Architecting digital repeatable systems for systemic Digital Transformation

Module 1-4

Better Architecting With (BAW)

Dr Alexander Samarin

SAMARIN.BIZ

WHY a common approach case: India 100 Smart Cities project

- Cities, as places where people live and work, are very important for our civilisation.
- All cities are different.
- Smart City is a city which makes the world better for citizen, society, business and government.
- In Smart Cities projects 50% – 70% of work is about IT, i.e. digital.
- Smart City is a city which is rebuilt as a digital system (DT is critical).
- Digital systems can be done well, be adaptable and easy repeated (his is the main secret of DT), but such emergent characteristics have to be purposefully created.
- If a common methodology for building Smart Cities as digital repeatable systems is used, then all Smart Cities are the 70% same on (an estimation).
- Coordinated and complementary building together Smart Cities as digital repeatable systems leads to substantial gains in cost, speed and quality of their implementations.
- An export version of the Smart Cities implementation is created automatically.
- The same can be done for healthcare, buildings, homes, industries, territories and governmental agencies.

Extra voice for agile (1)

SO WHICH ARE THE **PROBLEMS** – HERE SOME

- We are at a **methods war** for 50 years
- Practices are locked in **method prisons**
- Method prisons are controlled by **method gurus**
- All methods are **monolithic**
- Every method's description is **homegrown**
- Methods have **no common ground**

This is immature and foolish!!!

IVAR JACOBSON INTERNATIONAL

Creating winning teams.

Extra voice for agile (2)

WE ARE AT A METHODS WAR FOR 50 YEARS

Some root causes:

- Method gurus "borrow / steal" practices from one another.
- They rewrite the "borrowed / stolen" practices to fit within their method.
- In the process they "improve / misunderstand" the original idea.
- They don't collaborate for obvious reasons.
- They grow "fan clubs" (sects) with fanatic attitudes

This is not because method gurus are "bad', but because the arena the play at is "bad".

So the Winner is the best Marketeer,
not necessarily the best Method

IVAR JACOBSON INTERNATIONAL

Creating winning teams.

Extra voice for agile (3)

PRACTICES ARE LOCKED IN METHOD PRISONS

Today

- There are 100,000+ methods in the world
- Every method is a composition of some smaller mini-methods, called practices
- There are only a few 100s of practices
- Like the ingredients in a soup the practices are not separable and not reusable – *they are stuck in the method*

Practices can't easily be reused to create other methods

Extra voice for agile (4)

METHOD PRISONS ARE CONTROLLED BY METHOD WARDENS - GURUS

Today, the guru controls which practices goes into his/her method

- No one is an expert on everything (in a good method)
- But you can be an expert on something (such as a practice)

I was myself one of the method gurus controlling RUP – a weird role

- We need practice experts, **not method gurus**
- We still need methods but **no branding**.
- We still need **innovators** at all levels

IVAR JACOBSON INTERNATIONAL

Creating winning teams.

Extra voice for agile (5)

ALL METHODS ARE MONOLITHIC

- They are non-modular, such that
- You can't mix and match practices from different methods

Methods are like isolated islands – no bridges

IVAR JACOBSON INTERNATIONAL

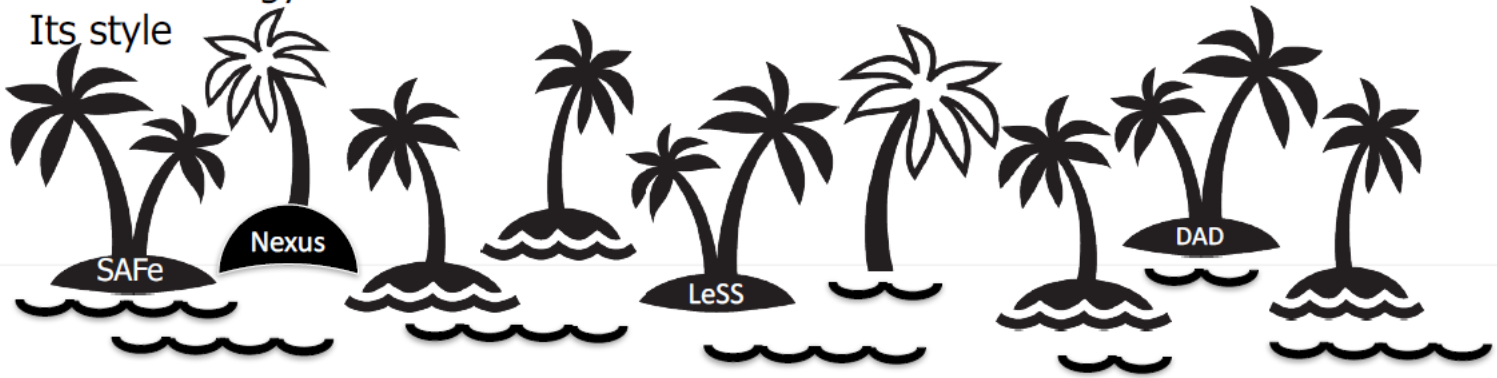
Creating winning teams.

Extra voice for agile (6)

EVERY METHOD's DESCRIPTION IS HOMEGROWN

Everything about the method is unique

- Its user experience
- Its structure
- Its terminology
- Its style



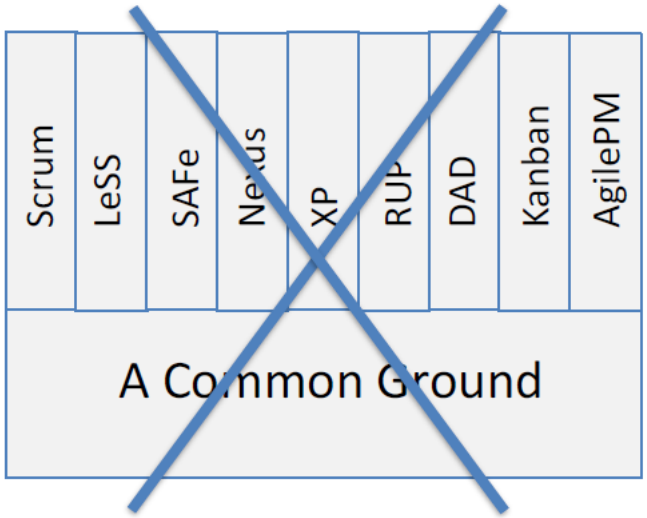
Comparing/mixing methods is like comparing/mixing cultures

IVAR JACOBSON INTERNATIONAL

Creating winning teams.

Extra voice for agile (7)

METHODS HAVE NO COMMON GROUND



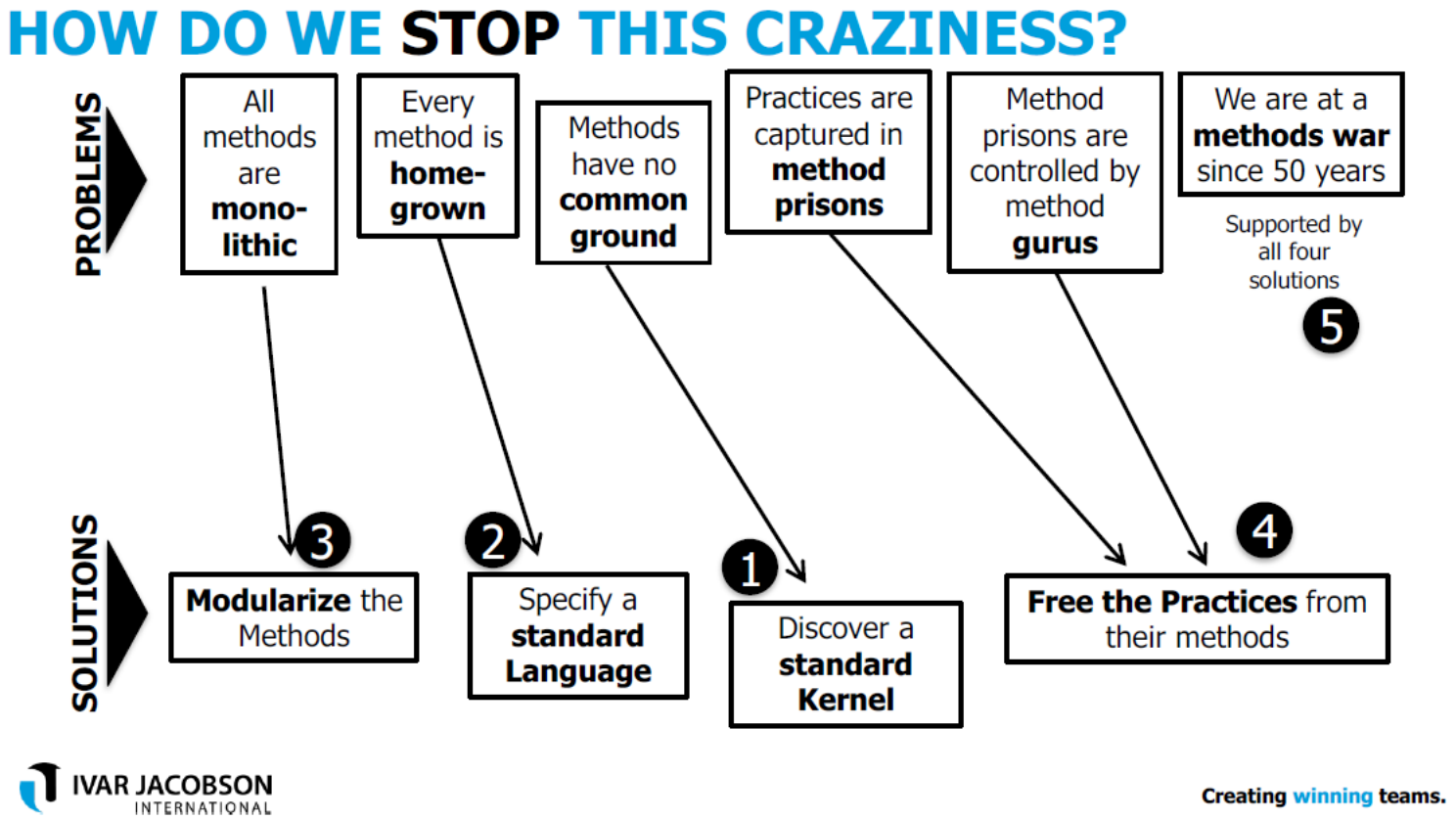
- They all deal with software so they should share a lot
- Fact is, they share almost nothing, not even the basics:
  - What is Software?
  - What is Software development?
  - What is Requirements, Design, Test?
  - What is Team, Way of Working?

Essence is a Common Ground for all methods/frameworks

IVAR JACOBSON INTERNATIONAL

Creating winning teams.

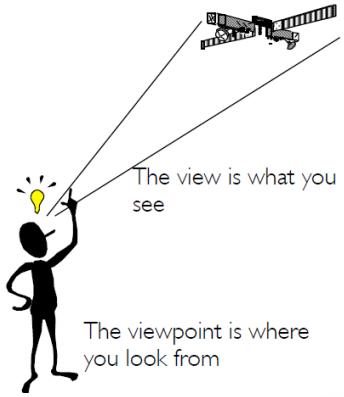
Extra voice for agile (8)



Essential requirements

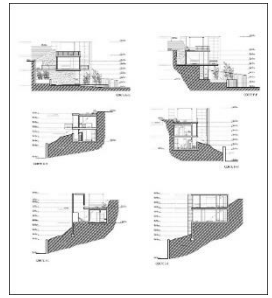
- The whole life cycle of a system to be build
  - conception, development, production, utilization, support, retirement and destruction
- Address need of all stakeholders
  - some non-IT model-types
- Methods to be applied by Subject Matter Experts (SMEs) not specially trained IT staff members
- Maintaining integrity of all models
- Different people in similar situation should find similar solutions/services/components or bring innovations

ISO/IEC/IEEE 42010:2011 architecture description



Views (system-in-focus dependent) are governed by **viewpoints** (system-in-focus independent)

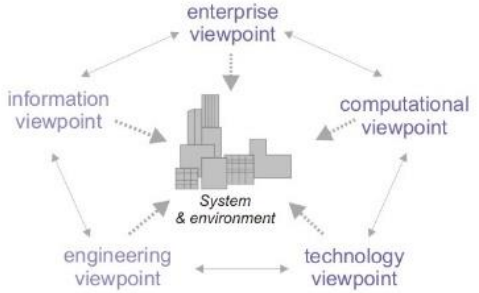
Geometrical views of buildings are viewed side by side — as a **composition**



Architecture views are often originated by different people — thus they **must be aligned** to be used together

Each view comprises one or many **models**. Any model consists of **artefacts** (e.g. applications, servers, products, reports, etc.) and relationships between them.

Models (system-in-focus dependent) are governed by **model-kinds** (system-in-focus independent).

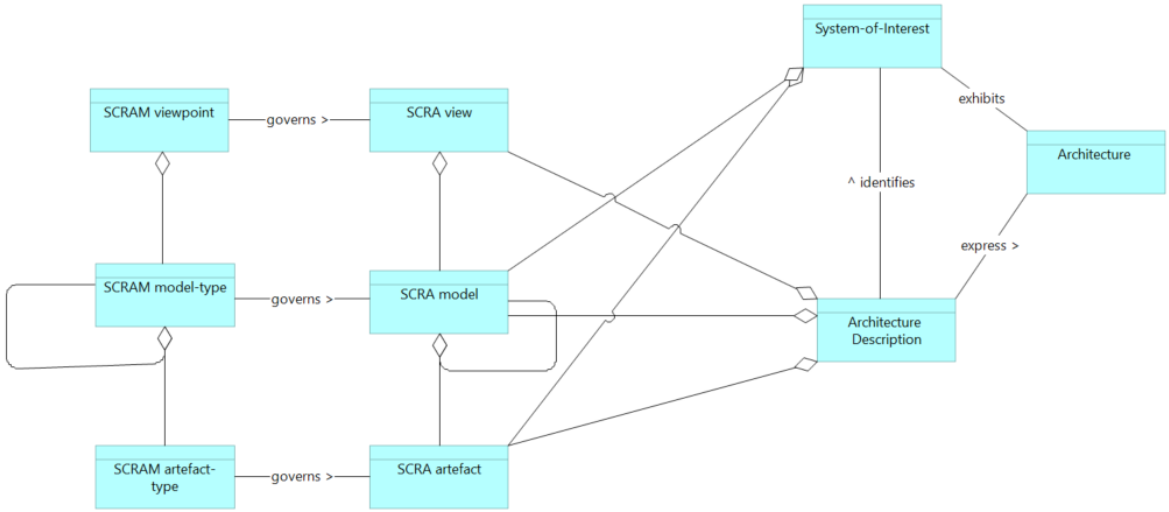


SCRAM: Collection of viewpoints, model-types, artefacts-types and patterns

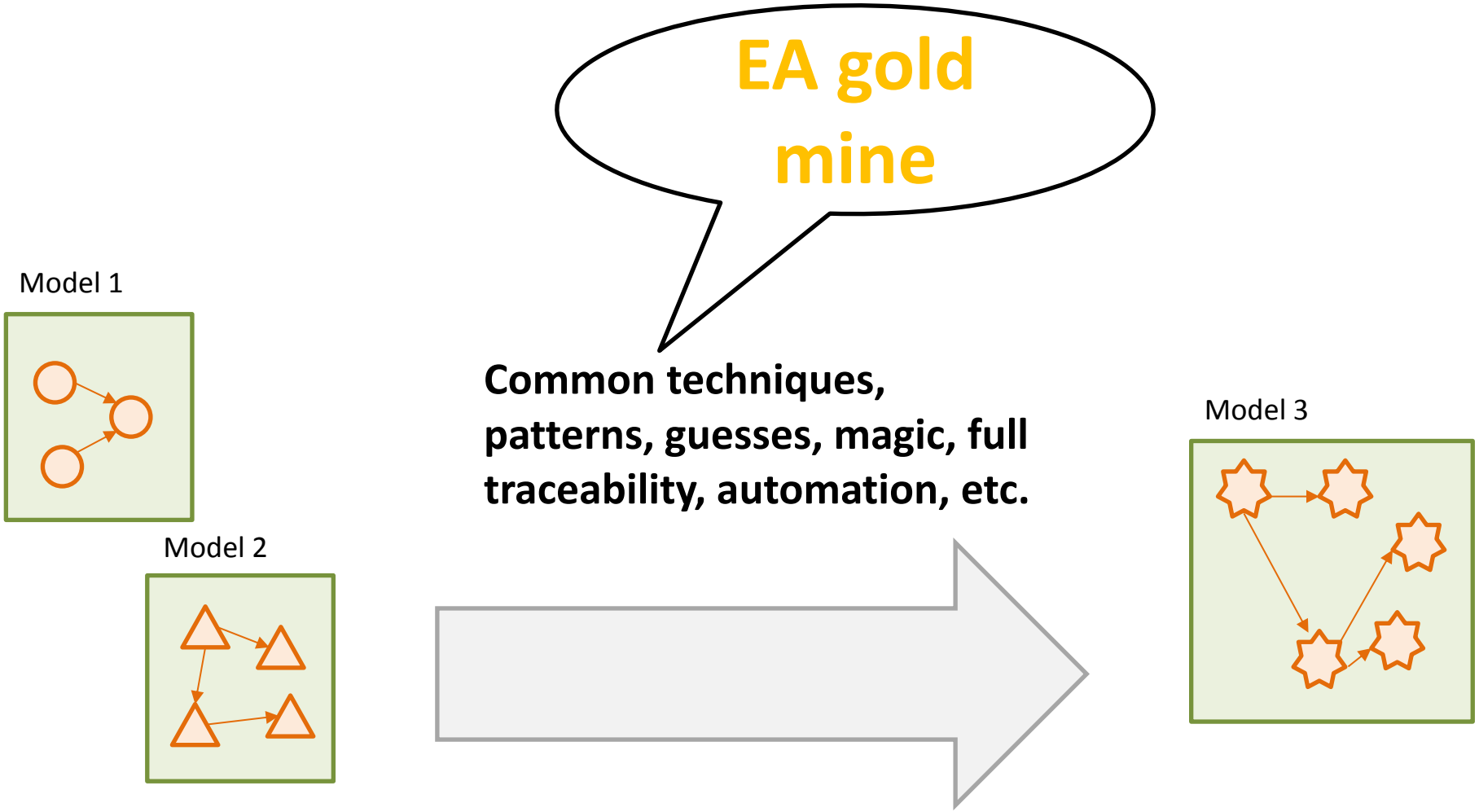
- We created the SCRAM the following way
  - Decomposed many "monolith" frameworks into smaller pieces
  - Sorted those pieces out
  - Structured those pieces
- **SCRAM viewpoints** collect one or many SCRAM model-types
- **SCRAM model-types** link one or many SCRAM model-types and/or **SCRAM artefact-types**
- **SCRAM patterns** are methods to create an SCRAM model-type from other SCRAM model-types
- If possible **models are digital**, i.e. formal, explicit, machine-readable and machine-executable

Core concepts SCRAM and SCRA

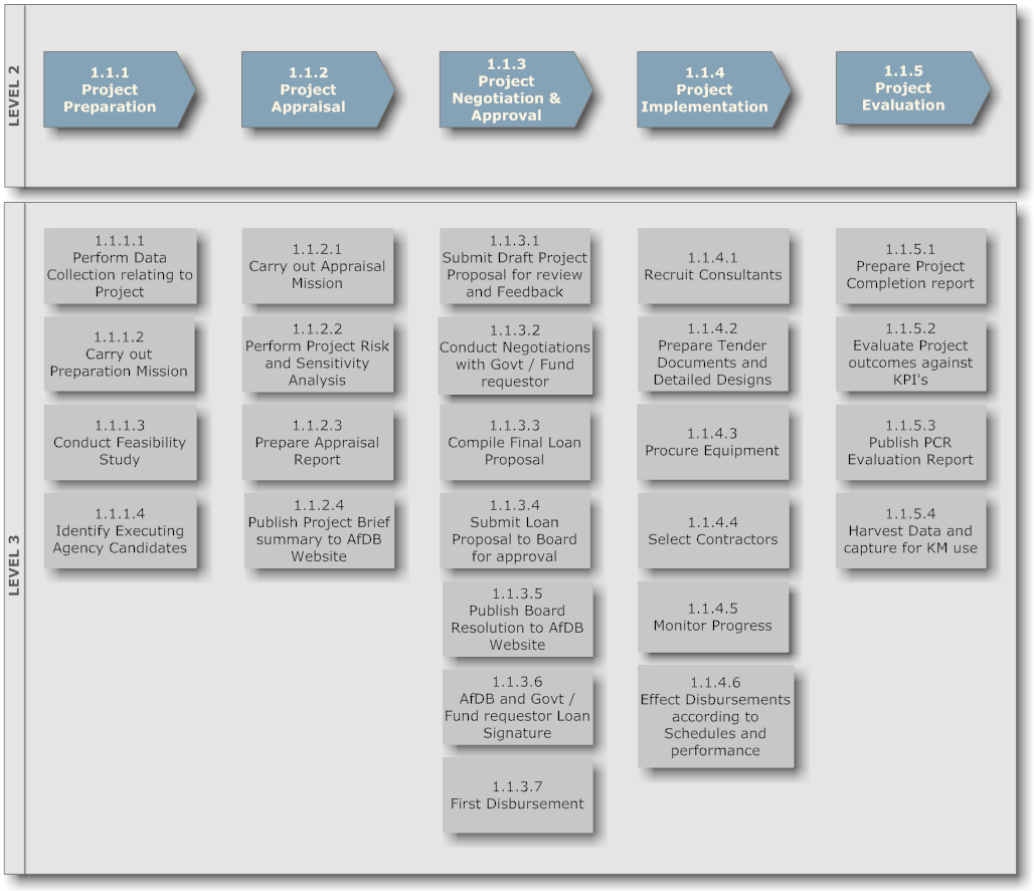
- SCRA **view**
- SCRA **model**
- SCRA **artefact**
- SCRAM **viewpoint**
- SCRAM **model-type**
- SCRAM **artefact-type**

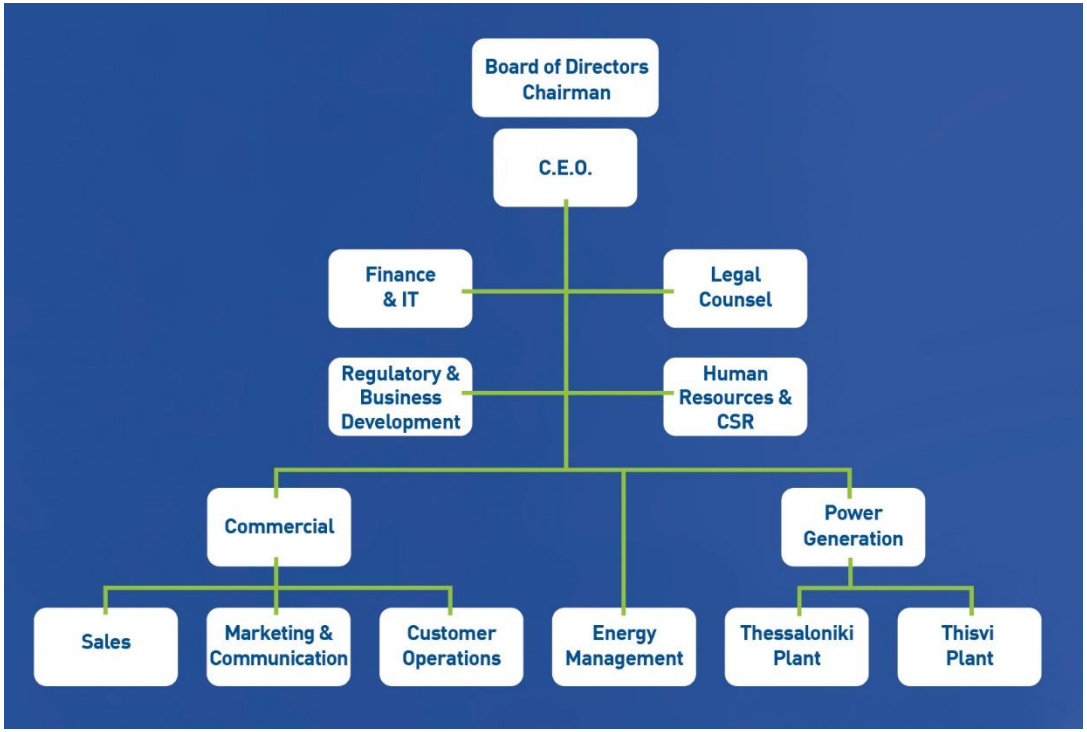


Some models may be generated from others



Example: from functions to org. units (1)





Pattern Structure IT Organisation (SITO)

- Business concern: How to structure a business unit
- Logic
  - Collect functions
  - Draw a matrix of mutual relationships between those functions
  - The relationships may be like “synergy”
  - The relationship may be like “prohibition”, e.g. SoD
  - Find clusters in the matrix

Function	F1	F2	F3	…
F1		+ (similar)	=	
F2	+ (similar)		- (SoD)	
F3	=	- (SoD)		
…				

Example: from functions to org. units (2)

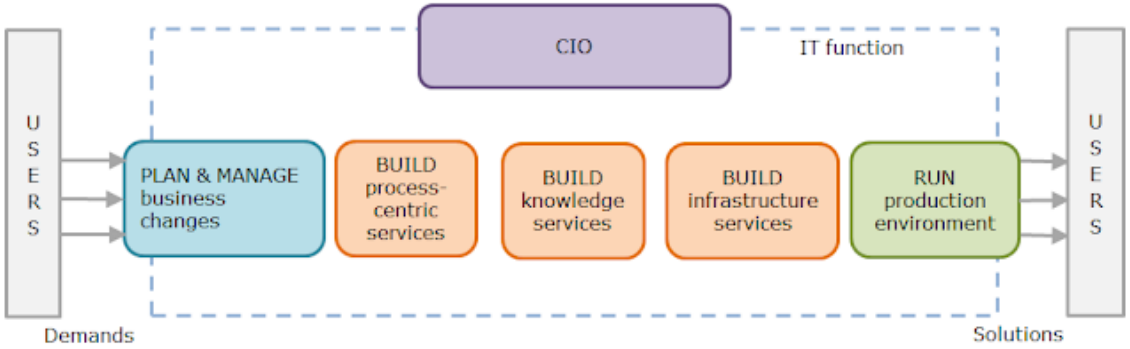
- Prohibition rules
  - P1 Separate doing and supervising/controlling – SoD
  - P2 Separate architecture/design and implementation – SoD, specialisation and quality at entry
  - P3 Separate implementation and operation – SoD, specialisation and quality at entry
  - P4 Policy vs applying it – legislation vs executive separation
  - P5 Specialisation
- Synergy rules
  - S1 Close work
  - S2 Architecture role to guide
  - S3 Synergy between technical and administrative activities (**how** you do something may be more important **what** you do)

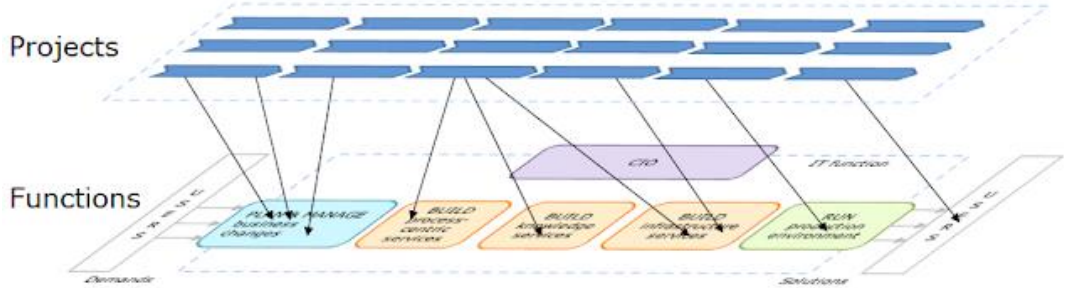
Example: from functions to org. units (3)

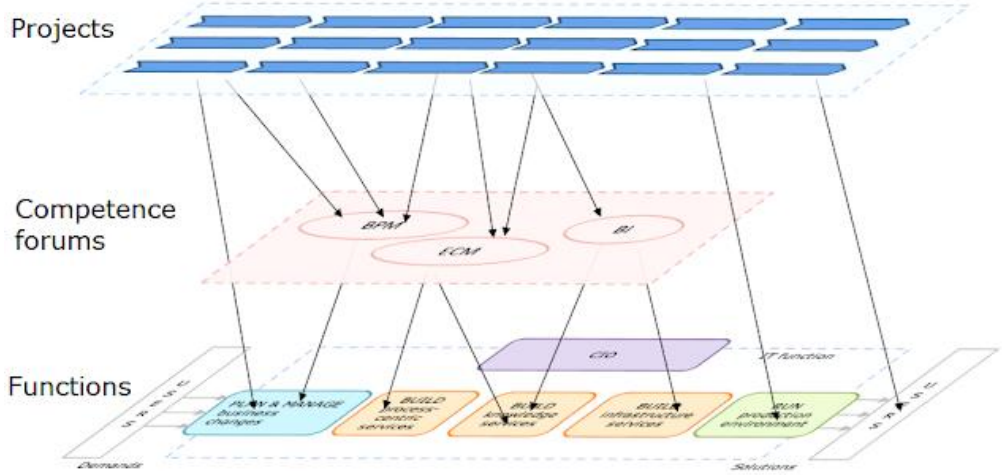
	GOVERN	ARCH	SAFE	PM	OM	BUILD	OPER	EVAL	INTERN
GOVERN		=	=	S1	S1	P1	P1	P1	=
ARCH			=	S2	S2	P2	P1	P1	=
SAFE				P4	P4	P4	P4	P1	=
PM					=	P1	P3	P1	=
OM						P3	P1	P1	=
BUILD							P3	P1	=
OPER								P1	=
EVAL									=
INTERN									

Functions (row) vs divisions (col)	PLAN and MANAGE business changes	BUILD process-centric services	BUILD knowledge services	BUILD infrastructure services	RUN production environment	INTERNAL services (CIO office)
GOVERN						
ARCH						
SAFE						
PM						
OM						
BUILD						
OPER						
EVAL						
INTERN						

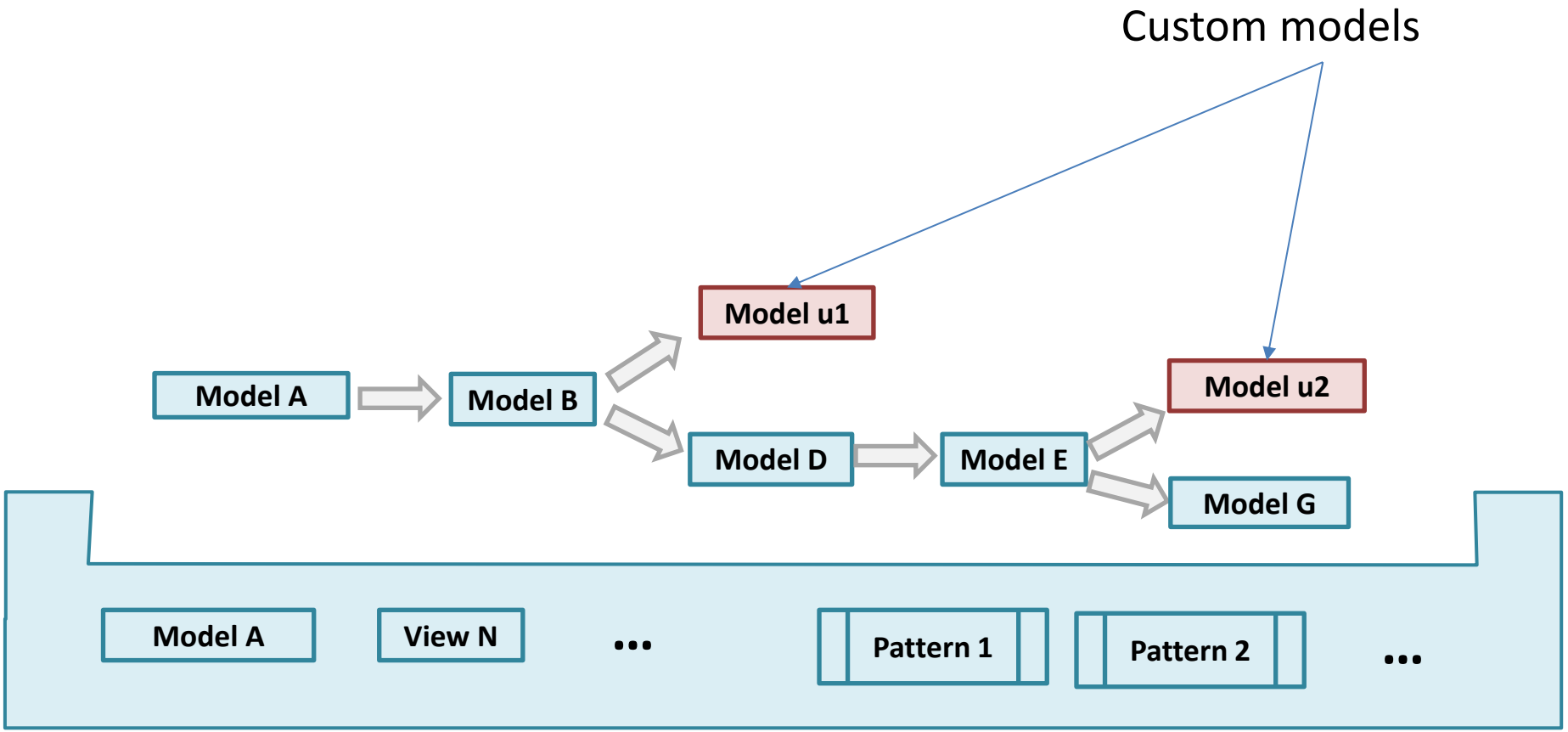
Example: from functions to org. units (4)





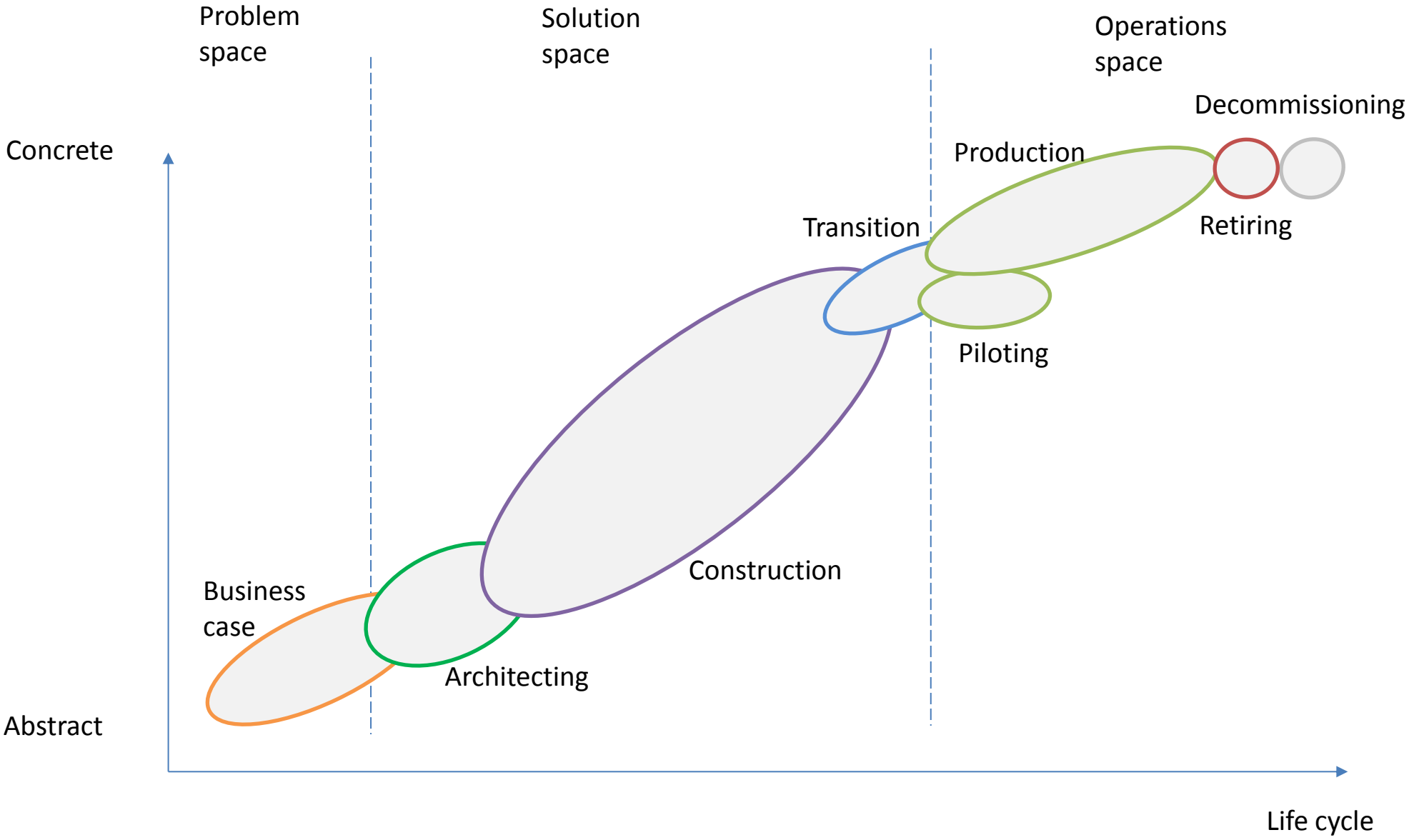


Natural dependencies of models

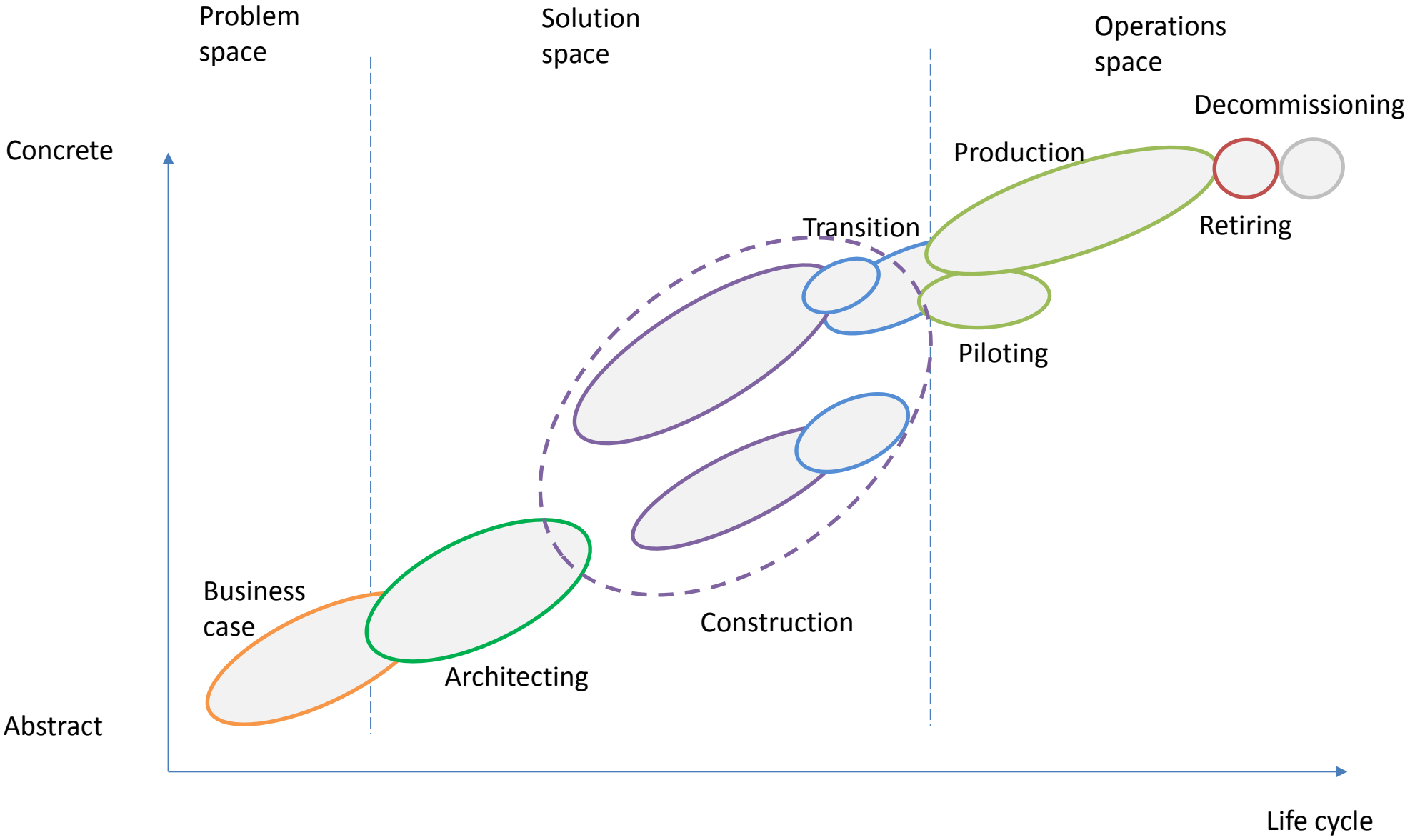


A model can be in one or many views

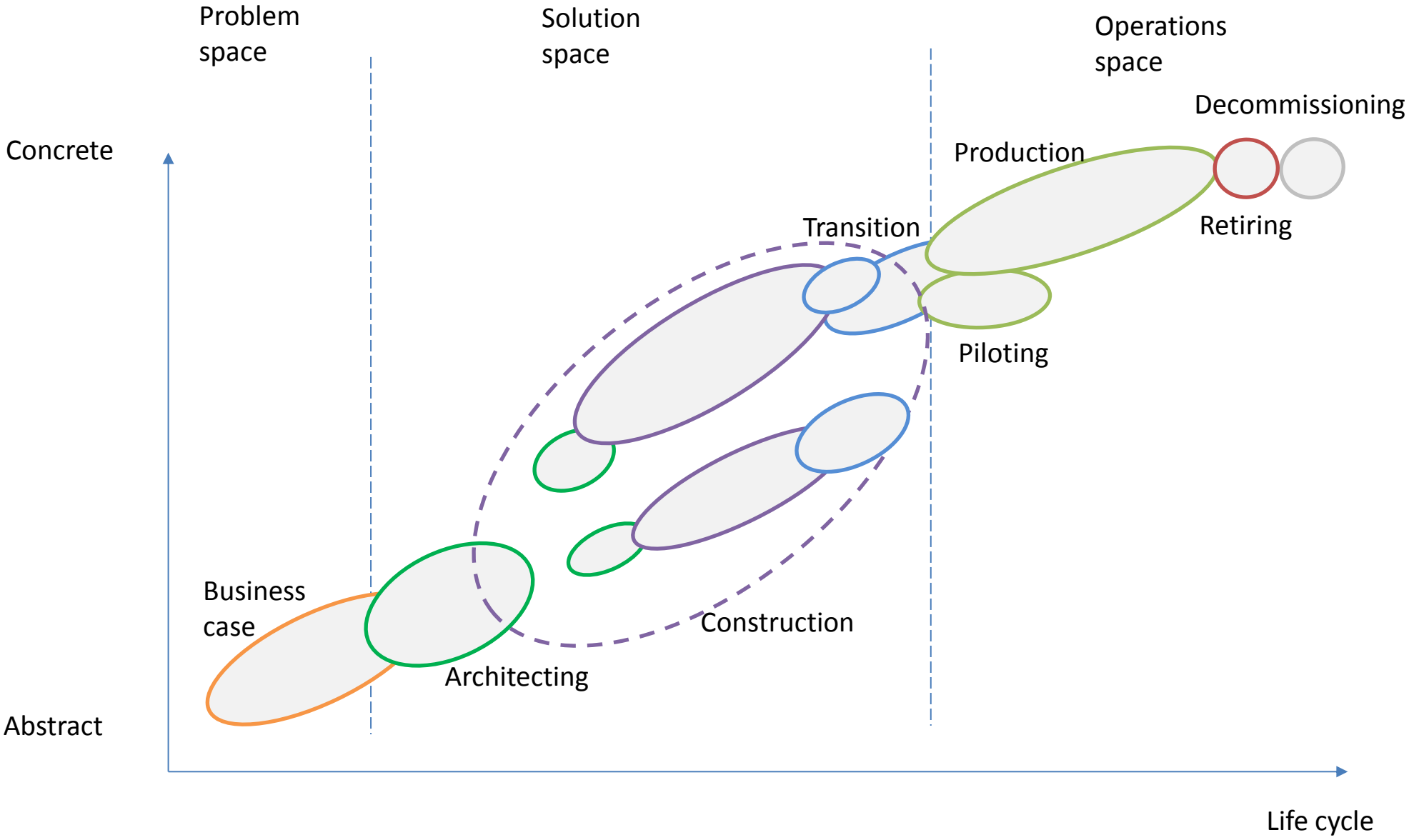
A common understanding of the LC (1)



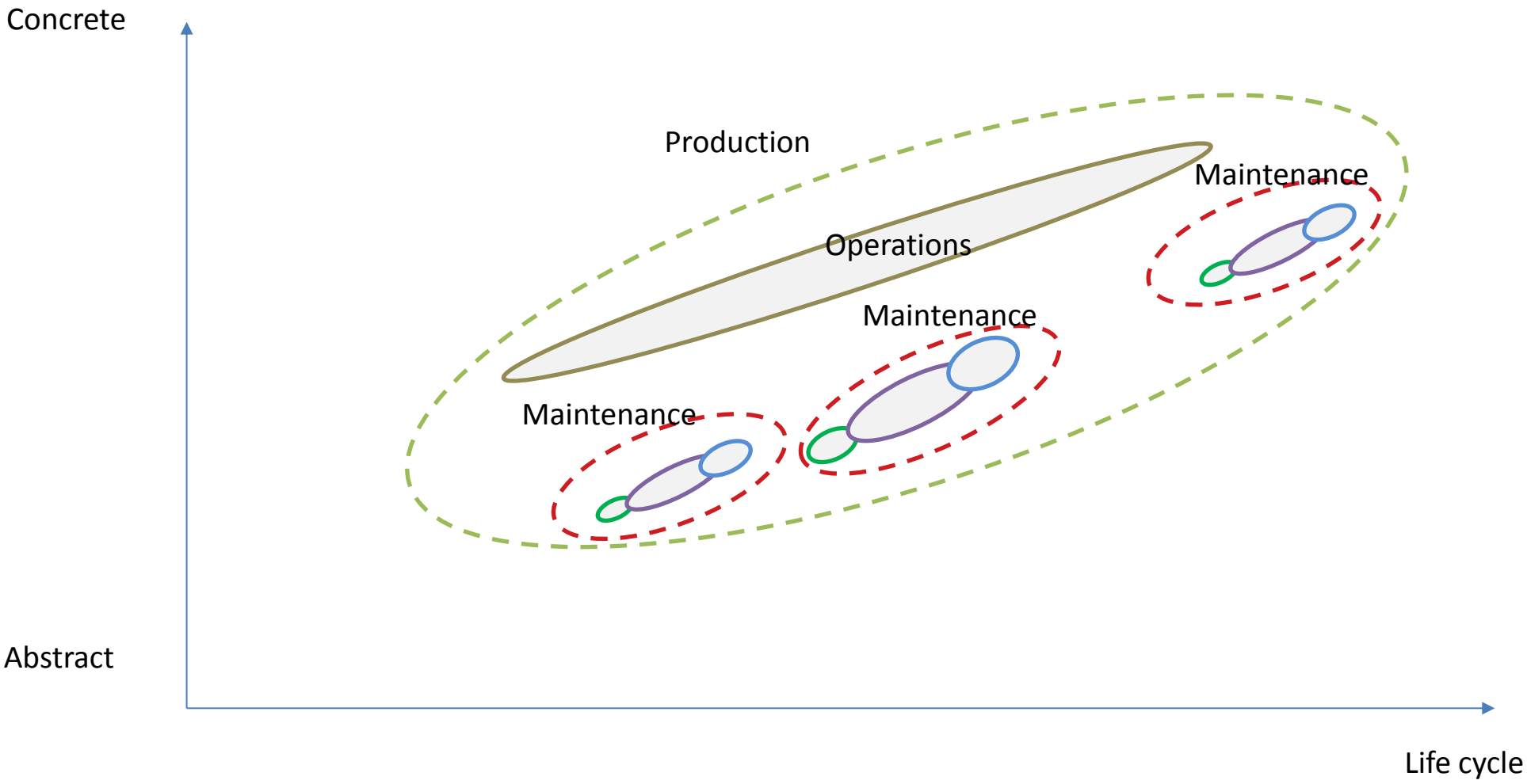
A common understanding of the LC (2)



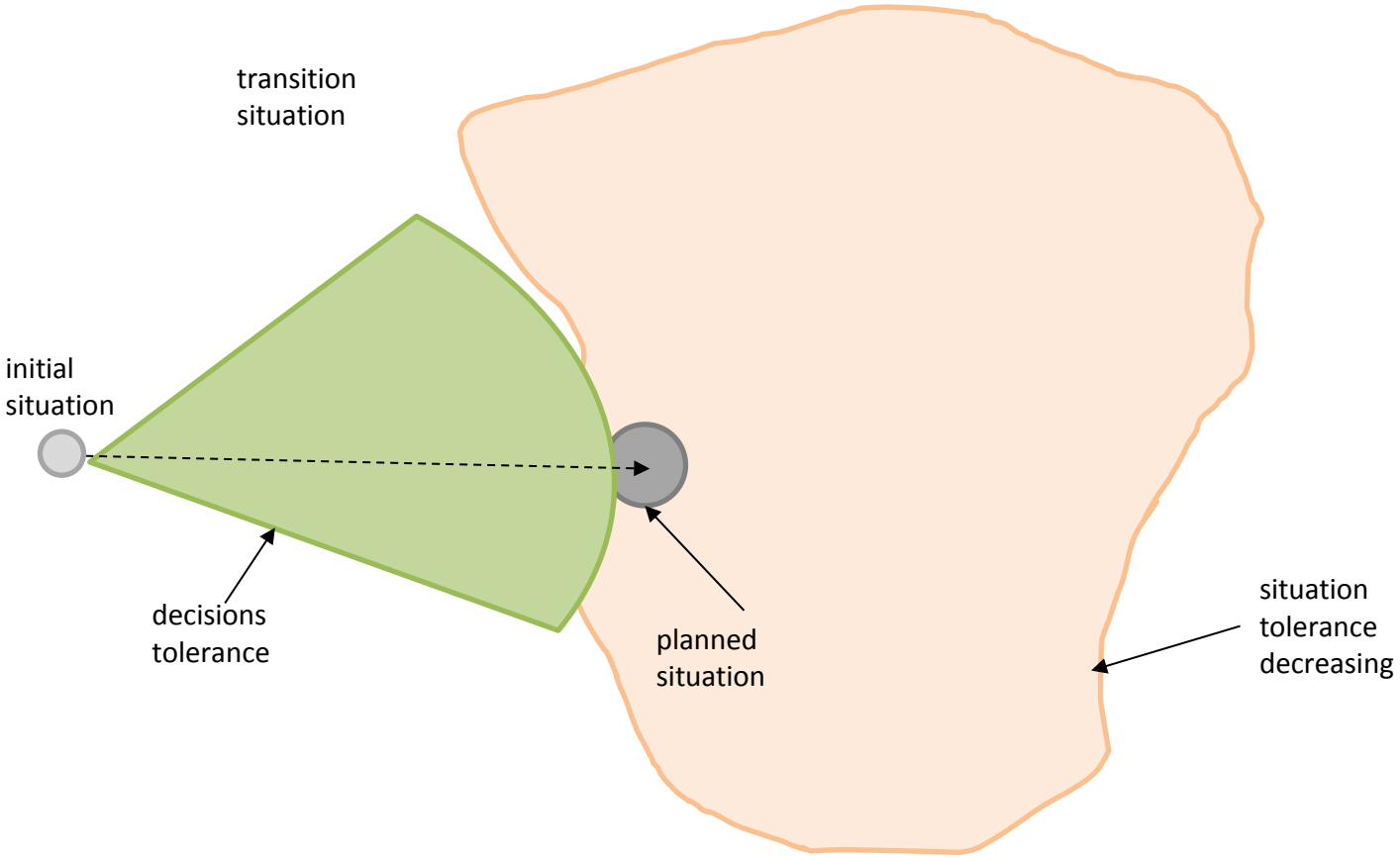
A common understanding of the LC (3)



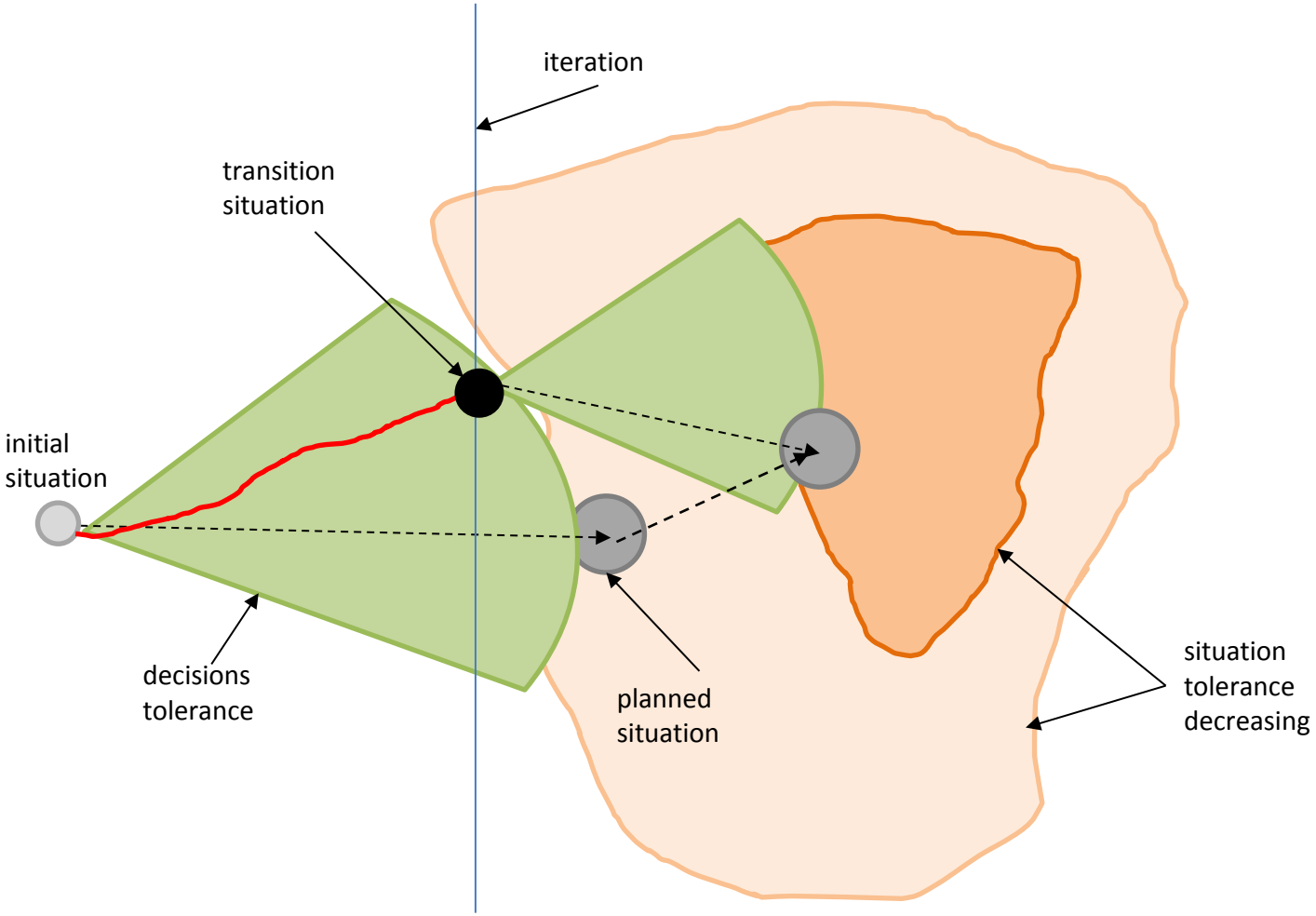
A common understanding of the LC (4)



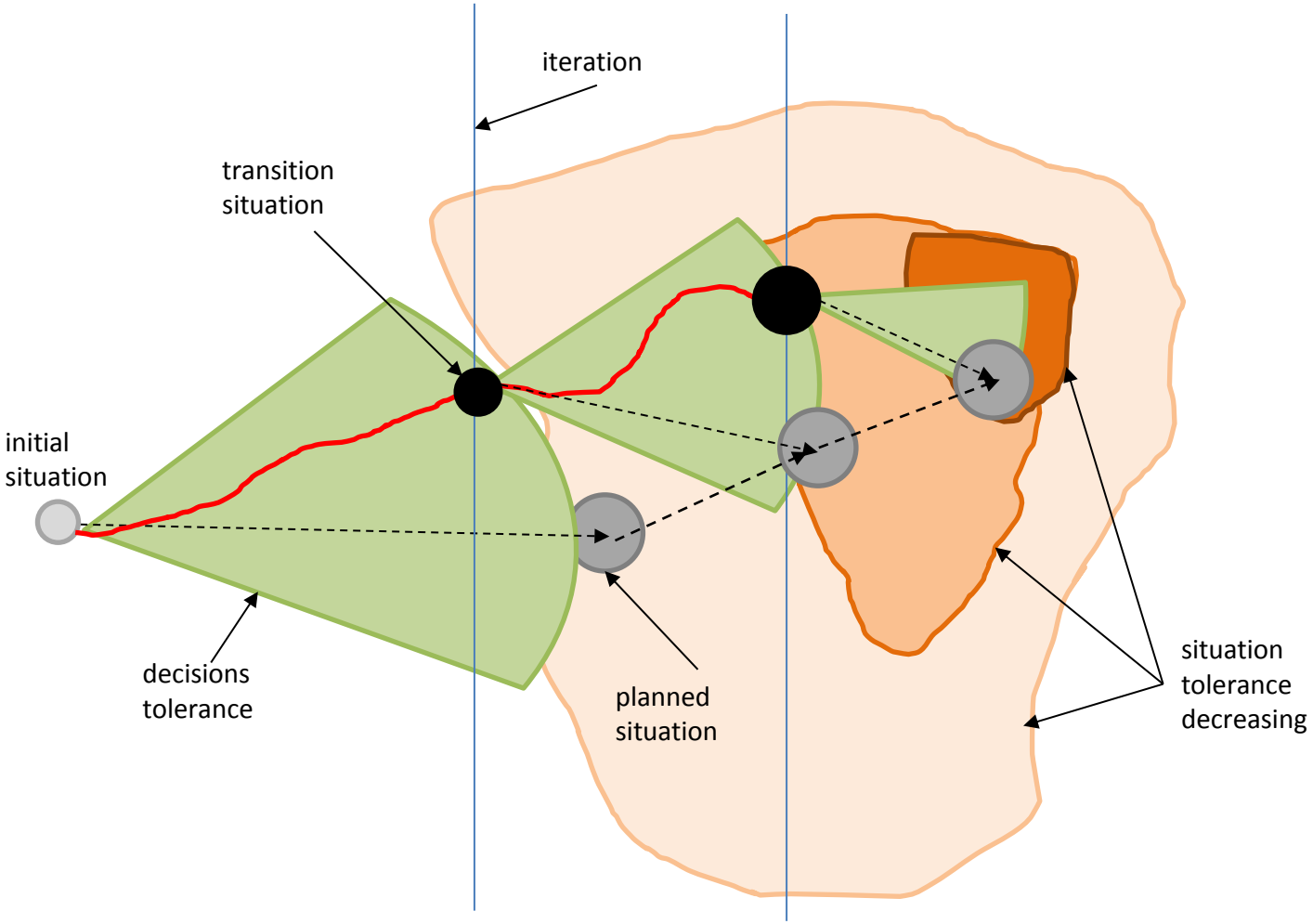
Iterative approach (1)



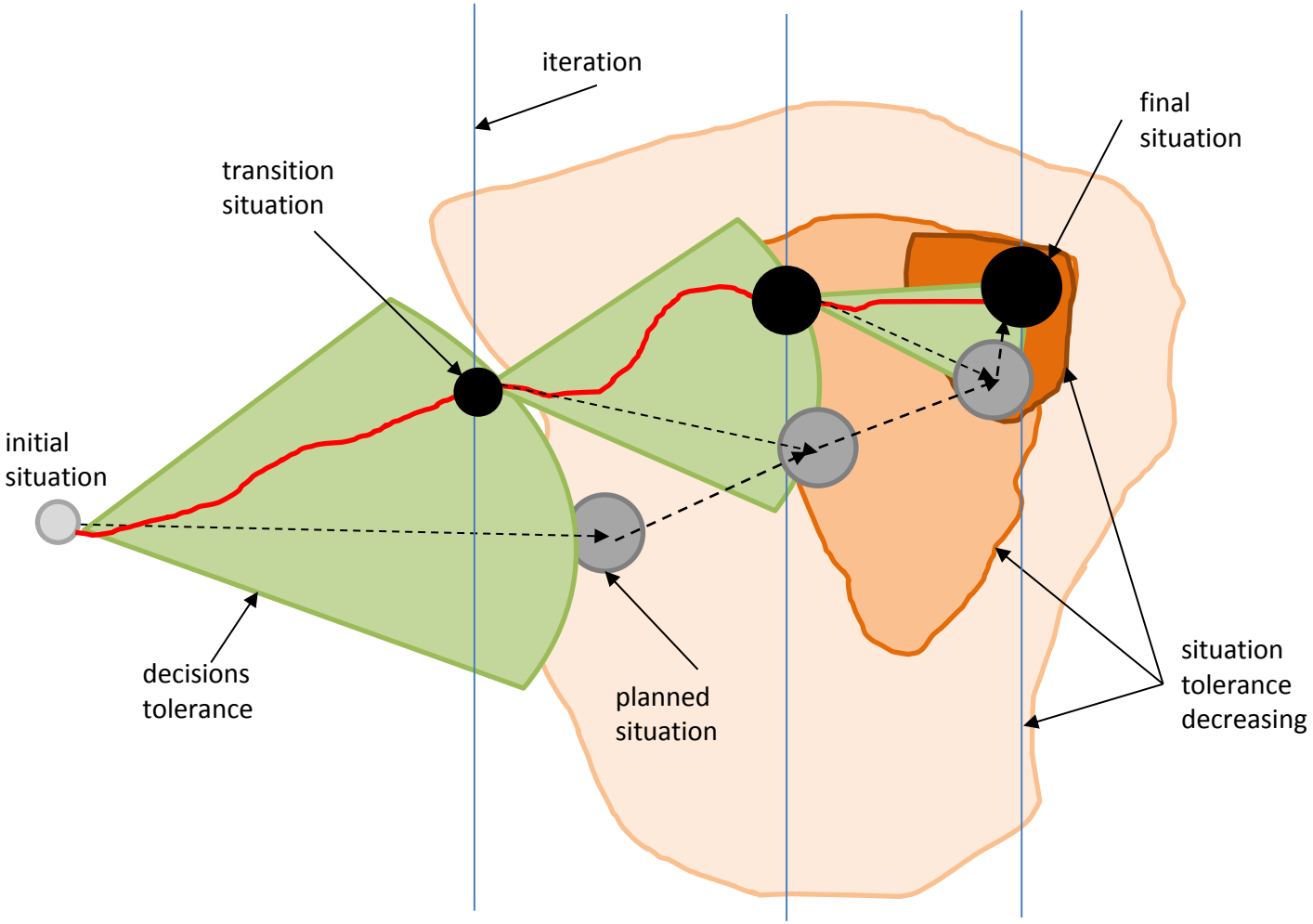
Iterative approach (2)



Iterative approach (3)



Iterative approach (4)



Two types of management

- **Life cycle management** which depends on the life cycle, e.g. what phases to finish or what phases to start
- **Work management** does not depends on the life cycle, e.g. what works to finish and what works to start, depending on various (typical in programme and project management) conditions such as availability of some resources, e.g. free staff
- And many management practices
  - PMI, PRINCE2, HERMS, TOGAF, ITSM, IT4IT, DevOps, Agile, etc.

No management practice covers the whole LC

	PMI	PRINCE2	TOGAF	ITSM	IT4IT	SCRUM	DevOps
Business Case		Fully			Partially		
Architecting	Partially	Partially	Fully				
Constriction	Fully	Partially	Partially	Partially	Partially	Fully	Fully
Transition	Partially	Partially	Partially	Fully	Partially	Partially	Fully
Pilot		Partially	Partially	Partially		Partially	Partially
Production				Fully	Partially		Partially
Retiring				Partially	Partially		
Decommissioning				Partially	Partially		

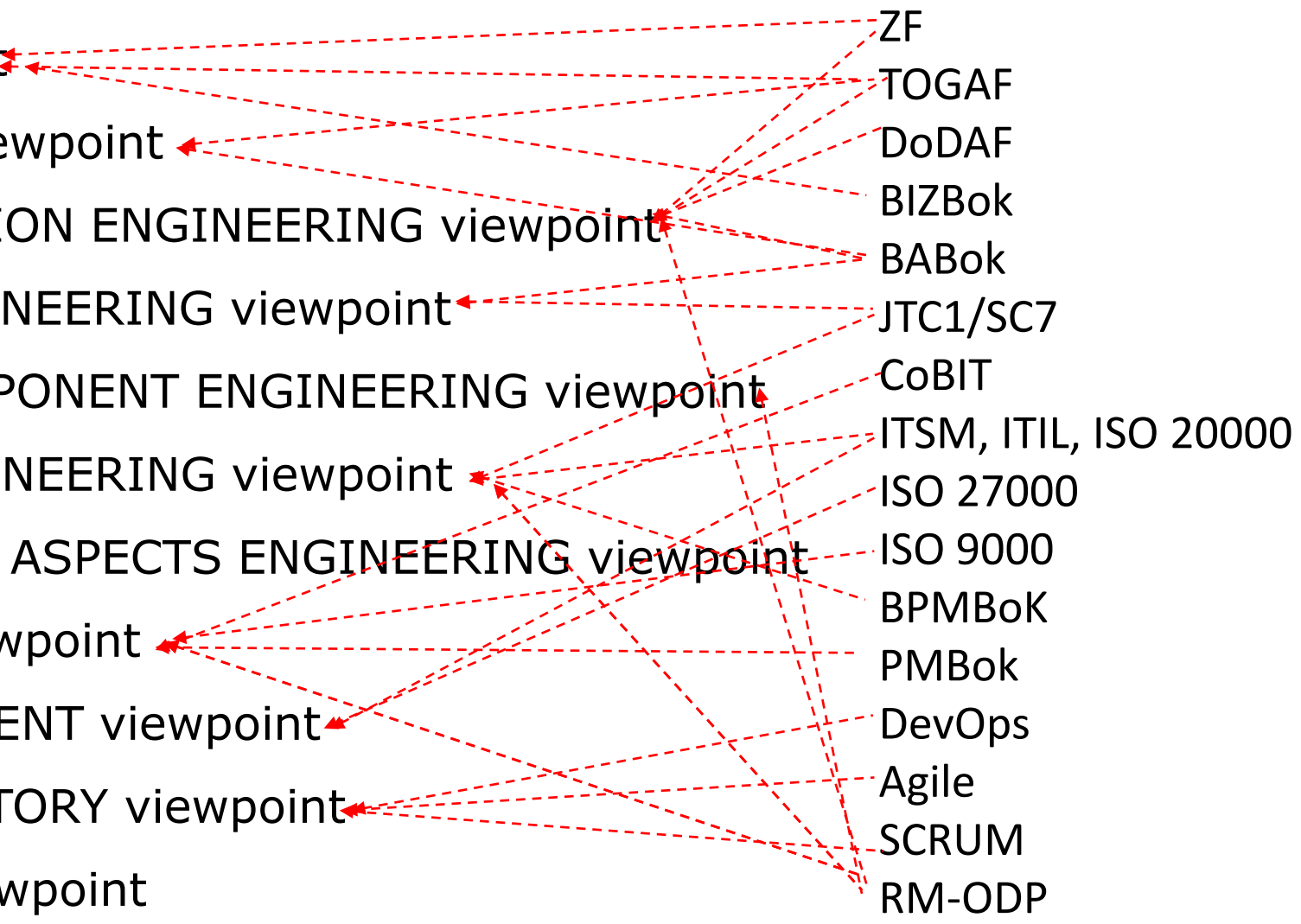
Recommendation: Consider different work management practice for different stages (rows).

Governance bodies

- **Standing or corporate governance** is different in each organisation. It may be some IT-Business management committee.
- **The traditional governance bodies** are
  - Project Management Office (PMO)
  - Change Advisory Board (CAB)
  - Architecture Review Board (ARB)
- **Steering committee** is a time-bound governance body
- **Project management group** is a time-bound governance body

SCRAM: a set of viewpoints (11) and model-types (107)

- VALUE viewpoint
- BIG PICTURE viewpoint
- SYSTEM-SOLUTION ENGINEERING viewpoint
- PLATFORM ENGINEERING viewpoint
- PLATFORM COMPONENT ENGINEERING viewpoint
- SOLUTION ENGINEERING viewpoint
- CROSSCUTTING ASPECTS ENGINEERING viewpoint
- CORPORATE viewpoint
- RISK MANAGEMENT viewpoint
- SOFTWARE FACTORY viewpoint
- STANDARDS viewpoint



SCRAM: VALUE viewpoint

- → Problem·space·overview·(see·7.2)¶
- → Problem·space·terminology·(see·7.3)¶
- → Problem·space·specifics·nomenclature·(see·7.4)¶
- → Problem·space·classifications·nomenclature·(see·7.5)¶
- → Stakeholders·nomenclature·(see·7.6)¶
- → Stakeholders'·concerns·nomenclature·(see·7.7)¶
- → Dependencies· between· generic· system· stakeholders,· stakeholders,· stakeholders'· concerns·and·categories·of·concerns·(see·7.8)¶
- → High-level·requirements·nomenclature·(see·7.9)¶
- → High-level·stories·nomenclature·(see·7.10)¶
- → High-level·use·cases·nomenclature·(see·7.11)¶
- → Problem·space·coverage·by·the·high-level·use·cases·(see·7.12)¶
- → The·mission·statement·(see·7.13)¶
- → The·vision·statement·(see·7.14)¶
- → The·strategic·goals·nomenclature·(see·7.15)¶

SCRAM: BIG PICTURE viewpoint

- → Solution·space·boundaries·(see·7.16)¶
- → Solution·space·terminology·(see·7.17)¶
- → Solution·space·constraints·nomenclature·(see·7.18)¶
- → Solution·space·classifications·nomenclature·(see·7.19)¶

- → Solution·space·essential·emergent·characteristics·nomenclature·(see·7.20)¶
- → Dependency· matrix· between· problem· space· essential· high-level· requirements· and· solution·space·constraints·vs.·solution·space·essential·emergent·characteristics·(see· 7.21)¶
- → Solution·space·architecture·principles·nomenclature·(see·7.22)¶
- → Dependency· matrix· between· solution· space· essential· emergent· characteristics· vs.· solution·space·architecture·principles·(see·7.23)¶
- → High-level·illustrative·diagrams·(see·7.24)¶
- → High-level·business·map·(see·7.25)¶
- → Beneficiaries'·journeys·nomenclature·(see·7.26)¶
- → High-level·reference·capability·map·(see·7.27)¶
- → High-level·process·map·(see·7.28)¶
- → High-level·architecture·(see·7.29)¶

SCRAM: SYSTEM-SOLUTION viewpoint

- → Capability·map·(see·7.30)¶
- → Low-level·use·cases·nomenclature·(see·7.31)¶
- → Function·map·(see·7.32)¶
- → Partners·nomenclature·(see·7.33)¶
- → Process·map·(see·7.35)¶
- → Services·nomenclature·(see·7.34)¶
- → Decisions·nomenclature·(see·7.36)¶
- → Events·nomenclature·(see·7.37)¶
- → Data·schemas·nomenclature·(see·7.38)¶
- → Information·flows·nomenclature·(see·7.39)¶
- → Document/content·classifications·nomenclature·(see·7.40)¶
- → Key·performance·indicators·nomenclature·(see·7.41)¶
- → Reports·nomenclature·(see·7.42)¶
- → Platforms·nomenclature·(see·7.43)¶

PLATFORM ENGINEERING viewpoint

- → Platform·overview·(see·7.44)¶
- → Platform·terminology·(see·7.45)¶
- → Platform·components·nomenclature·(see·7.46)¶
- → Platform·solutions·nomenclature·(see·7.47)¶
- → Standards·and·norms·nomenclature·(see·7.48)¶
- → Platform·software·factory·configuration·(see·7.49)¶
- → Platform·governance,·management·and·operations·manual·(see·7.50)¶

PLATFORM COMPONENT ENGINEERING viewpoint

- → Platform·component·overview·(see·7.51)¶
- → Platform·component·terminology·(see·7.52)¶
- → Platform·component·capability·map·(see·7.53)¶
- → Platform·component·function·map·(see·7.54)¶
- → Platform·component·process·map·(see·7.55)¶
- → Platform·component·business·specifications·(see·7.56)¶
- → Platform·component·information·specifications·(see·7.57)¶
- → Platform·component·application·specifications·(see·7.58)¶
- → Platform·component·data·specifications·(see·7.59)¶
- → Platform·component·infrastructure·specifications·(see·7.60)¶
- → Platform·component·security·specifications·(see·7.61)¶
- → Platform·component·services·and·APIs·nomenclature·(see·7.62)¶
- → Platform·component·software·factory·configuration·(see·7.63)¶
- → Platform·component·management·and·operations·manual·(see·7.64)¶

SOLUTION ENGINEERING viewpoint

- → Solution·overview·(see·7.65)¶
- → Solution·terminology·(see·7.66)¶
- → Solution·capability·map·(see·7.67)¶
- → Solution·function·map·(see·7.68)¶
- → Solution·process·map·(see·7.69)¶
- → Solution·business·specifications·(see·7.70)¶
- → Solution·information·specifications·(see·7.71)¶
- → Solution·application·specifications·(see·7.72)¶
- → Solution·data·specifications·(see·7.73)¶
- → Solution·infrastructure·specifications·(see·7.74)¶
- → Solution·security·specifications·(see·7.75)¶
- → Solution·services·and·APIs·nomenclature·(see·7.76)¶
- → Solution·software·factory·configuration·(see·7.77)¶
- → Solution·management·and·operations·manual·(see·7.78)¶

CROSSCUTTING ASPECTS ENGINEERING viewpoint

- → Interoperability·aspect·(see·7.79)¶
- → Security·aspect·(see·7.80)¶
- → Privacy·aspect·(see·7.81)¶
- → Safety·aspect·(see·7.82)¶
- → Reliability·and·resilience·aspect·(see·7.83)¶
- → Low·cost·of·operations·and·time-to-market·aspect·(see·7.84)¶
- → Self-reference·aspect·(see·7.85)¶

CORPORATE viewpoint

- → Organisational·structure·(see·7.86)¶
- → Governance·structure·(see·7.87)¶
- → Project·management·(see·7.88)¶
- → Corporate·manual·(see·7.89)¶
- → Independent·evaluation·(see·7.90)¶
- → Ethics,·integrity·and·ani-corruption·(see·7.91)¶

- → Environment·and·health·(see·7.92)¶

RISK MANAGEMENT viewpoint

- → Risk·management·aspect·(see·7.93)¶
- → Compliance·management·aspect·(see·7.94)¶
- → Regulatory·management·aspect·(see·7.95)¶
- → Crime·prevention·and·detection·(see·7.96)¶
- → Auditing·(see·7.97)¶
- → Independent·investigation·(see·7.98)¶
- → Crisis·management·(see·7.99)¶

SOFTWARE FACTORY viewpoint

- → Software·factory·overview·(see·7.100)¶
- → Prototyping·practices·(see·7.101)¶
- → Engineering·practices·(see·7.102)¶
- → Assembling·practices·(see·7.103)¶
- → Testing·practices·(see·7.104)¶
- → Deployment·practices·(see·7.105)¶
- → Monitoring·practices·(see·7.106)¶

STANDARDS viewpoint

- → Existing·standards·nomenclature·(see·7.107)¶
- → Potential·standards·nomenclature·(see·7.108)¶

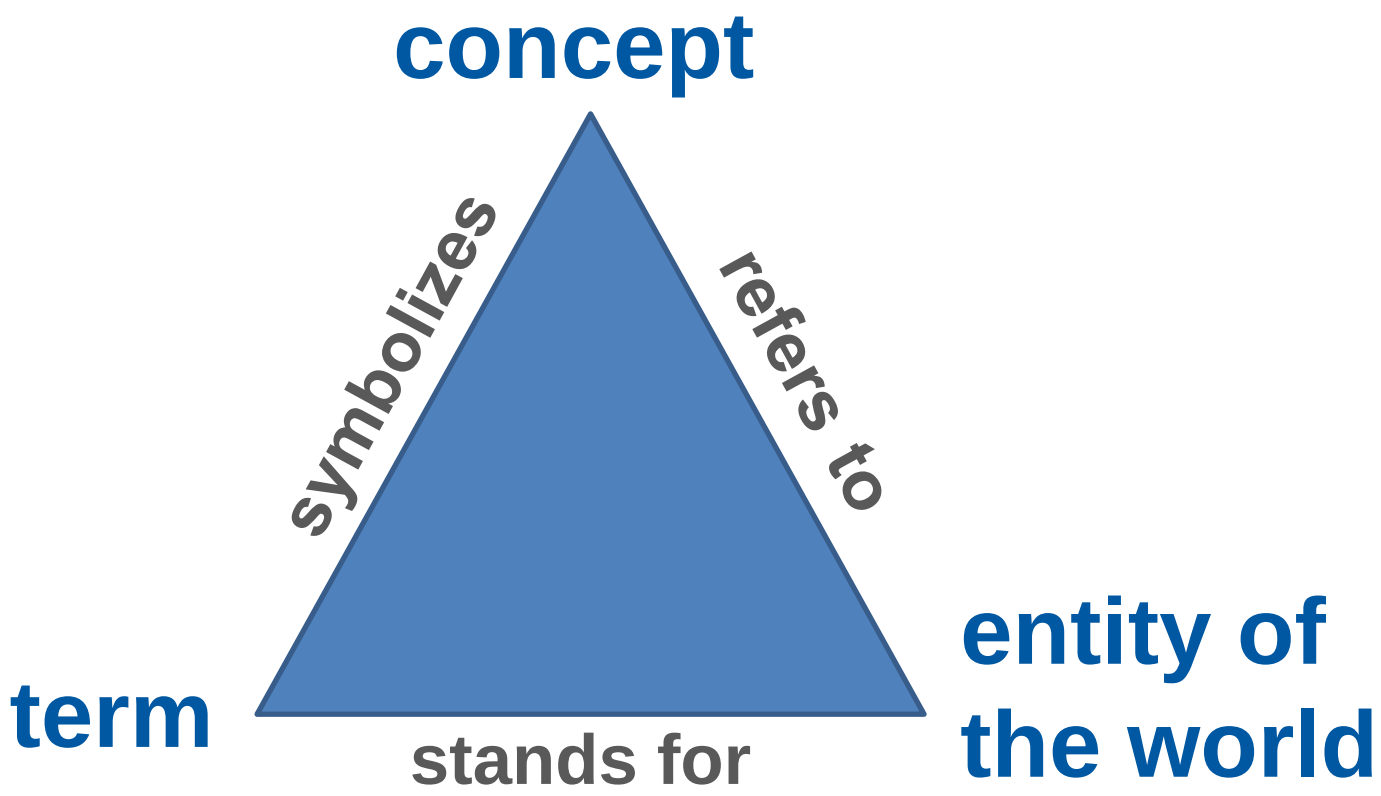
Great! However you must be very good with terminology

Architecture documentation

- considers many different domains
- employs science and technology which are social endeavours… an important task for them is communication, as enabled by language
- is written in a specialist language

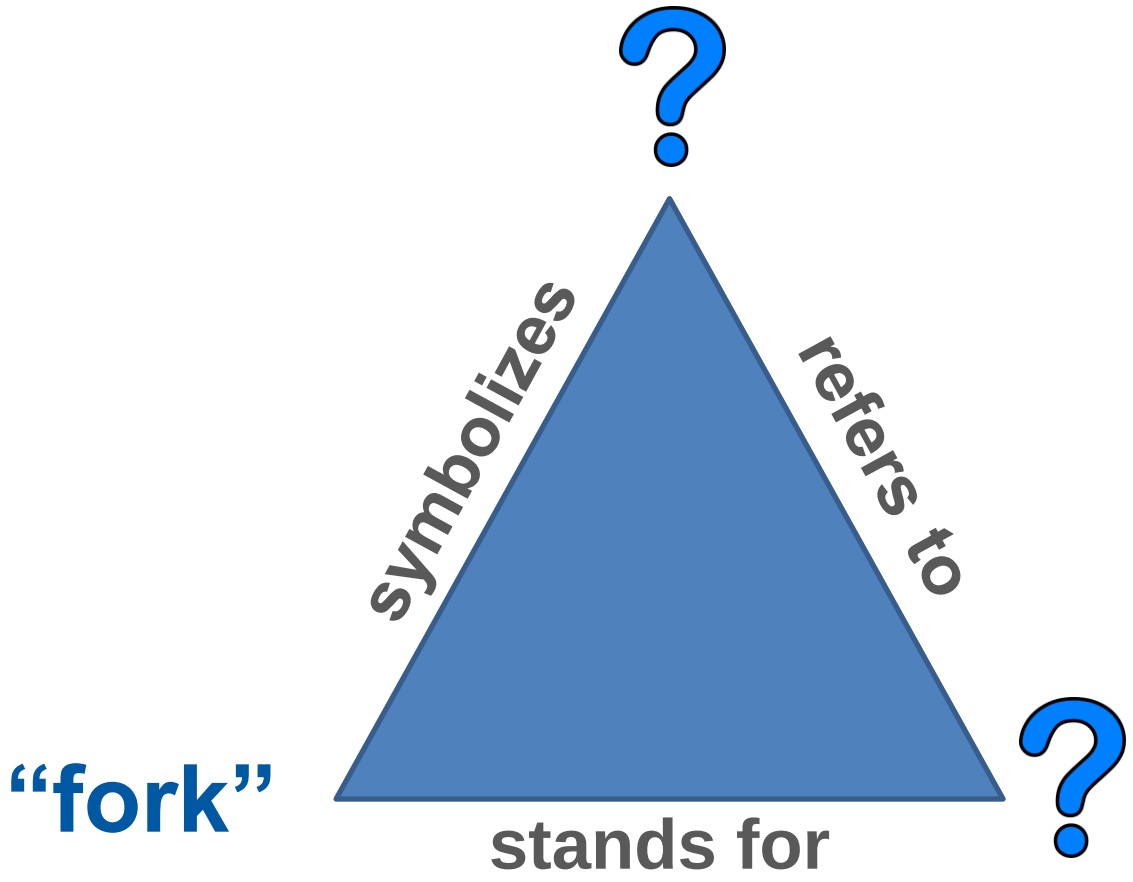
Terminology basics (1)

- communication of specialist knowledge and information is dependent on the creation and use of terminology



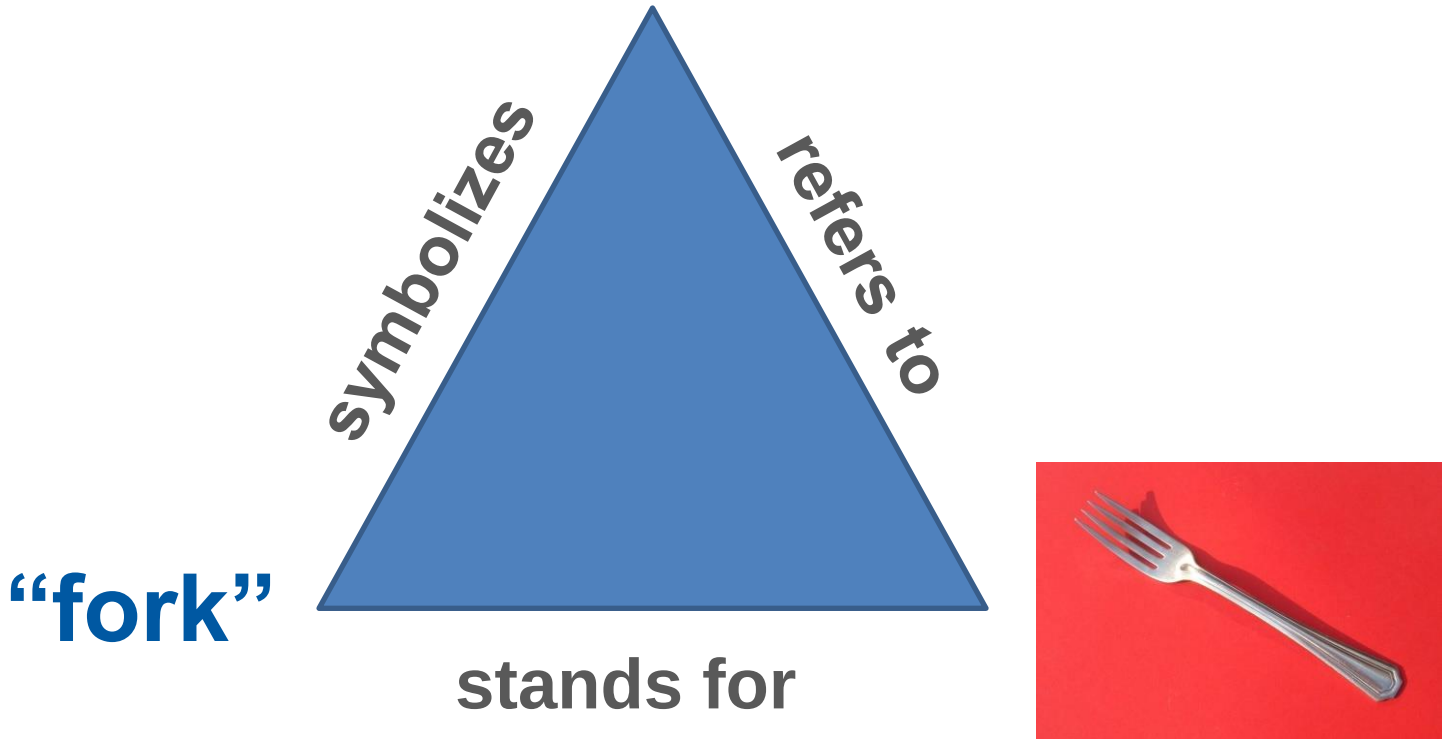
Terminology basics (2)

Think of the term “fork”



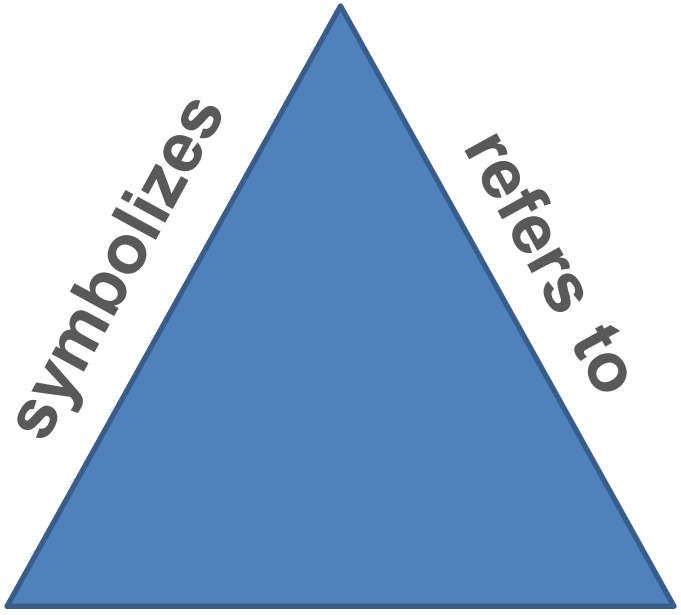
Terminology basics (3)

implement with a handle and two or more prongs used for lifting food to the mouth or for holding it when cutting



Terminology basics (4)

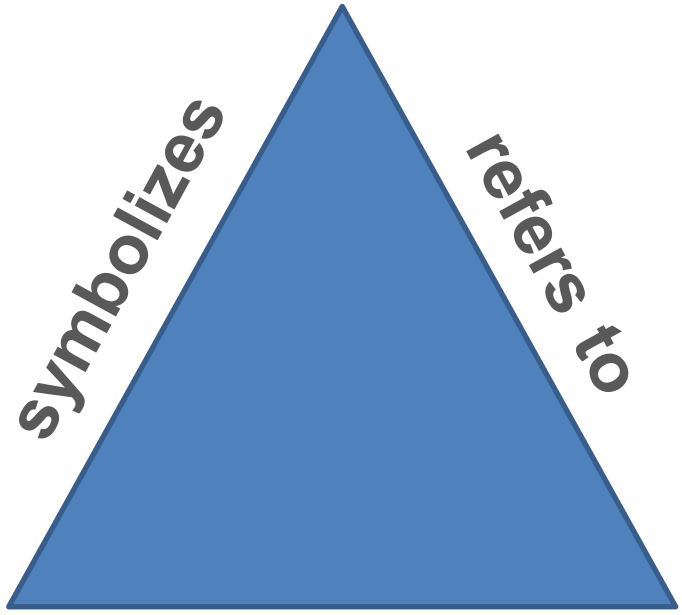
part of a glove at the junction of two fingers, or finger and thumb



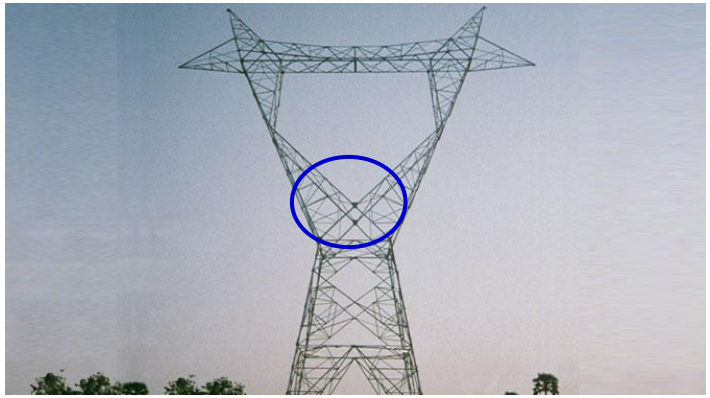
“fork” (IEC 60903, live working)

Terminology basics (5)

component which is part of a *top* *hamper* of a *tower*

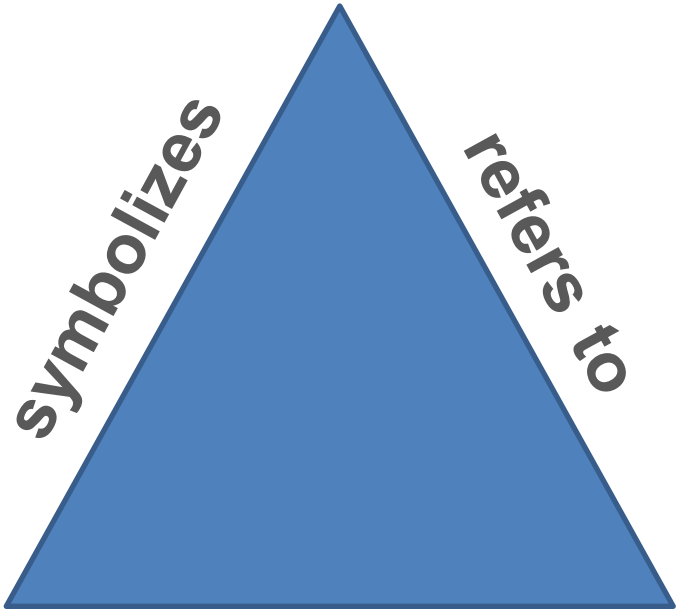


“fork”
(IEV 466-08-13, overhead lines)



Terminology basics (6)

attachable tool used to hold a component in position when using other tools



"*holding* fork"
(IEC 60743, live working)

System of concepts basics (1)

- A new concept is defined as existing concepts with some essential characteristics
- Example
  - Smart city is a city which is rebuilt as a digital system

- Some existing concepts may come from the linguistic dictionary, e.g. English Oxford Dictionary

System of concepts basics (2)
Validating rule

- A term in a sentence may be replaced by its definition without making the new sentence invalid

125 **3.4**
126 **city***
127 urban community falling under a specific administrative boundary
128 NOTE 1: A city is sometimes referred to as a municipality or a local government.
129 NOTE 2: Cities can help to alleviate increasing pressure on the environment and natural resources caused by global
130 urbanization through the development of holistic and integrated policies.
131 NOTE 3: Cities are always that has defined geographical boundaries.
132 NOTE 4: Cities refer to any geographically located population.
133 SOURCE : ADAPTED FROM ISO 37100:2016, 3.2.1

urban

ADJECTIVE

1 In, relating to, or characteristic of a town or city.
'the urban population'
+ More example sentences + Synonyms

178 **3.8**
179 **community***
180 group of people with an arrangement of responsibilities, activities and relationships

- "**city** – urban community falling under a specific administrative boundary"
- "**city** – relating to a town or city group of people with an arrangement of responsibilities, activities and relationships falling under a specific administrative boundary -- **NONSENSE**

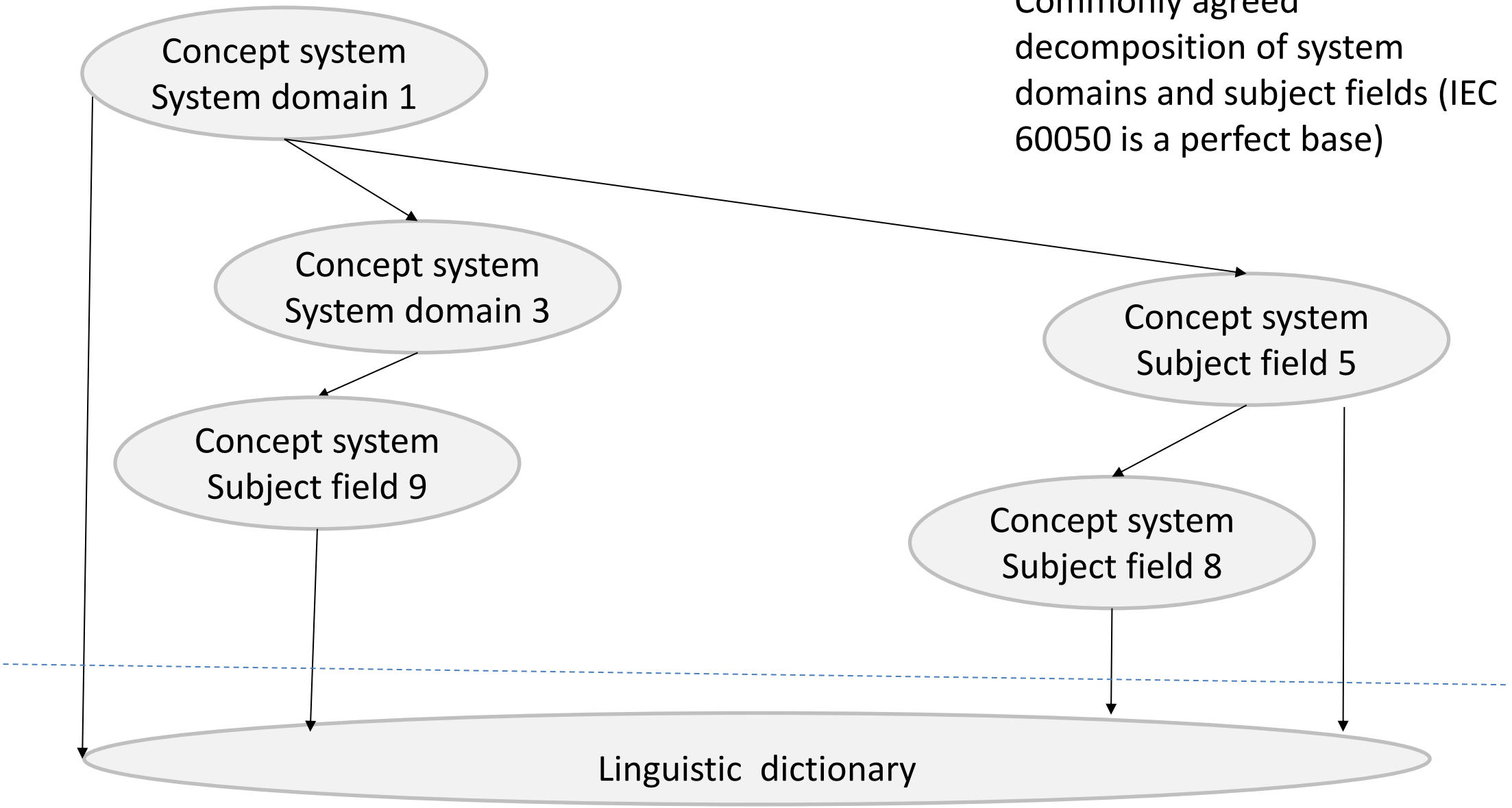
The problem

- The complexity
  - each **system domain** is multidisciplinary
  - a system domain **concept system** is a composition of many concepts from various concept systems - from other system domains and **subject fields**
- An example – 40 concepts for IEC 60050-831
  - 1 concept (“city”) has to be defined
  - 1 concept (“smart city”) has to be derived
  - 35 concepts to be borrowed from other domains and subject fields
  - 3 concepts are joint two concepts – they need only examples
- The problem – how to create and maintain a system domain (e.g. Smart Cities) concept system efficiently and effectively

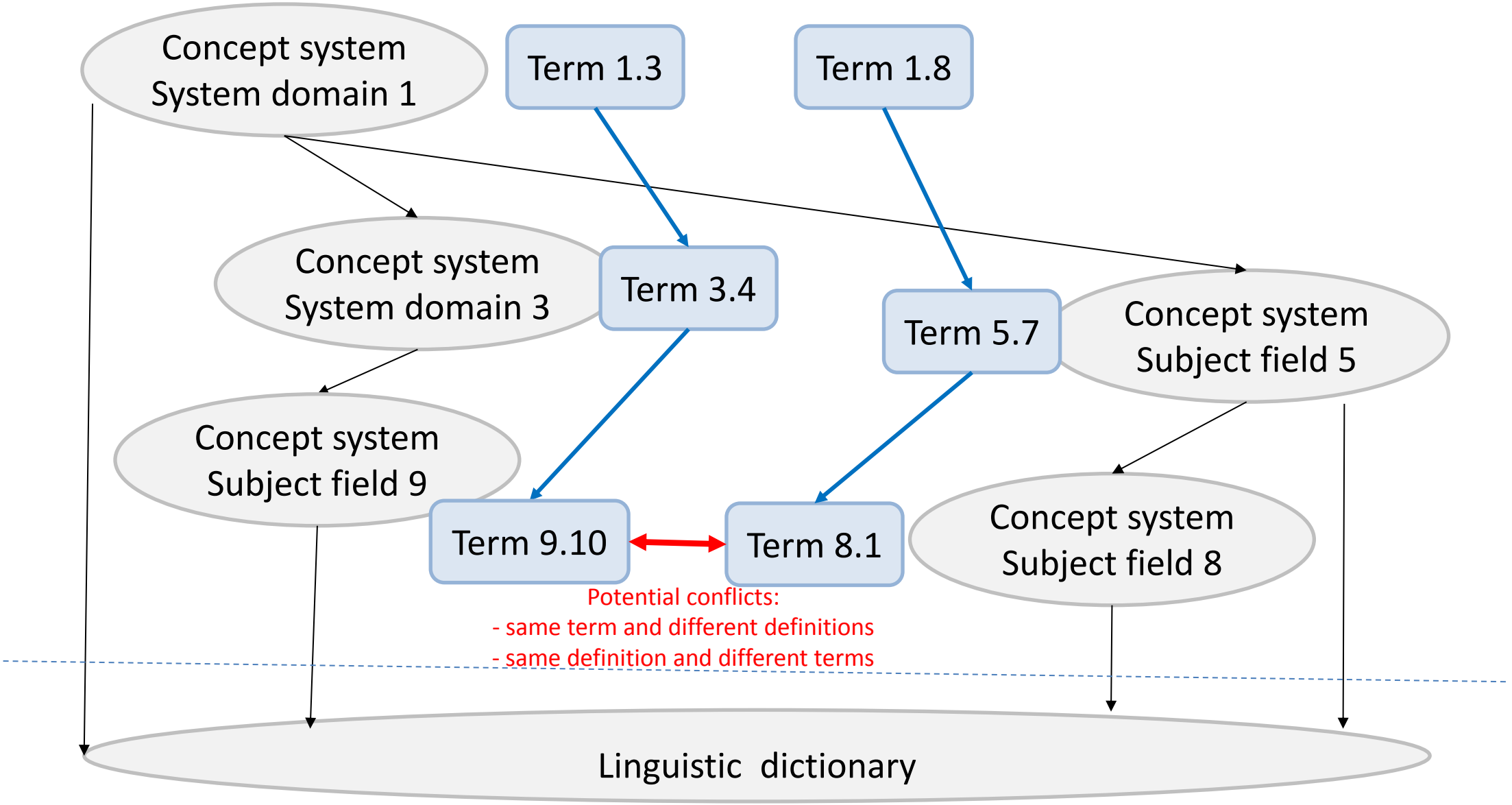
Potential solution outline

- Theory
  - Terminology
  - Concept maps
  - Graphs (sets of vertices and edges)
- **Method**
  - Decomposition into a set of concept systems with an explicit ownership
  - Procedure for assembling a new concept system from existing concept systems
- Tooling
- Externalising within the IEC SyCs
- Externalising within the IEC TCs
- Externalising within the international standardisation

Decomposition into a set of concept systems



Understanding of potential conflicts

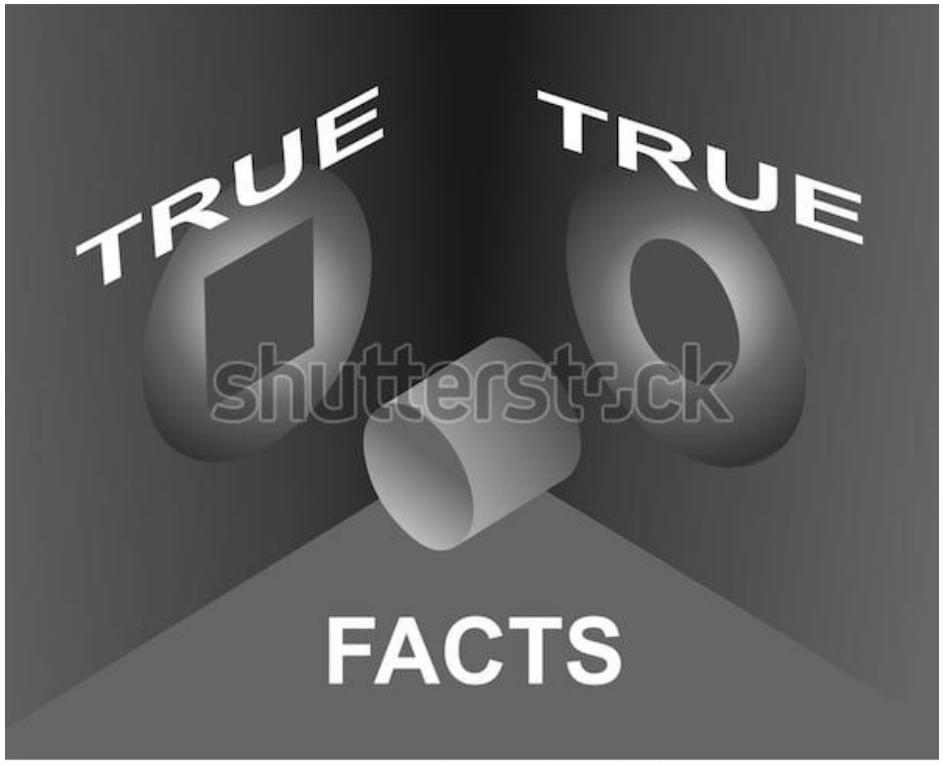


Procedure to avoid conflicts “by-design”

- Each concept system is made as a **planar directed acyclic graph**
  - a proof is available (Thanks to **Anatoly Kazakov**) at https://sec2017.blogspot.com/search/label/%23Glossary
- Consistency of each concept system is formally validated (we need a tool, ideally, ontology based)
- Several agreed rules are used for assembling a new concept system from existing concept systems
  - adopting an existing concept
  - modifying an existing concept
  - defining a new concept
- All concept systems and their elements are versioned

A good concept system is very valuable

- Never ever argue about a concept in isolation
  - always consider context
  - employ your concept system



Questions?

- E-mail: alexandre.samarine@gmail.com
- Mobile: +41 76 573 40 61