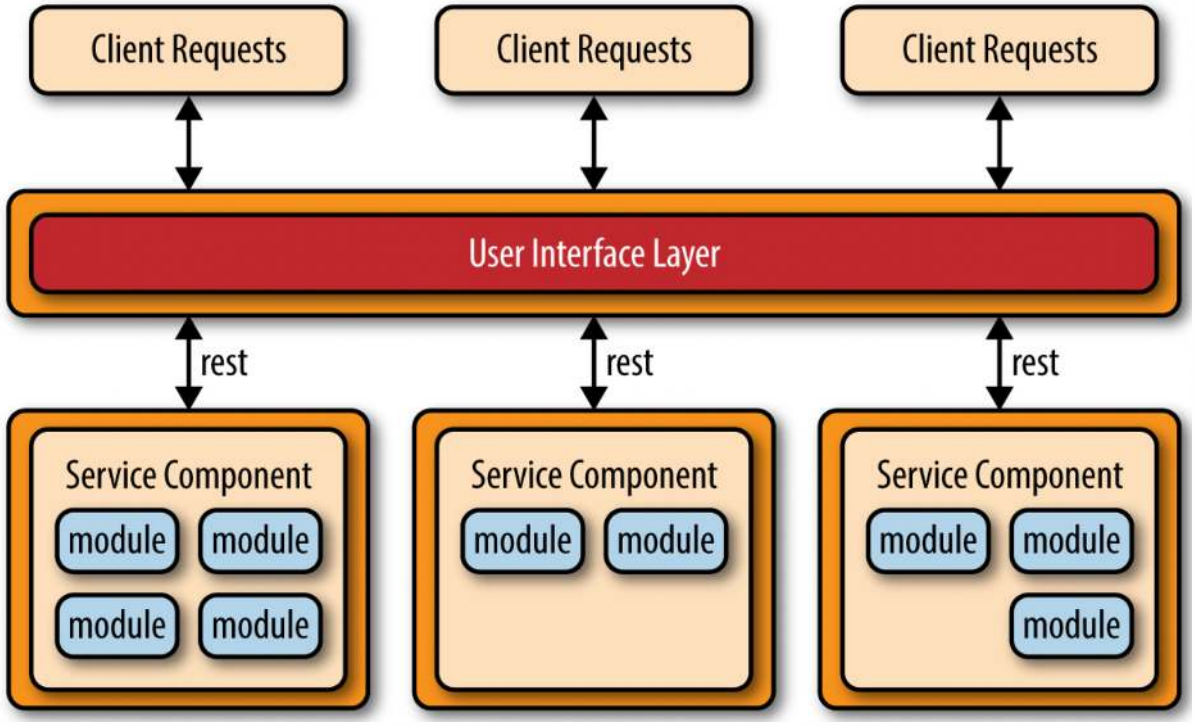
- Useful for websites that expose small, self-contained individual services through some API
- Single-purpose cloud-based RESTful web services found by Yahoo, Google, and Amazon
- Very fine grained service components



Credits: O'REILLY https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch04.html#:~:text=While%20there%20are%20literally%20dozens,and%20the%20centralized%20messaging%20topology

Microservices - REST based

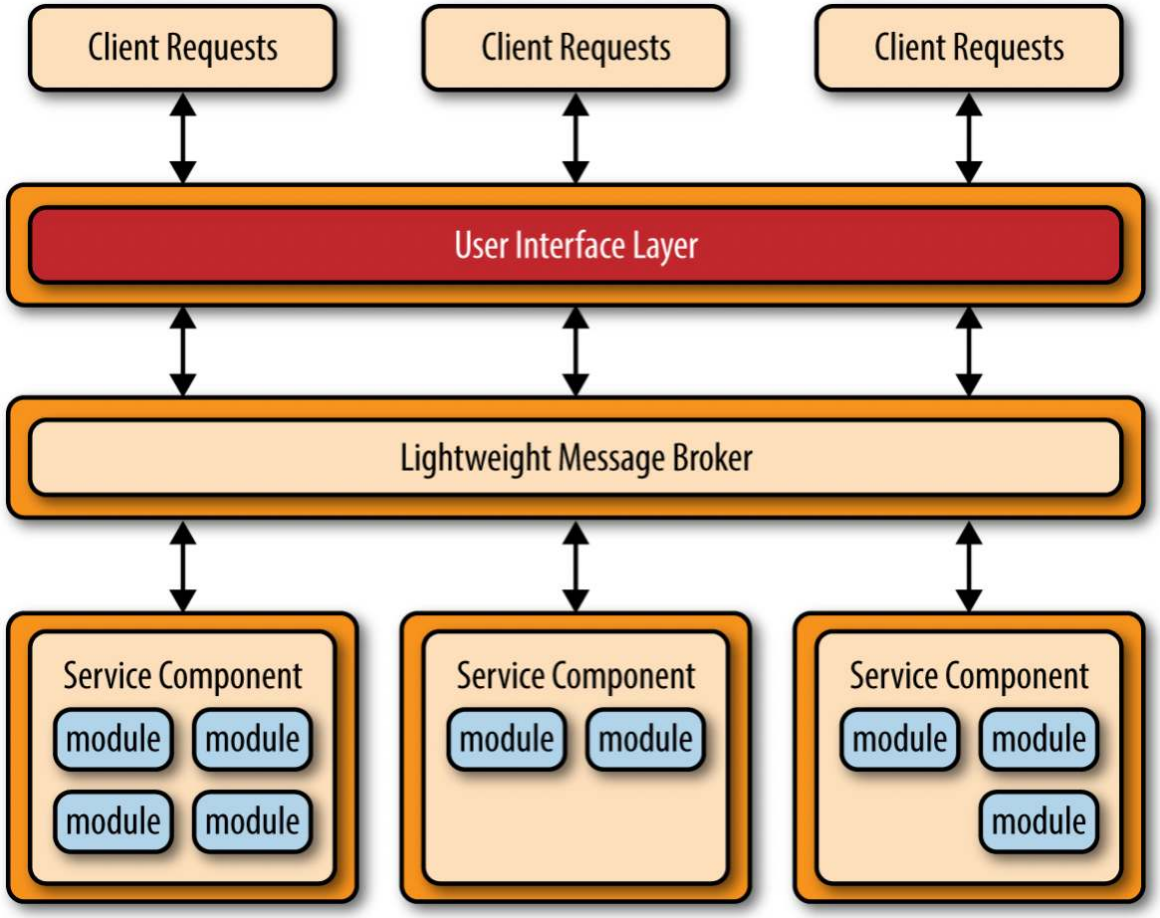
- Client requests are received through traditional web-based or fat-clients than through API
- Service components are larger, more coarse-grained, and represent a portion of the business application
- Small to medium-sized business applications with relatively low degree of complexity



Credits: O'REILLY https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch04.html#:~:text=While%20there%20are%20literally%20dozens,and%20the%20centralized%20messaging%20topology

Microservices - Centralized messaging

- Instead of using REST, it uses a lightweight centralized message broker
  - does not perform orchestration, transformation, or complex routing (=/= SOA)
  - lightweight transport
- Found in larger business applications / requiring more sophisticated control over the transport layer between the user interface and the service components
- Advanced queuing mechanisms, asynchronous messaging, monitoring, error handling, load balancing and scalability



Credits: O'REILLY https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch04.html#:~:text=While%20there%20are%20literally%20dozens,and%20the%20centralized%20messaging%20topology

Patterns - general solution to general problem

- Read/watch some intro
  - Choose patterns from a source (given by me or found by yourself)
  - Apply them to your system
  - Relation to your dominant decompositions?
  - Justify your choice (by relating them to the stakeholder concerns)
- Be aware: a smart way to divide the work differs a lot
  - Think from the perspective from a concern and think how you will address the concern in the design.

Homework

Who decides what and when

- Who are the deciding actors?
- What do they decide about?
- When are those decisions on the
- What happens when you swap a

Dominant decomposition

- Choose one style — dominant
  - explain how it fits your domain!
- Choose another style — secondary
  - how is it related?

Patterns - general solution to general problem

- Read/watch some intro
  - Choose patterns from a source (given by me or found by yourself)
  - Apply them to your system
  - Relation to your dominant decompositions?
  - Justify your choice (by relating them to the stakeholder concerns)
- Be aware: a smart way to divide the work differs a lot
  - Think from the perspective from a concern and think how you will address the concern in the design.

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