Abschiedsvorlesung
Prof. h.c. Dr. Frank J. Furrer

TU Dresden, January 29, 2024 – APB/E023

TU Dresden, 2013 - 2023

https://www.gjplastics.co.uk/caution-sign/

Prof. Dr. Frank J. Furrer

Software Development:
From Black Art to Engineering Science

A pleasurable Promenade from 1848 to 2024

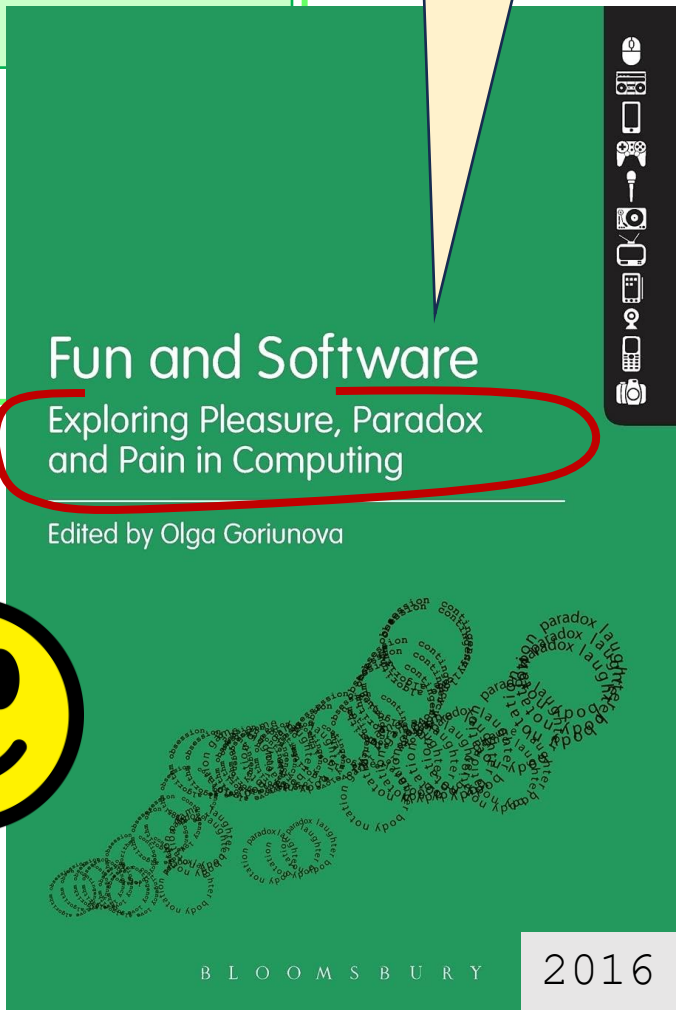
Lecture downloadable (pdf file) from:
https://st.inf.tu-dresden.de/teaching/Prof_FJ_Furrer/

Very special
Personal Literature
Recommendations

Topics

Software Development: From Black Art to Engineering Science
A pleasurable Promenade from 1848 to 2024

Principle-based Software Engineering



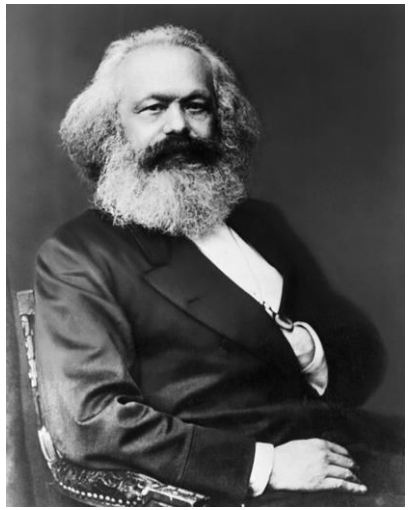
https://pixabay.com/de/

Prof. Dr. Frank J. Furrer

Software Development:
From Black Art to Engineering Science

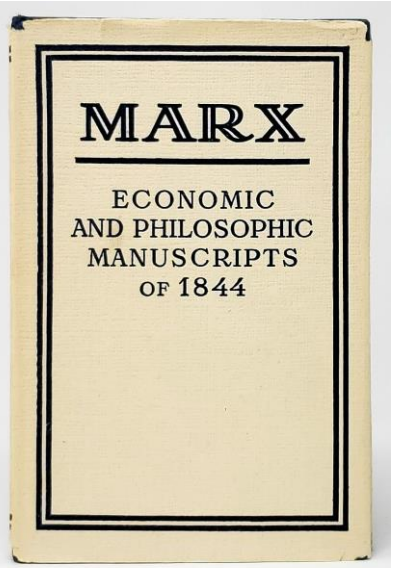
Introduction

https://www.geo.de



Philosopher Karl Marx argues that what distinguish human beings from animals are **tools** – the product of human's intelligence, experience, and creativity

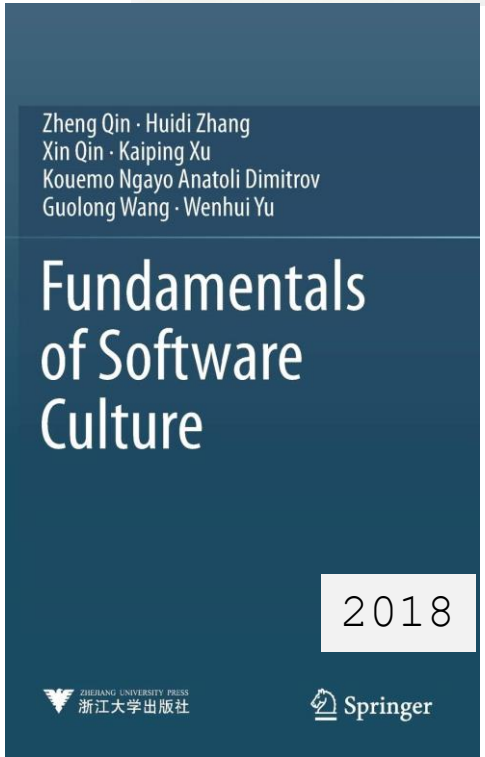
https://biblio.sg/book



The evolutionary history of **tools** is the evolutionary history of human civilization

https://networkencyclopedia.com/mainframe

As the most important **tool** nowadays
– **the computer** –
is absolutely a sign of culture



And **software**
– the approach to control computers –
is the crystallization of human wisdom

Zheng Qin et. al. / ISBN 978-981-13-0700-3 / 2018

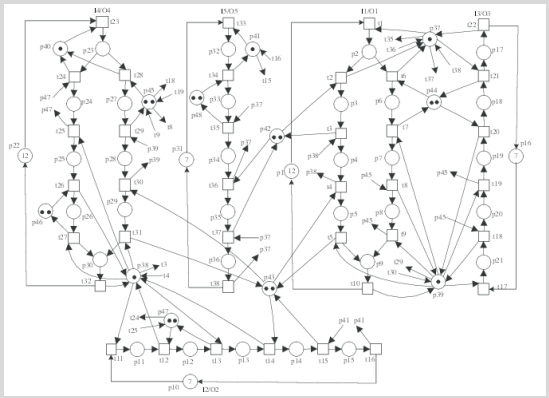
Prof. Dr. Frank J. Furrer

Software Development:

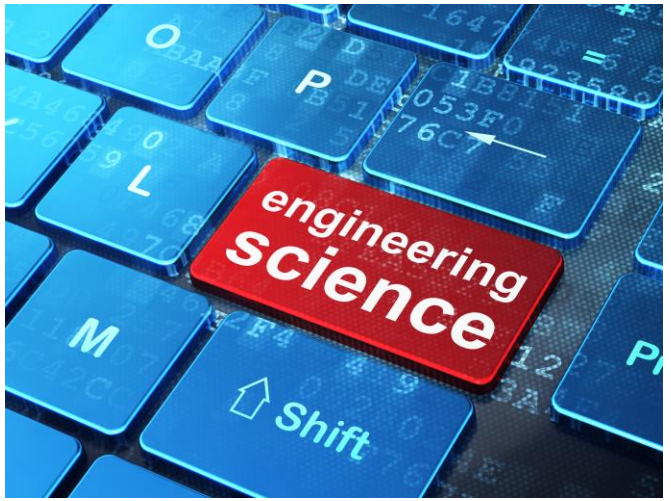
From Black Art to Engineering Science

http://de.123rf.com

1848 – 2024



A pleasurable Promenade



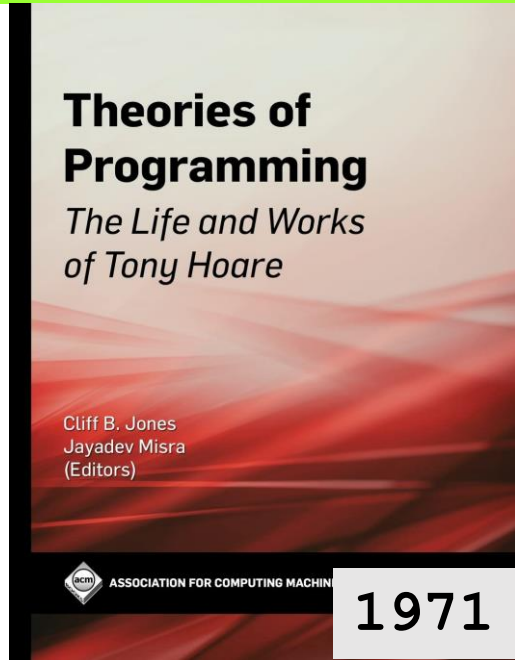
https://www.canstockphoto.com/science-concept-engineering-science-on-30698717.html

https://timenote.info/de/Ada-Lovelace https://www.researchgate.net/figure/A-large-scale-Petri-net-model-from-Li-and-Zhou-2005-24_fig6_224170463

http://de.123rf.com

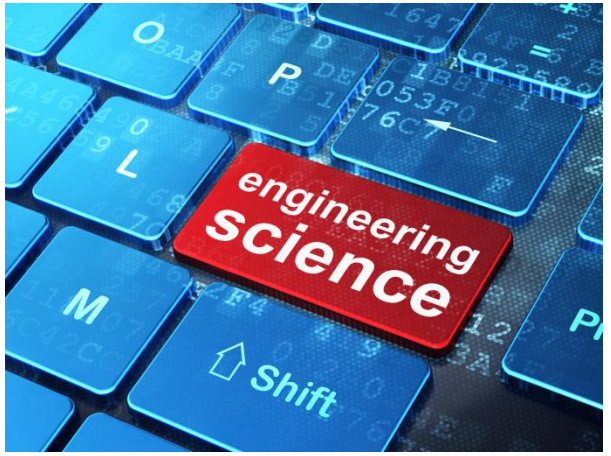
Software Development started 1848 as a **black art**:

- Experimental approaches/Intuition
- No theory of programming
- Ad-hoc formalism
- «Pen and paper» tools
- "Genius at work"
- Each program an individual piece of black art



Software Development becomes an **Engineering Science**:

- Founded in proven, teachable, and enforcable principles
- Higly formalized, based on mathematics/logics
- Strong, complexity-minimizing system architecture
- Focus on quality attributes (Safety, security, performance, …)
- Provably correct-by-design
- Massively automated
- Thoroughly supported/guided by artificial intelligence
- Reliably protected from accidents/incidents by runtime monitoring



https://www.canstockphoto.com/science-concept-engineering-science-on-30698717.html

Software Knowledge

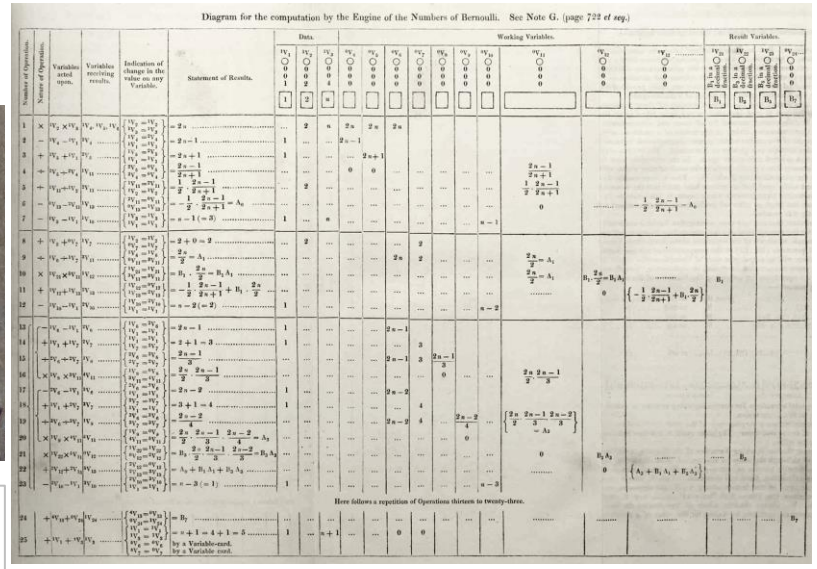
Continuous, accelerated generation of software-knowledge

Time

1848

2024

Ada Lovelace
1st Program



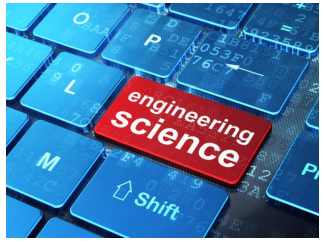
… < 100 Lines of Code

https://www.dreamstime.com

176 years

Global Software Market:
2022: 589 Billion (Milliarden)
2030: 1'789 Billion US$
+ Embedded Software
https://www.precedenceresearch.com

https://timenote.info/de/Ada-Lovelace https://projectlovelace.net/problems/ada-lovelaces-note-g



Today's 5 Questions:

1 - Is today's software development already an engineering science?

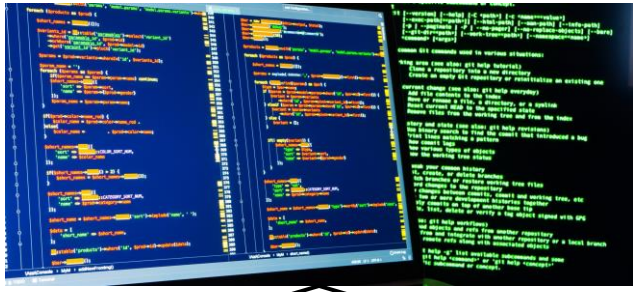
2 - Does today's software development still have a component of black art?

3 - Which is the path for software development towards true engineering science?

4 - How do we know when we have reached the state of engineering science?

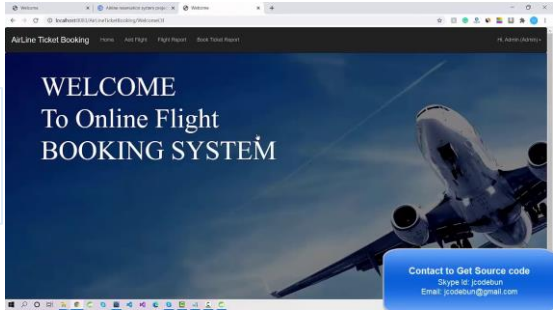
5 - Which mechanisms can we use to achieve the state of engineering science?

Software

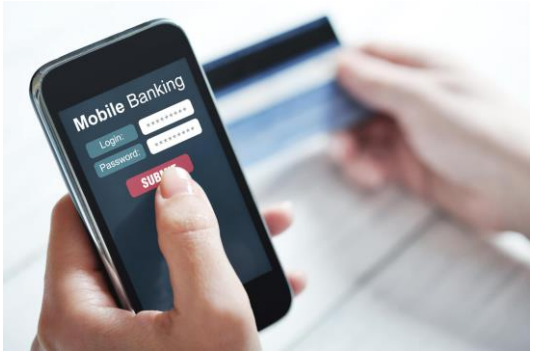


Commercial Systems

Airline Reservation System

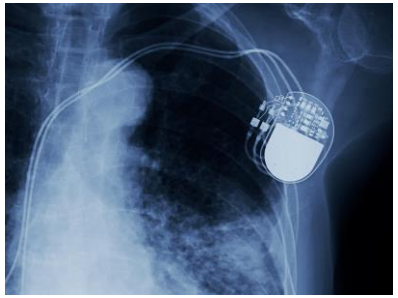


Mobile e-Banking



→ **NO** Impact on the physical World

Cyber-Physical Systems



Heart Pacemaker

Fresh Water Treatment Plant

→ **Massive** Impact on the physical World

https://www.youtube.com/watch?v=dtTkuy_Jpy8 https://www.ebas.ch https://www.tampabaywater.org https://medlineplus.gov

https://www.mtu.edu/cs/undergraduate/software/what/

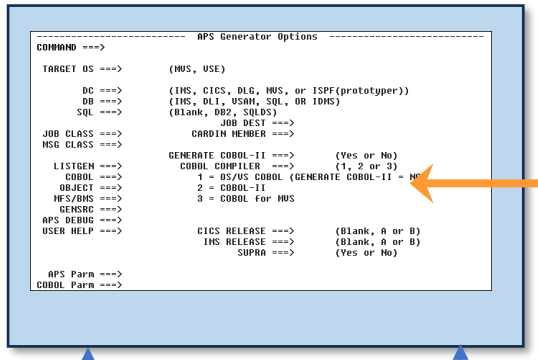
Risks:

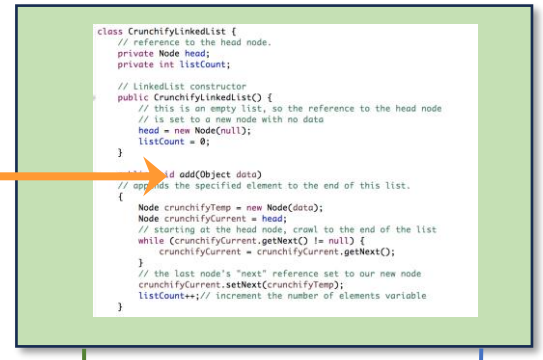
- Safety
- Security

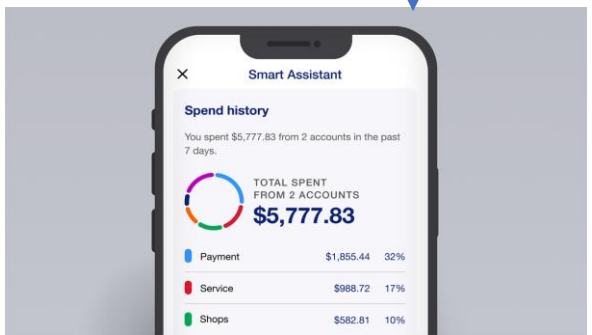
Commercial Systems

Cyber-Physical Systems

Most Functions are implemented in Software







Sensor

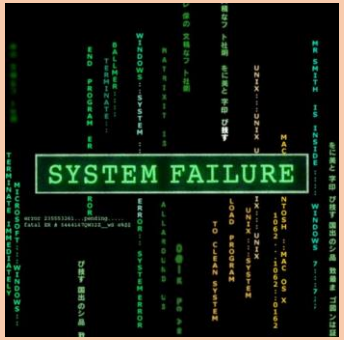
Actuator

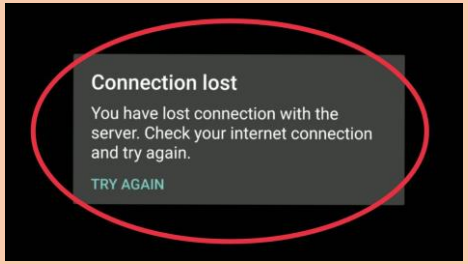
https://www.usbank.com

https://www.britannica.com/money

https://www.mtu.edu/cs/undergraduate/software/what/

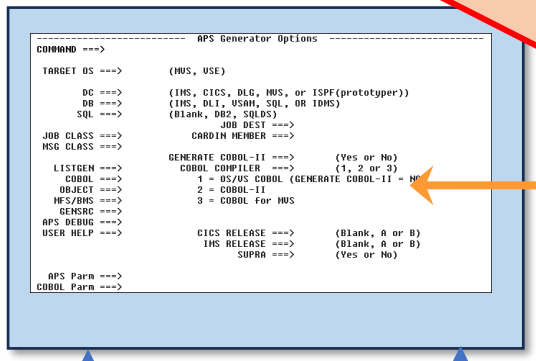
Major Risk # 1: Faults, Failures, Errors

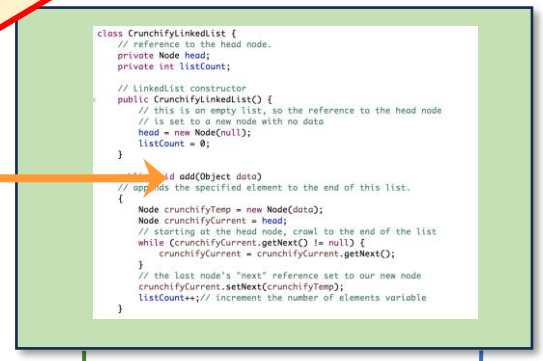


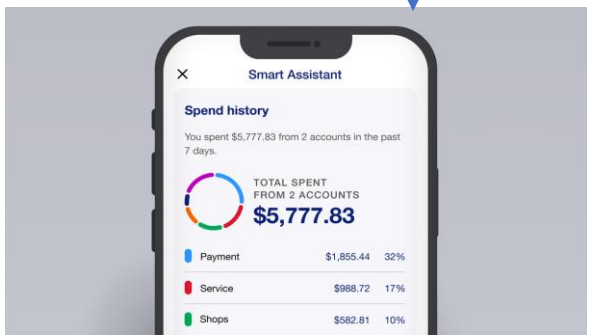


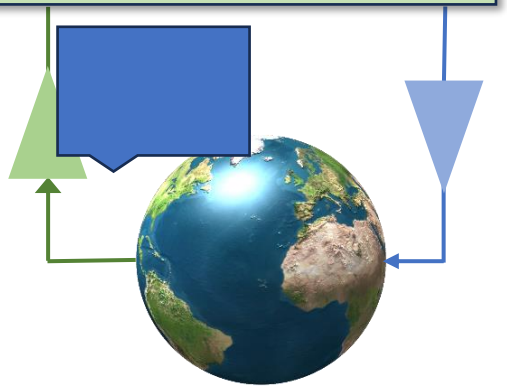
Major Risk # 2: Malicious Activities

Lost Functionality are implemented in Software





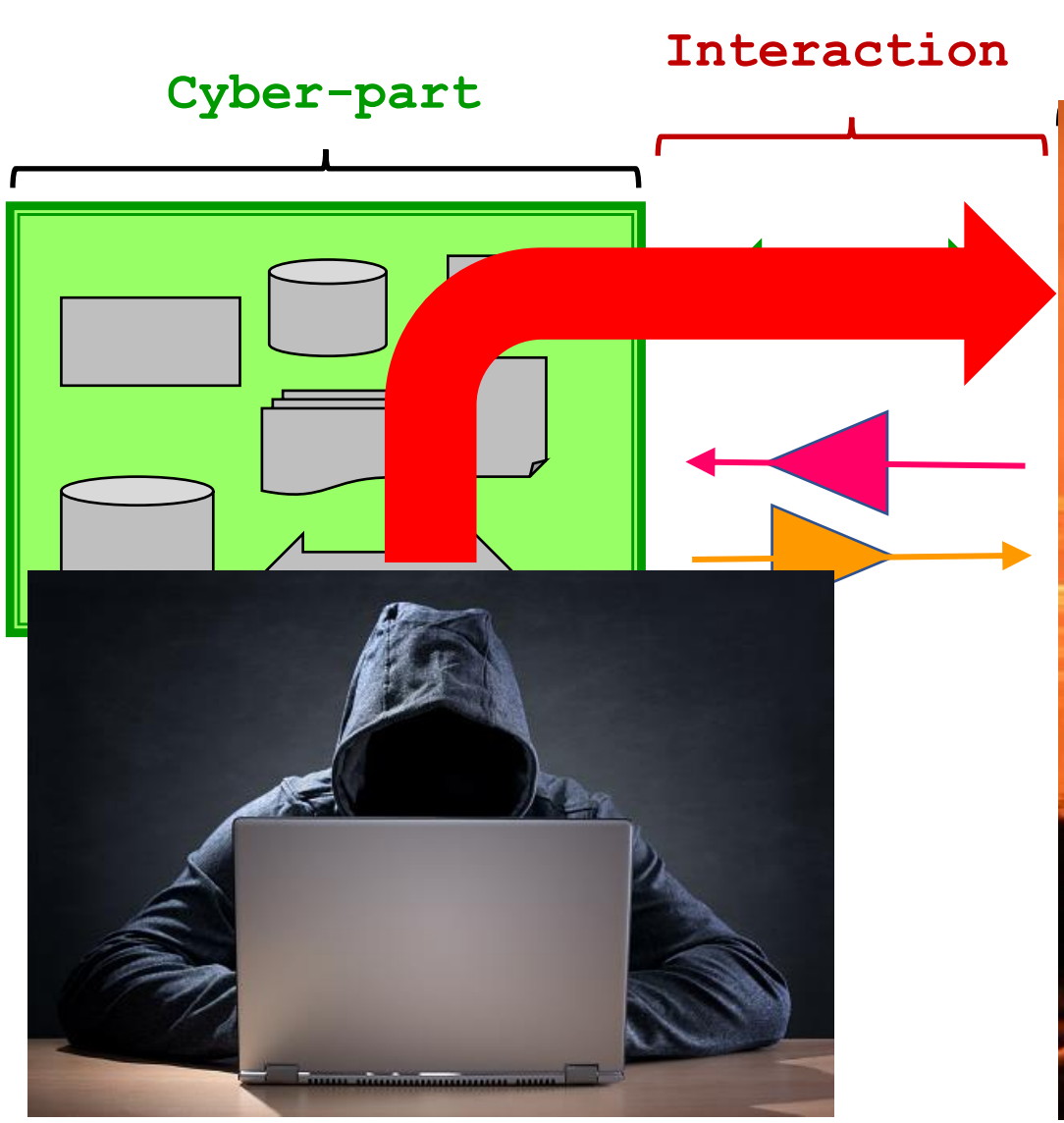




https://www.usbank.com

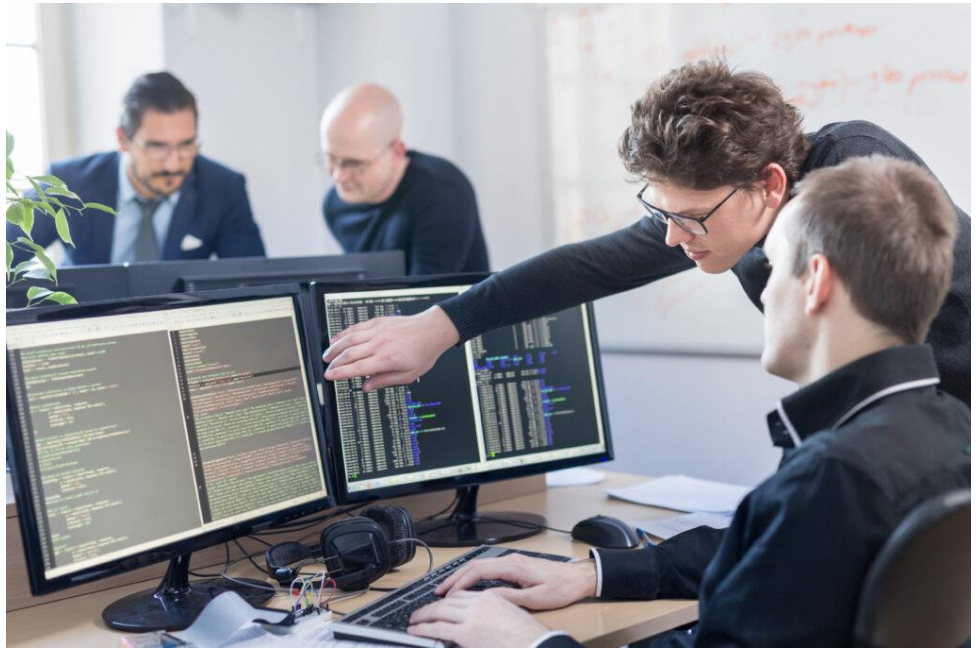
https://www.britannica.com/money

https://www.mtu.edu/cs/undergraduate/software/what/



http://www.etemaaddaily.com

https://themaddev.com/the-5-problem-solving-skills-of-great-software-developers/



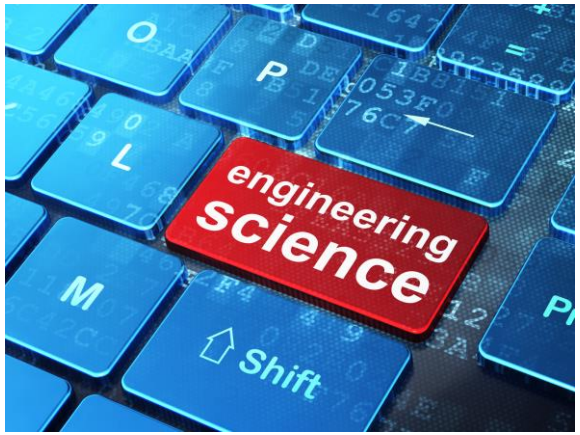
It is our undeniable responsibility to build, maintain, and evolve **safe** and **secure** systems

https://www.terreactive.ch

A safe and secure system
is the result of
competent and responsible
engineering

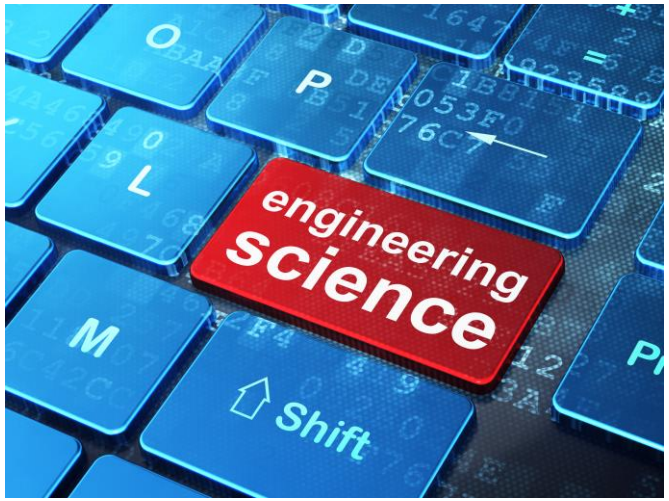
https://www.canstockphoto.com/science-concept-engineering-science-on-30698717.html

http://de.123rf.com



Software Development: 2 key Questions

http://de.123rf.com



https://www.canstockphoto.com/science-concept-engineering-science-on-30698717.html

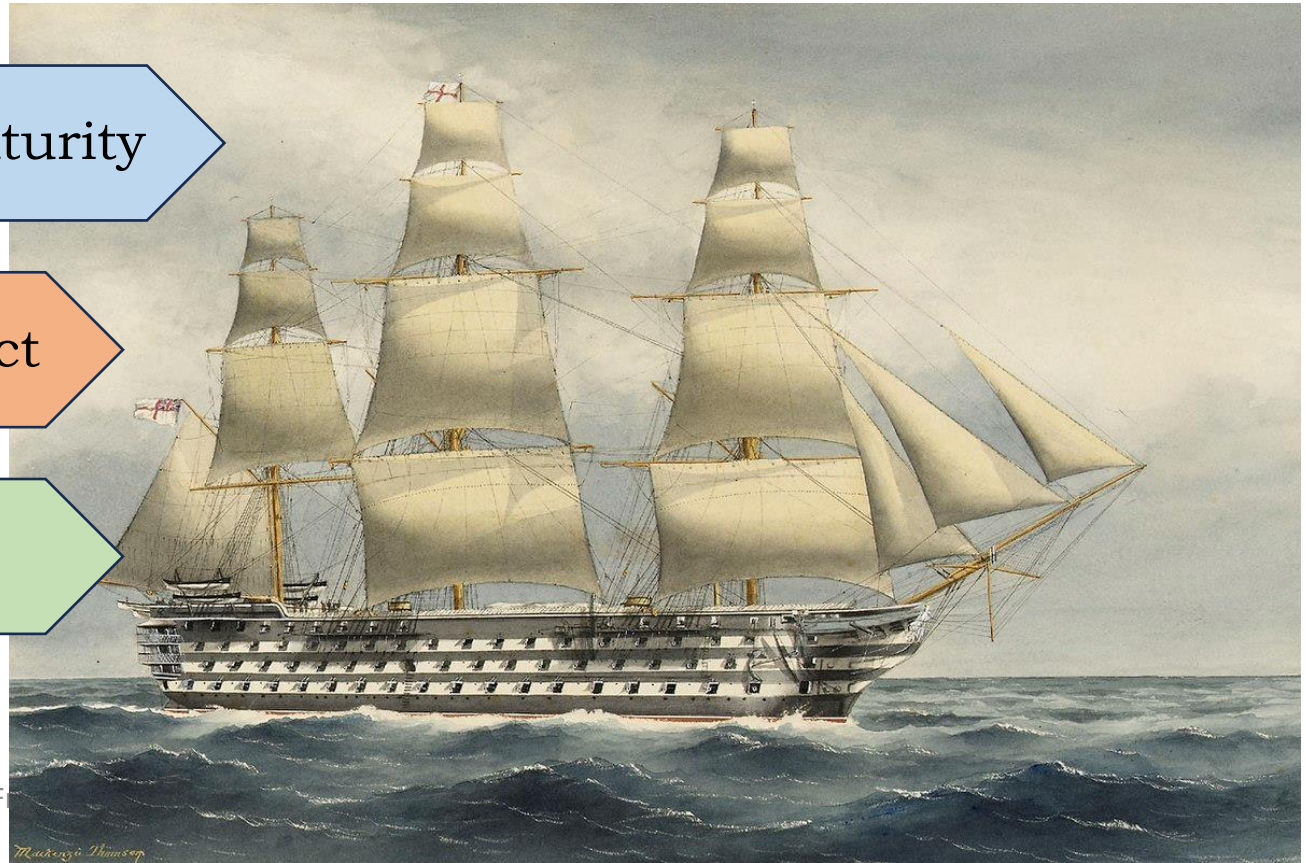
What is needed for this Journey ?

How do we know when we have reached Engineering Science ?

https://en.wikipedia.org/wiki/HMS_Victoria_%281859%29

What is needed for this Journey ?

1. Development & Operations **Process** Maturity
2. System & Software **Architecture** Impact
3. Degree of Abstraction/**Formalization**



Black Art

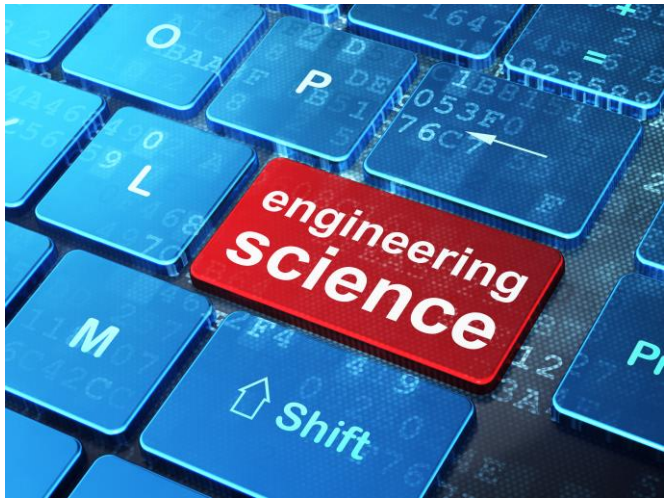
Engineering Science

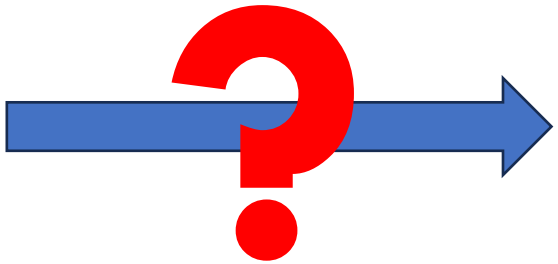
low — Development & Operations **Process** Maturity — high

low — System & Software **Architecture** Impact — high

low — Degree of Abstraction/**Formalization** — high

1848 **2024** **20??**





https://www.canstockphoto.com/science-concept-engineering-science-on-30698717.html

How do we know when we have reached Engineering Science ?

https://www.imago-images.de/fotos-bilder/F1-Flagge-Ziel

Black Art

Engineering Science

Safety Accidents

Software-induced
- SafetyAccidents
- Security Incidents

⇒ Zero

Security Incidents

https://www.independent.co.ug

https://www.healthcareitnews.com

1848 2024 20??

https://www.imago-images.de/fotos-bilder/F1-Flagge-Ziel

Prof. Dr. Frank J. Furrer

Software Development:
From Black Art to Engineering Science

8 Examples

https://fighterjetsworld.com

Airbus A400M Crash at Seville (Spain) in 2015

Reason:

Configuration data for 3 engines **missing**

⇒ **Software Integrity Check violated**

Boeing 737 MAX Crashes:

Lion Air Flight 610 on October 29, 2018

Ethiopian Airlines Flight 302 on March 10, 2019

https://www.independent.co.ug

Reason:

Functional faults & Overriding the Pilot

⇒ **Unsafe Software**

https://www.theguardian.com/technology/2022/

Binance, the world's largest cryptocurrency exchange, was the target of a hack that incurred possible losses of half a billion dollars (October 2022)

Reason:

Exploitable Vulnerability in the Implementation

⇒ **Unsecure Software**

In 2014 **56** Million Credit Cards were stolen from "The Home Depot (USA)"

Reason:

Malware infiltrated the IT-system

⇒ **Unsecure Software**

https://www.scmagazine.com

http://bilder1.n-tv.de

Knight Capital:

Computer-Trader
= high-frequency automated computer-trading

[10'000 Trades/sec
Holding: Milliseconds]

Computer-traded Loss on 1.8.2012 (NYSE): **440 Million US$** (in 20 minutes)

On 1.8.2012 at 9:30: The computers generated (without human activity) millions of ***faulty trades***

At 9:58 Knight Capital had lost **440 Million US$**

Reason: **Programming mistake** in the high-frequency automated trading algorithm after a software-update

https://www.mytechlogy.com

Attack on a water treatment plant (Safety accident)

https://www.chemistryworld.com

On February 5, 2021 a **water treatment plant** operator for the city Oldsmar on Florida's west coast saw his cursor being moved around on his computer screen, opening various software functions that **control** the water being treated

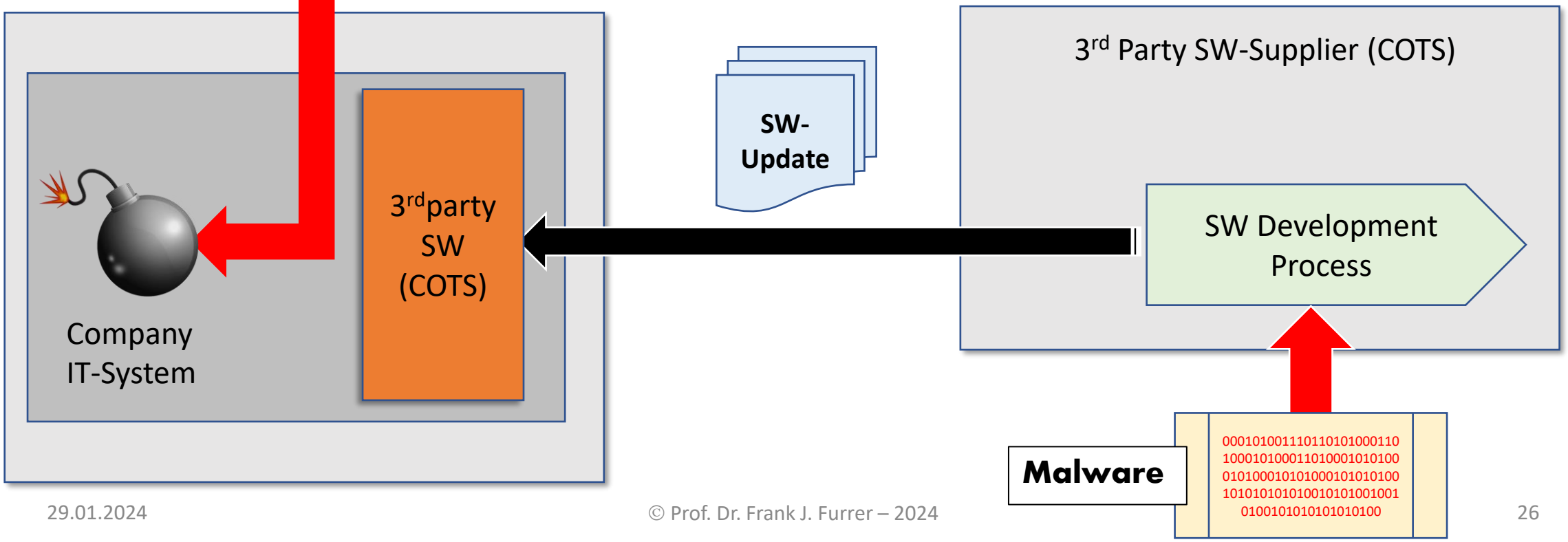
The cyber-intruder boosted the level of *sodium hydroxide* in the water supply to 100 times higher than normal.

Sodium hydroxide is used to control water acidity and remove metals from drinking water in treatment plants. **Sodium hydroxide poisoning can cause burns, vomiting, severe pain and bleeding**

Source: https://blogs.manageengine.com/corporate/manageengine/pam360/2021/02/17/cyberattack-on-floridas-water-treatment-plant-what-it-means-to-global-organizations.html

Supply Chain Attack (**Security** threat)

December 13, 2020: Malicious actors are currently exploiting SolarWinds Orion products. The Orion platform is a suite of products to *monitor* the health of IT networks (https://www.solarwinds.com). SolarWinds acknowledged that hackers had inserted malware into its *software update distribution* mechanism. This security incident resulted in malicious code being pushed to more than 16'000 customers (industry & government)



Korea JoongAng Daily, November 9, 2023:

Robotic arm kills worker after mistaking him for box of red peppers

https://www.thesun.co.uk/news/24677007/robot-kills-factory-worker-crushing-conveyor-belt/

A man in his 40's died after an industrial robotic arm **mistook** him for a box of red peppers.

According to the police, the accident occurred at 7:45 p.m. on Tuesday 7.11.23 during a **test** at an agricultural produce distribution center in Goseong, South Gyeongsang, Korea

An employee from a facility maintenance company was testing the sensor accuracy on the robotic arm

⇒ Unsafe Software

https://koreajoongangdaily.joins.com/news/2023-11-09/national/socialAffairs/Robotic-arm-kills-man-in-industrial-accident-in-South-Gyeongsang/1909597

Prof. Dr. Frank J. Furrer

Software Development:
From Black Art to Engineering Science

Technology Maturity

Safety Accidents

Security Incidents

Software-induced
- SafetyAccidents
- Security Incidents

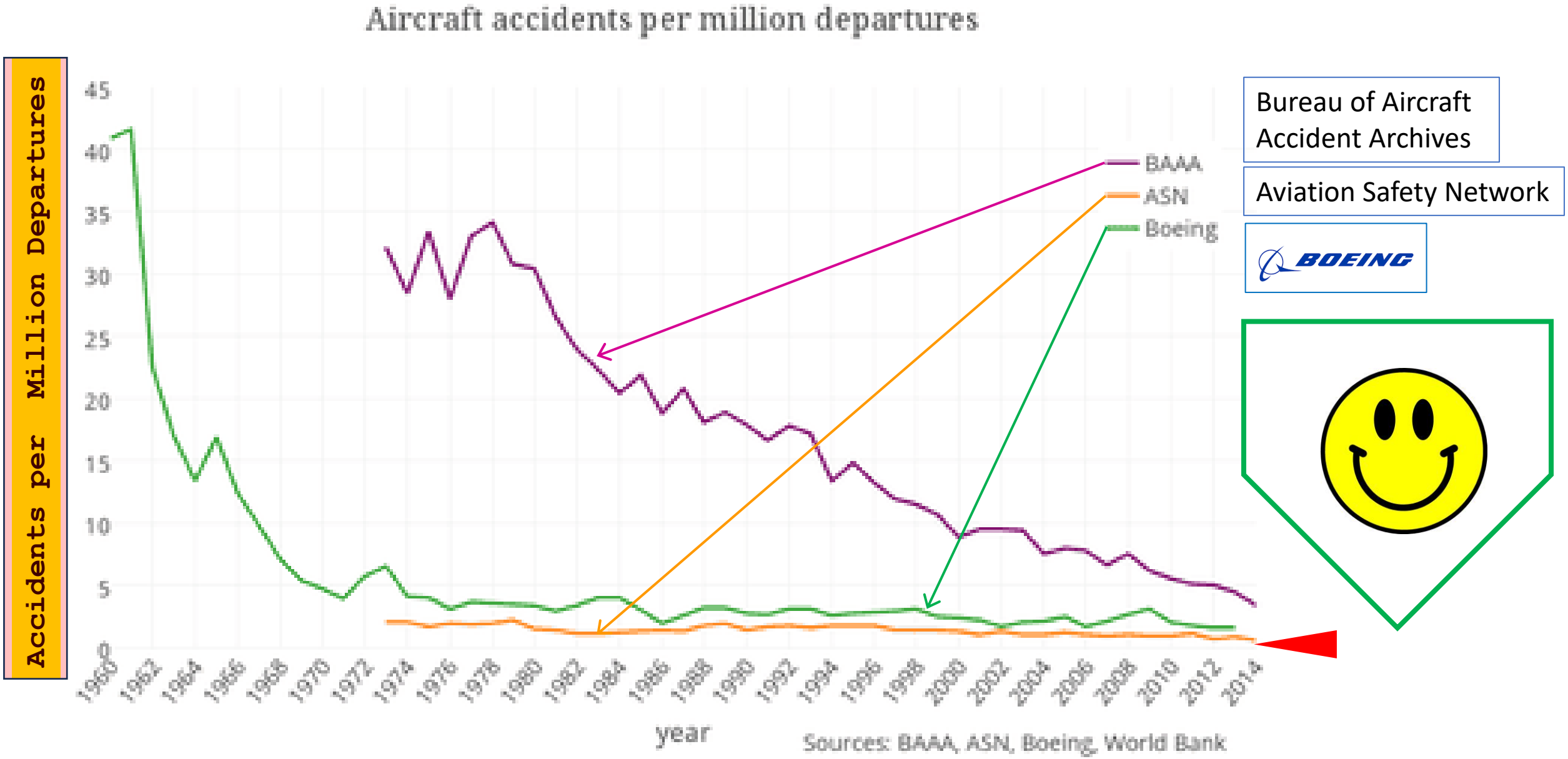
⇒ **Zero**

Low **Software Technology Maturity** **High**

Technology Maturity

Example: **Aircraft Accidents**

https://www.timesnownews.com/auto/nepal-plane-crash-72-seater-passenger-aircraft-crashes-at-pokhara-international-airport-article-97002941



Bureau of Aircraft Accident Archives

Aviation Safety Network

BOEING

https://www.theguardian.com/world/datablog/2014/dec/29/aircraft-accident-rates-at-historic-low-despite-high-profile-plane-crashes

Accident/Incident Rate

high

low

SYSTEM COMPLEXITY

Safety Accidents

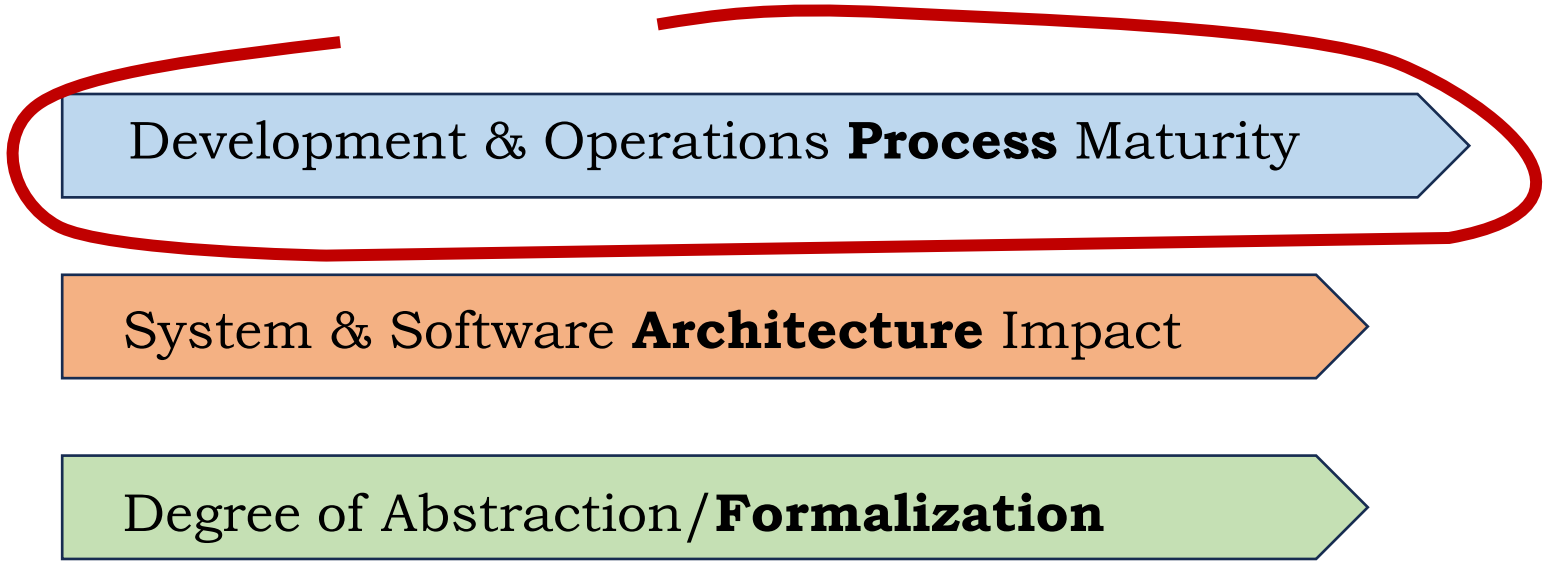
Security Incidents

10^6 10^7 10^8 10^9 10^{10}

Software-System: Million Lines of Source Code (SLOC)

Prof. Dr. Frank J. Furrer

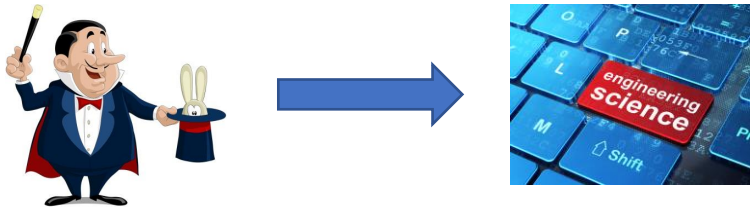
Software Development:
From Black Art to Engineering Science



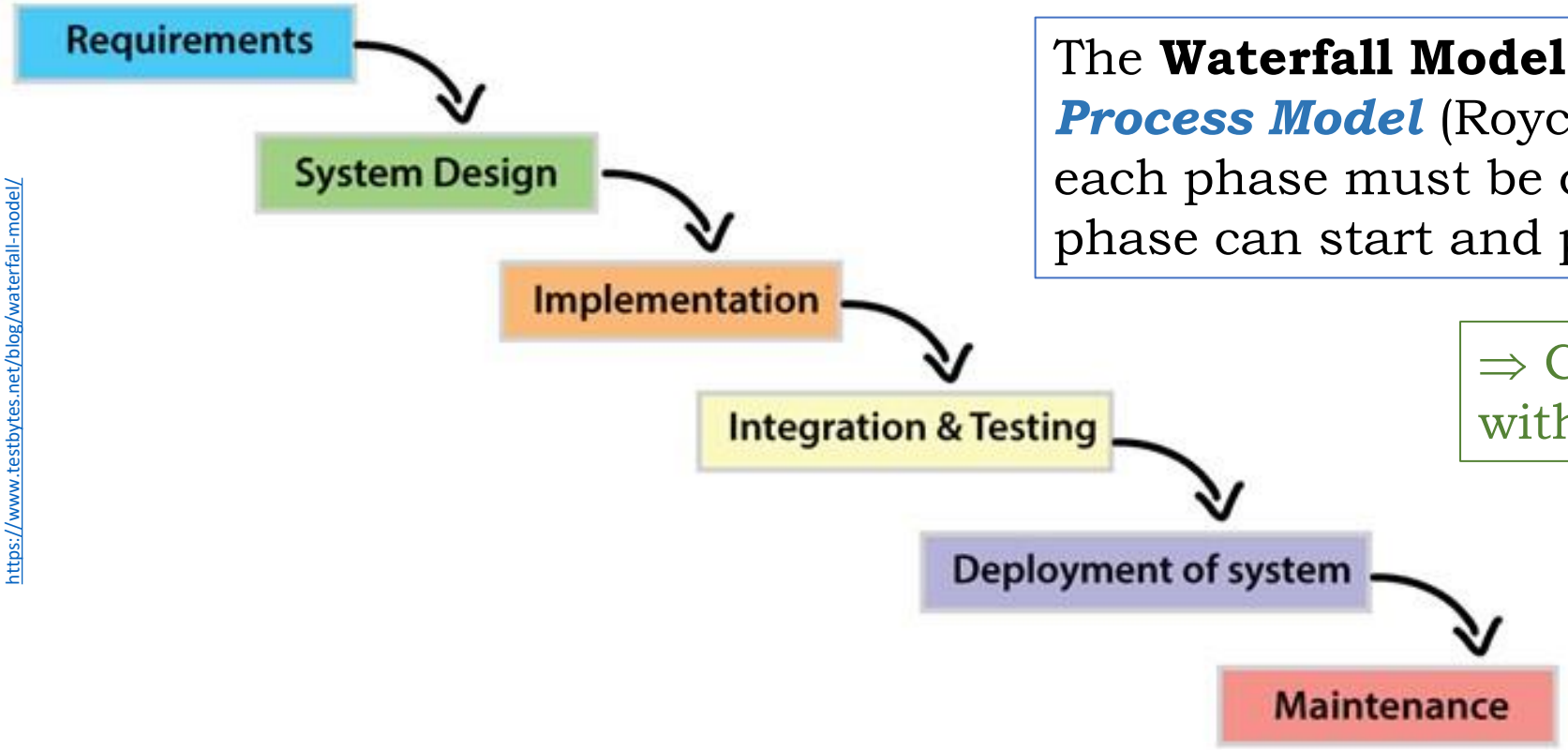
1

low — Development & Operations **Process** Maturity — high

https://www.canstockphoto.com/3d-man-software-development-analysis-15143918.html



Waterfall Model



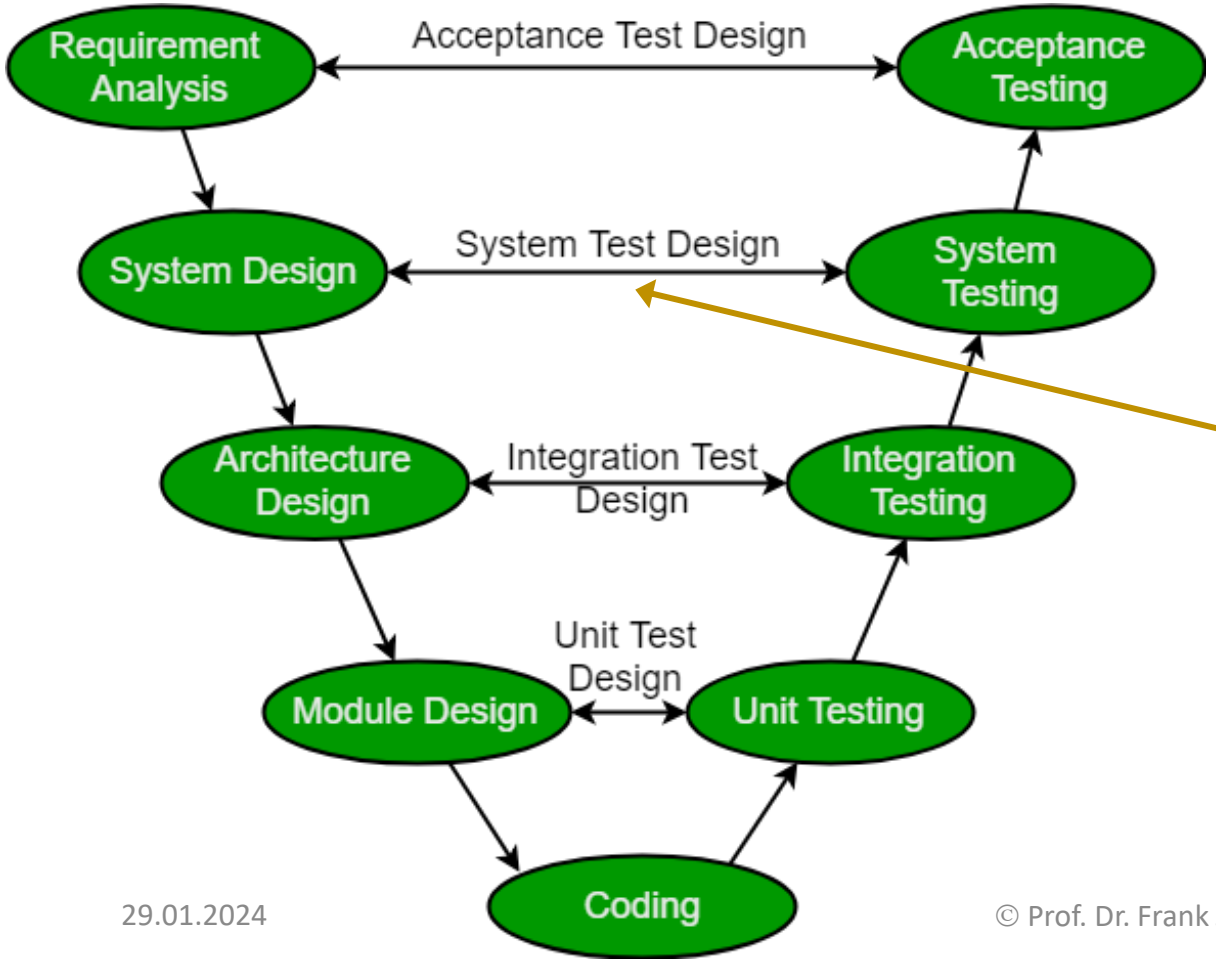
https://www.testbytes.net/blog/waterfall-model/

The **Waterfall Model** is the first SW-Development ***Process Model*** (Royce, 1970). In the waterfall model, each phase must be completed before the next phase can start and phases never overlap in time.

⇒ Generations of projects executed with the waterfall model!

Difficulties: Sequential Model
- **Long** development cycles
- **Changes** during development very difficult

V-Model



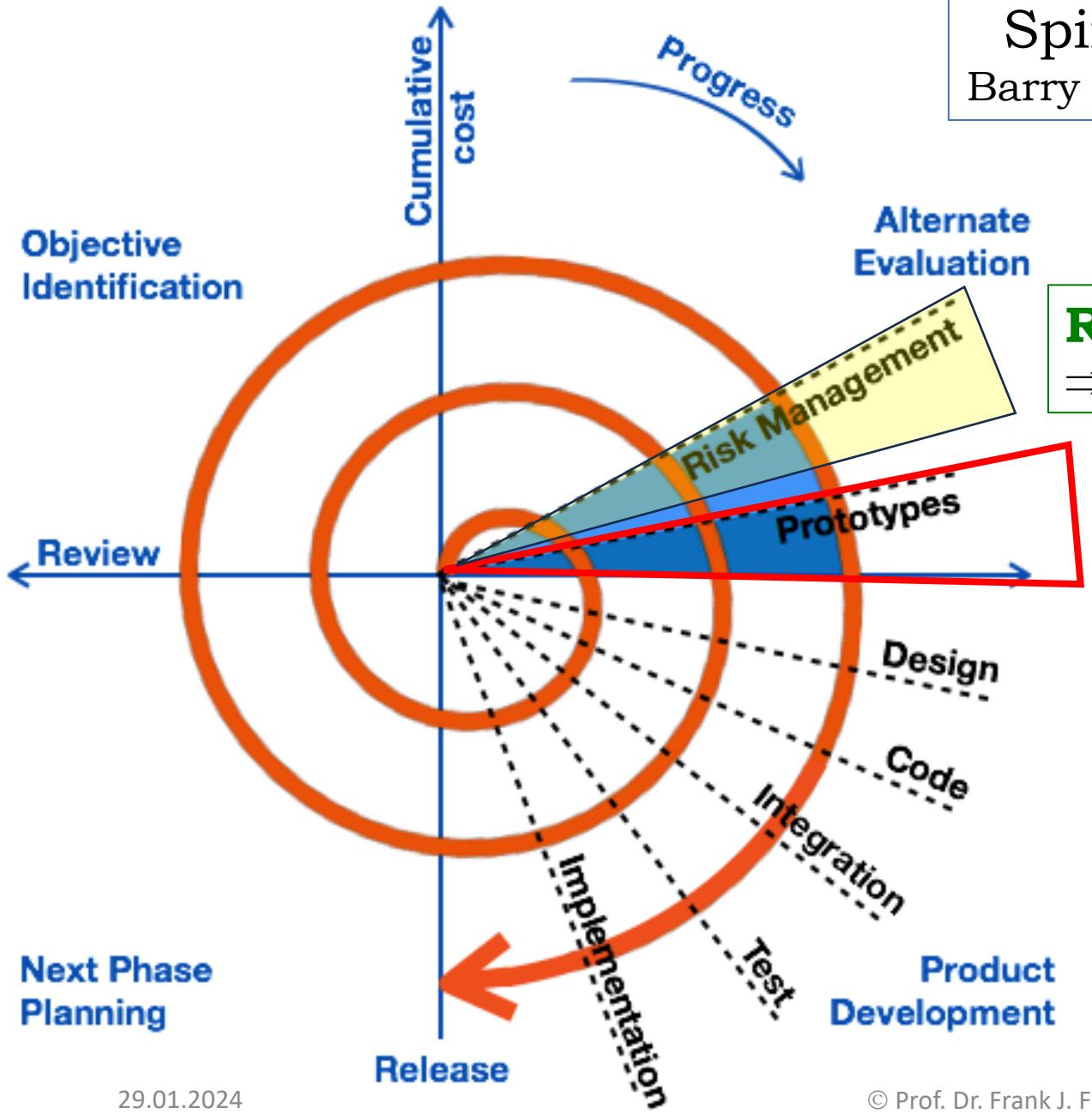
The **V-model** first appeared at Hughes Aircraft circa 1982

The V-model is a ***process model*** where process steps follow in a sequential order. It is also known as Verification and Validation model. It is based on the association of a **testing phase** for each corresponding development stage.

Difficulties: Sequential Model
- **Long** development cycles
- **Changes** during development very difficult

https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/

Spiral Model
Barry Boehm, 1986



https://www.tutorialspoint.com/sdlc/sdlc_spiral_model.htm

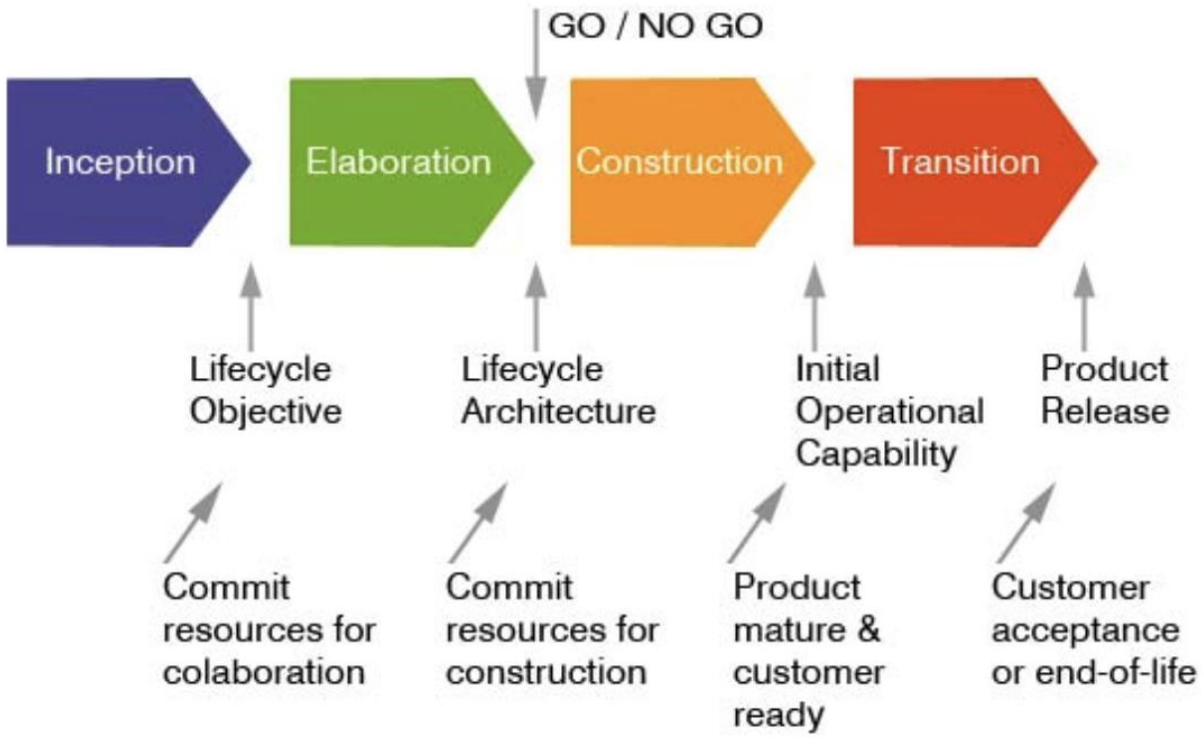
Dependable, trustworthy software

Risk Management
⇒ Integrated Part of the Development Process

Continuous Evolution, including:
- ✓ Architecture
- ✓ Design Documentation
- ✓ Manuals
- ✓ etc.

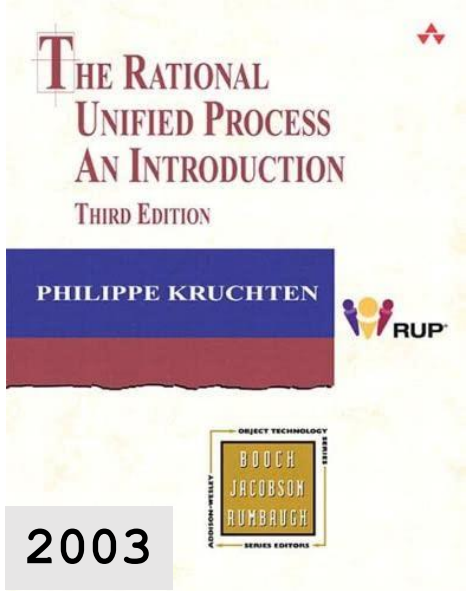
Rational Unified Process (RUP)

Rational Unified Process (RUP)



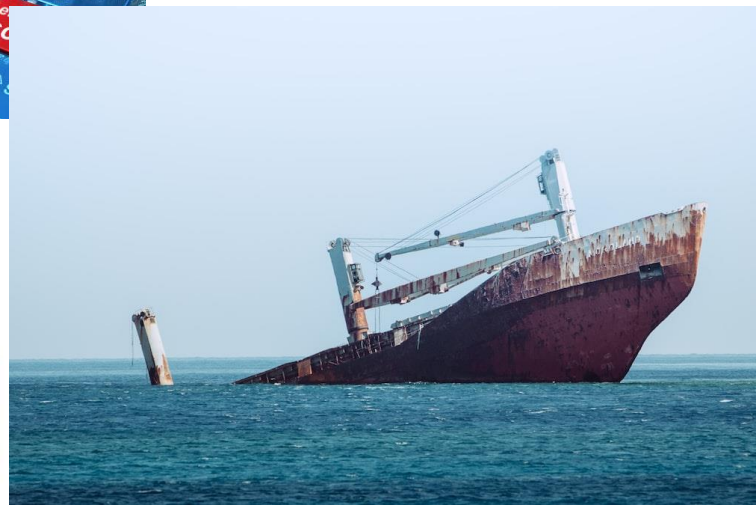
1999:
Developed by Grady Booch, Ivar Jacobson and James Rumbaugh („The three Amigos“) at that time with Rational Software, later IBM

The first development process using **modeling** (UML)



1991: Requirement Capability Maturity Model Integration (CMMI)

https://unsplash.com/de/s/fotos/sinking-ship



Characteristics of the Maturity Levels

- Level 5 **Optimized** — Focus on process improvement.
- Level 4 **Controlled** — Processes measured and controlled.
- Level 3 **Defined** — Processes characterized for the organization and is proactive. (Projects tailor their processes from organization's standards)
- Level 2 **Managed** — Process characterized for projects and is often reactive.
- Level 1 **Initial** — Processes unpredictable, poorly controlled and reactive.

https://www.istockphoto.com/

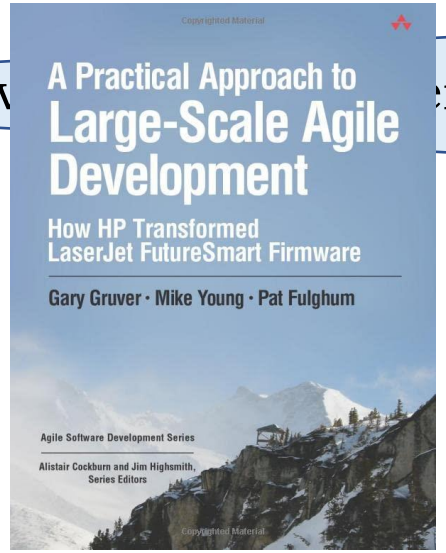
https://www.indiamart.com

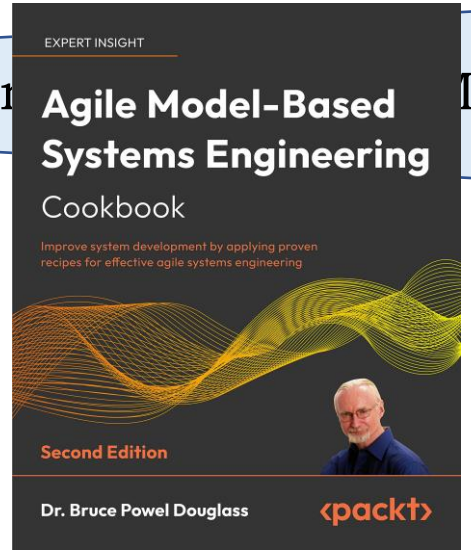
https://teamibr.com/capabilities/software-engineering/cmme-level-3-experienced

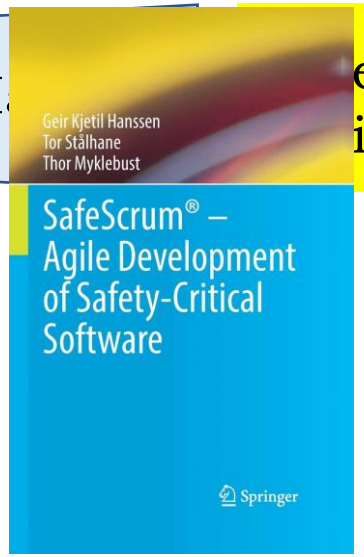
2001: Agile Methods

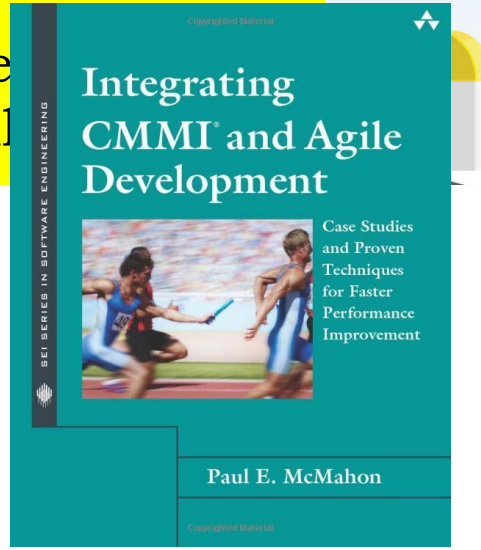
Dev … e … il …

https://kanbanzone.com/resources/agile/





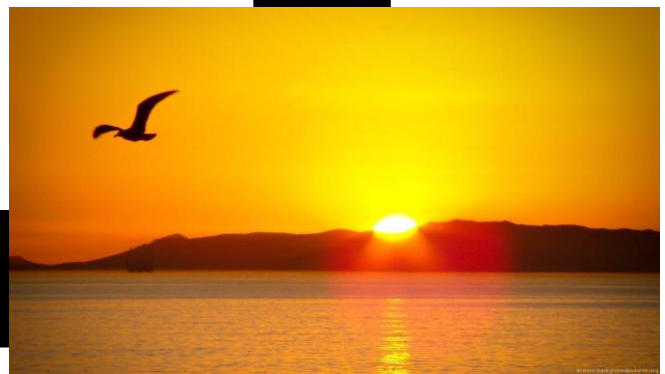




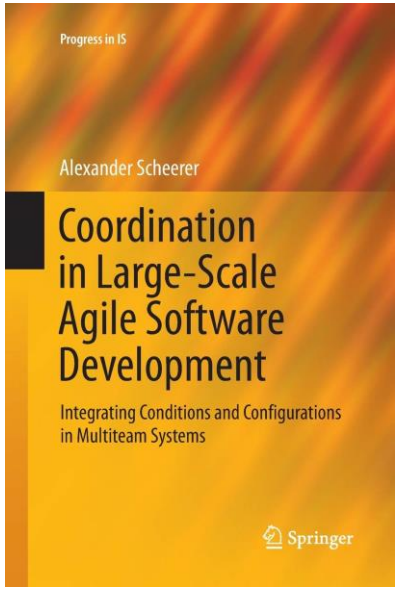
Fallback into the Dark Age of Software

https://www.swr.de/wissen/

– 2024



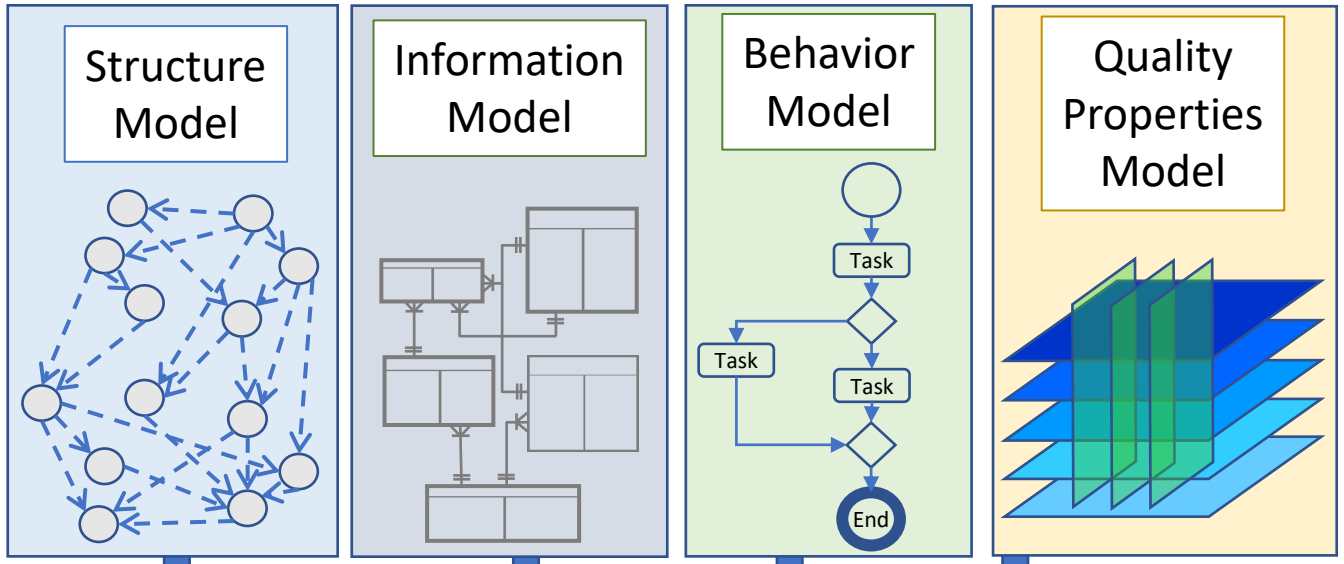
https://wallpapercave.com



…

Today: Model & Principle Based Engineering

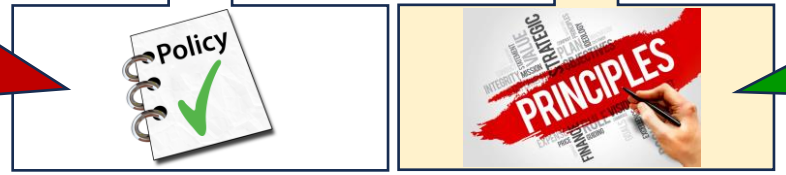
Formal System Model



Development Process & Operations

Lecture Part 2: Principle-Based Engineering

Mandatory Rules to comply with company goals, legal & compliance requirements and ethical standards



Policies **Principles**

Construction rules to develop, implement, and operate dependable, future-proof systems

Development Process Limits

(i) It is **impossible** to find all ***design faults*** in a large hardware/software system that may contain millions of lines of software code.

(ii) It is **impossible** to find all ***implementation faults*** in a large hardware/software system that may contain millions of lines of software code.

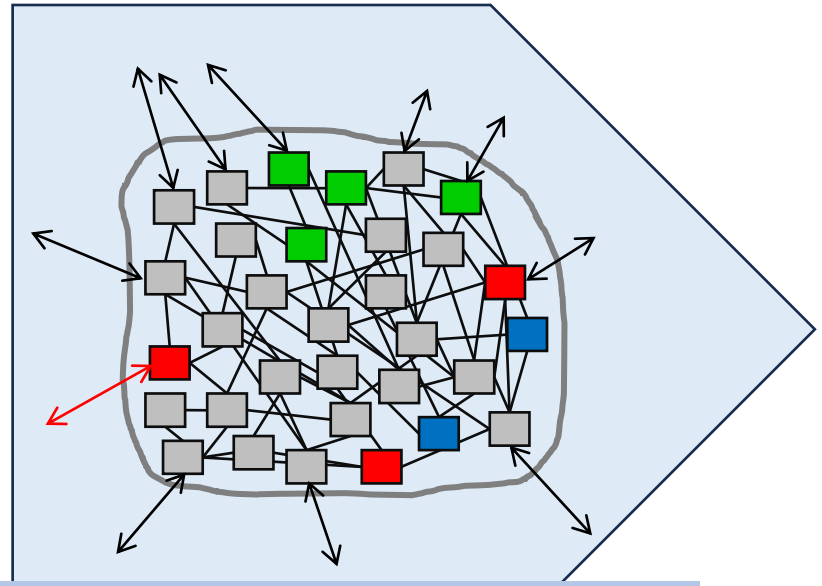
(iii) It is **impossible** to foresee all ***operational conditions*** of the system in its real environment (especially off-nominal conditions).

(iv) It is **impossible** to mitigate all consequences of ***execution platform failures*** (Hardware failures, network outages, support systems non-availability [such as GPS or car-to-car comm, 5&6G]).

(v) It is **impossible** to fully identify emergent properties or ***emergent behaviour*** while assembling the system (especially negative emergence in large systems-of-systems)

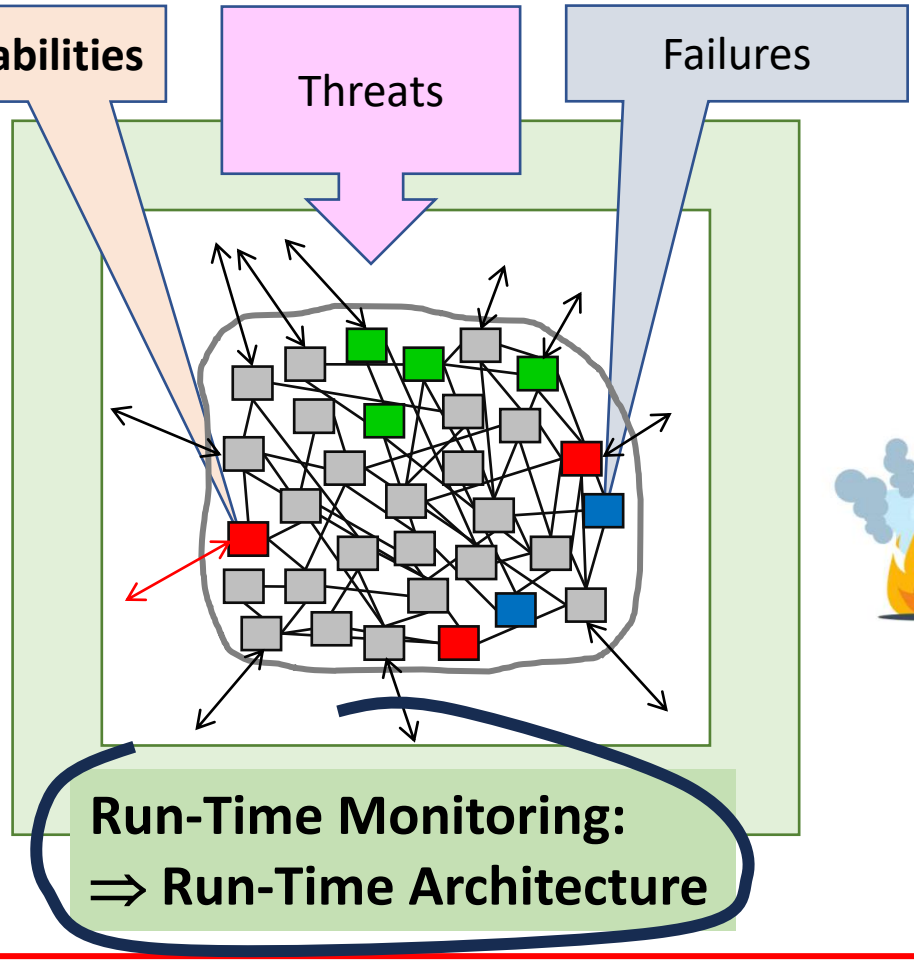
Hermann Kopetz, 2023

… therefore:



Development Process
⇒ Design-Time Architecture

Deployment



Run-Time Monitoring:
⇒ Run-Time Architecture

Design-Time Architecture:

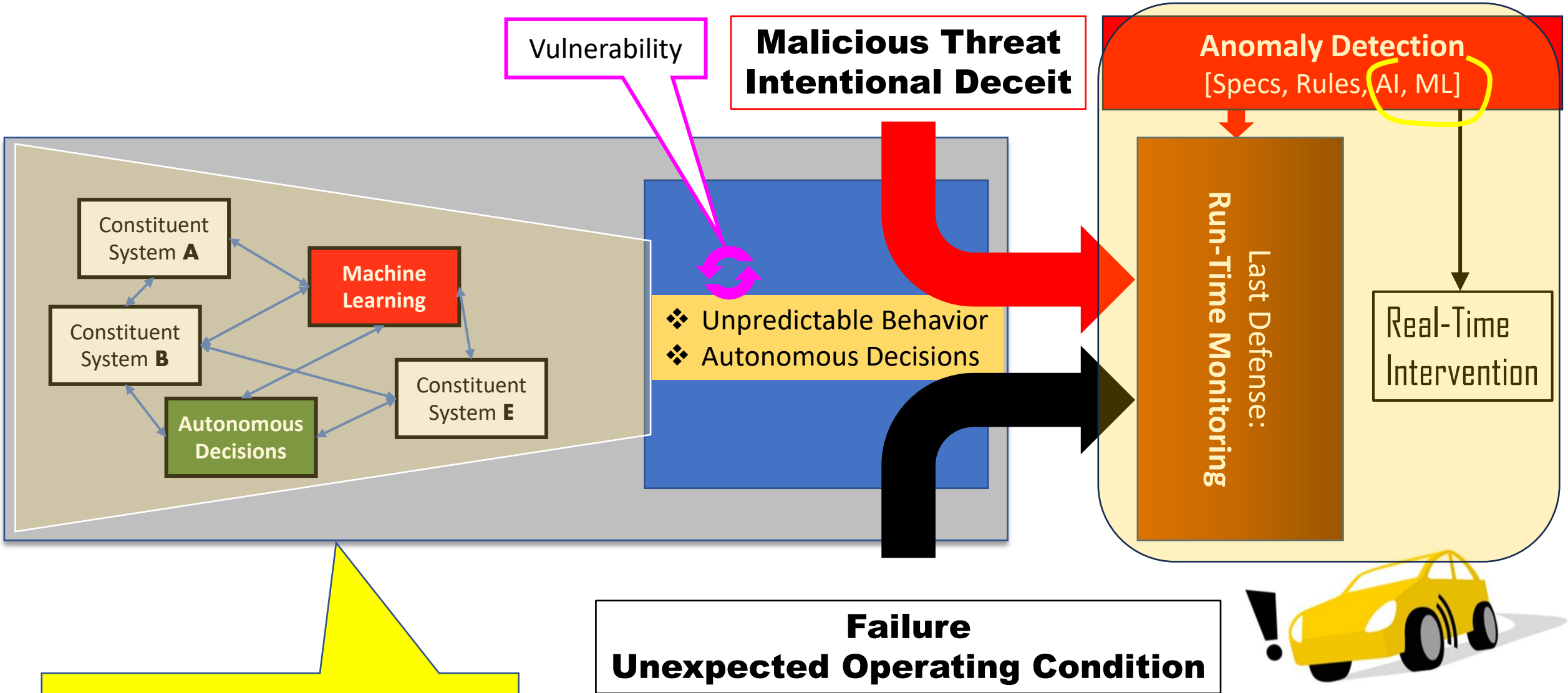
Detect, identify, analyze, and mitigate as many vulnerabilities as possible

Run-Time Architecture:

Prevent the consequences of undetected, unexpected or emerging vulnerabilities

https://www.istockphoto.com

⇒ We need a last line of defense: ***Runtime monitoring*** and ***intervention***



https://verkehrshelden.com/programm/achtung-auto

Process: Vision

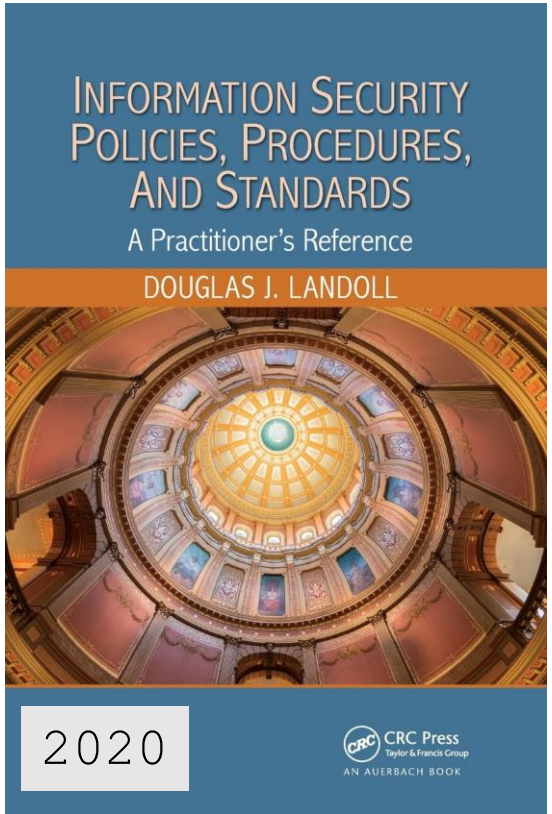
https://www.istockphoto.com/de/vektor/

Development Process & Operations

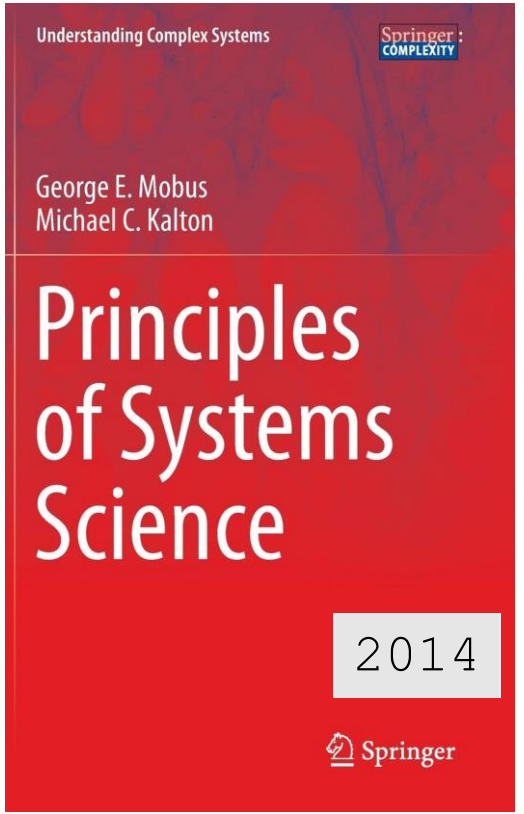
Mandatory Rules to comply with company goals, legal & compliance requirements and ethical standards

Policies **Principles**

Construction rules to develop, implement, and operate dependable, future-proof systems



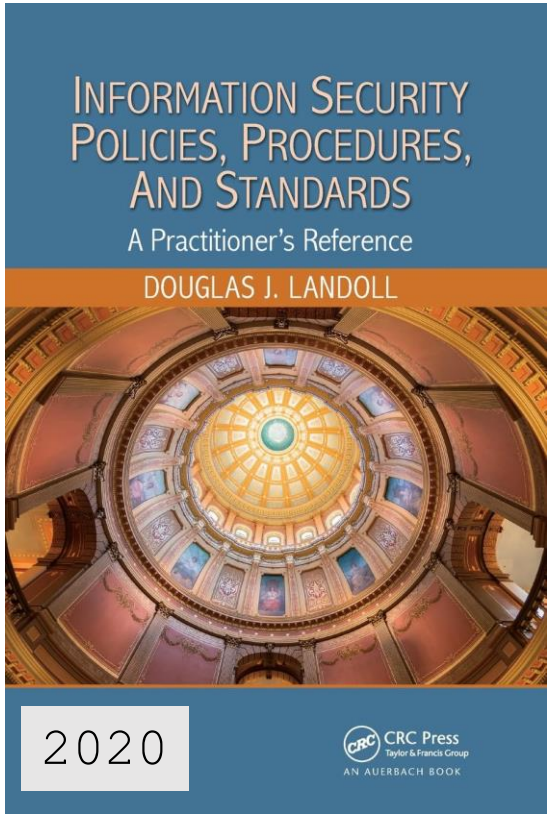
https://www.indiafilings.com/learn/corporate-governance/

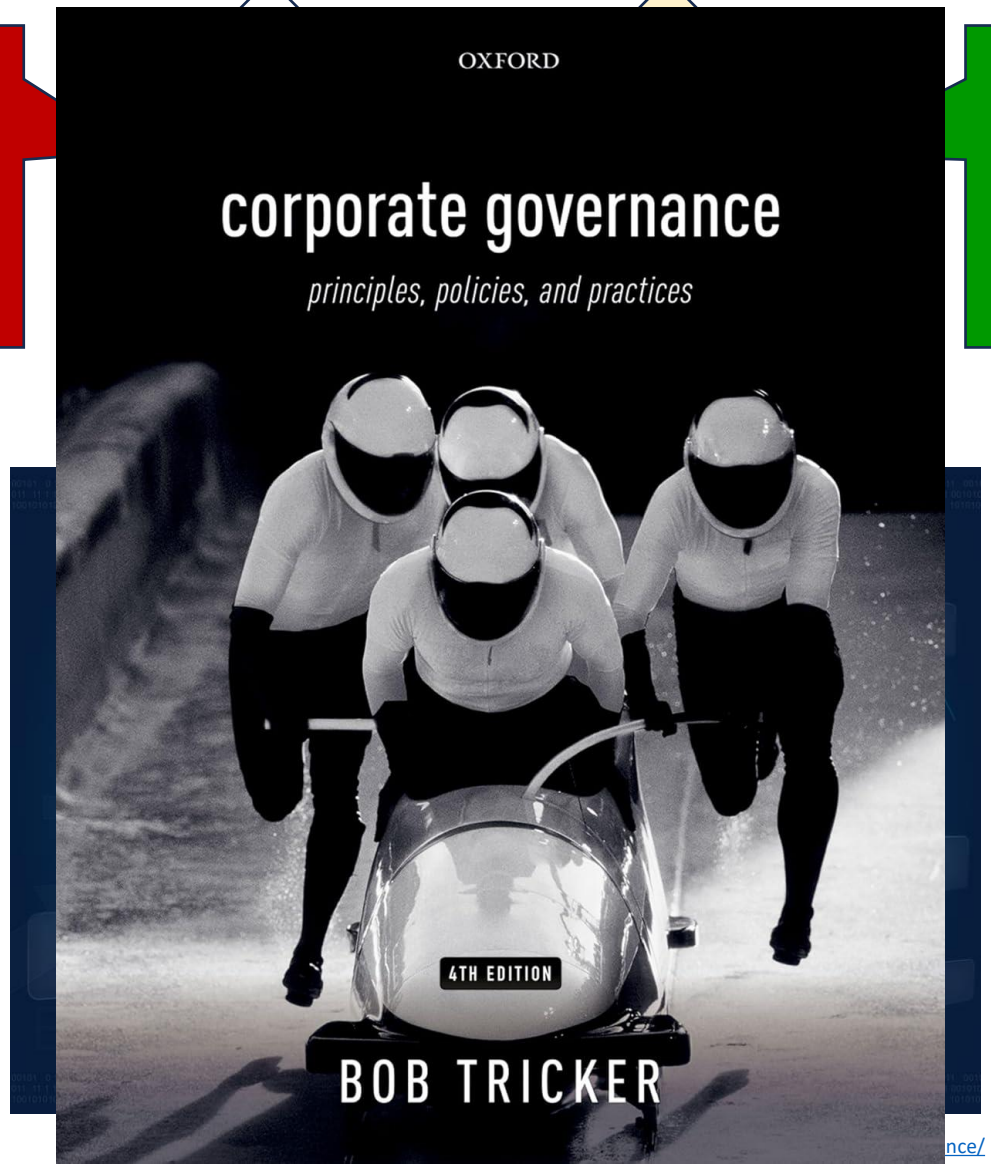


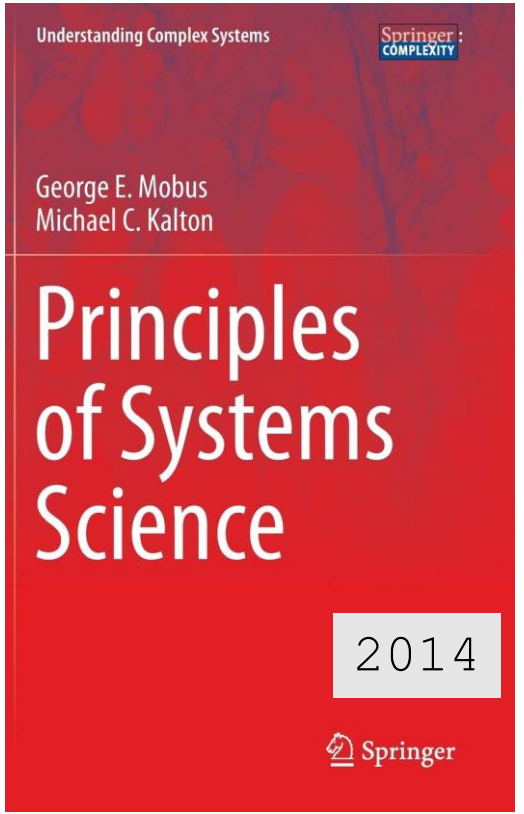
Development Process & Operations

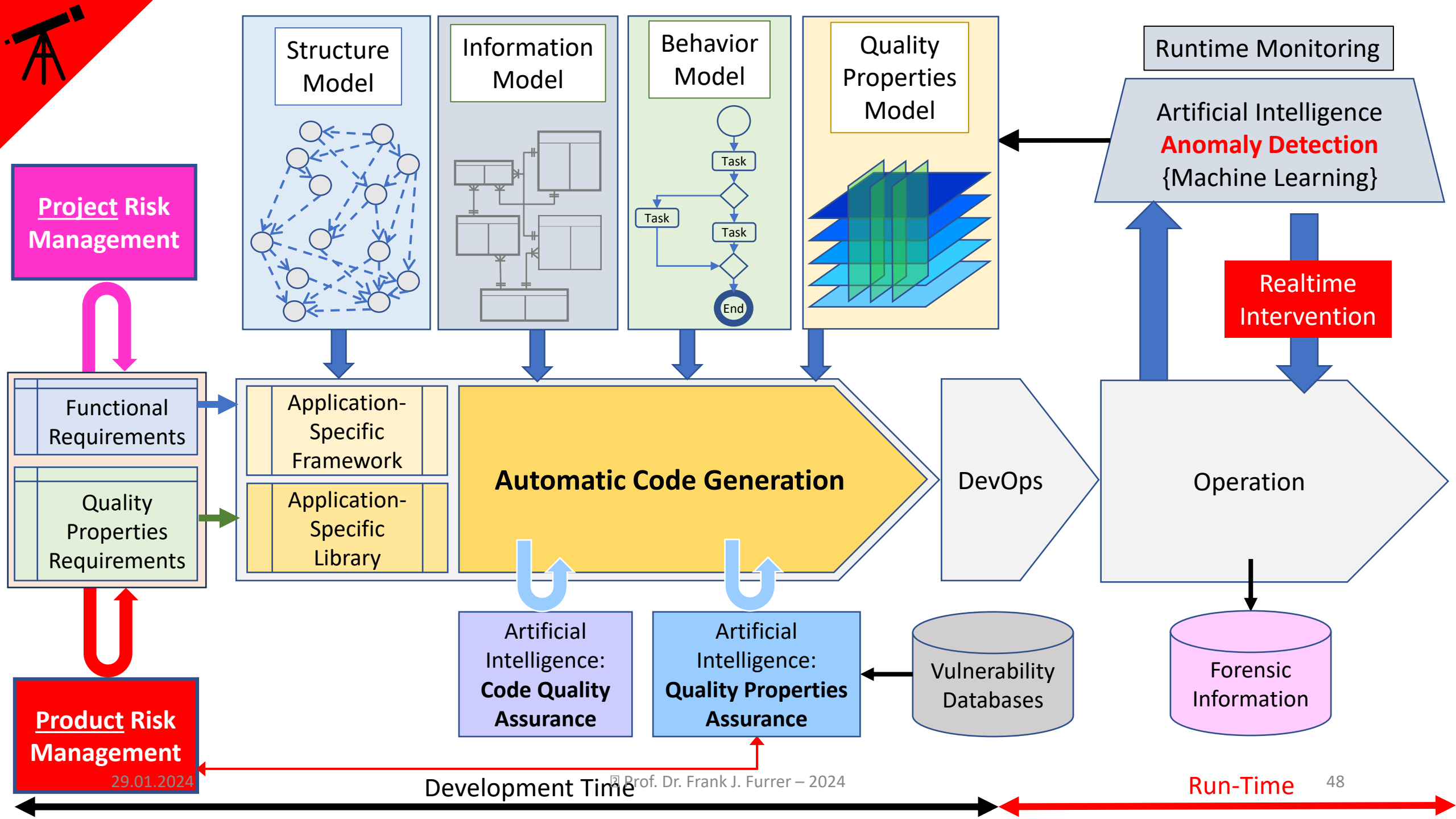
Mandatory Rules to comply with company goals, legal & compliance requirements and ethical standards

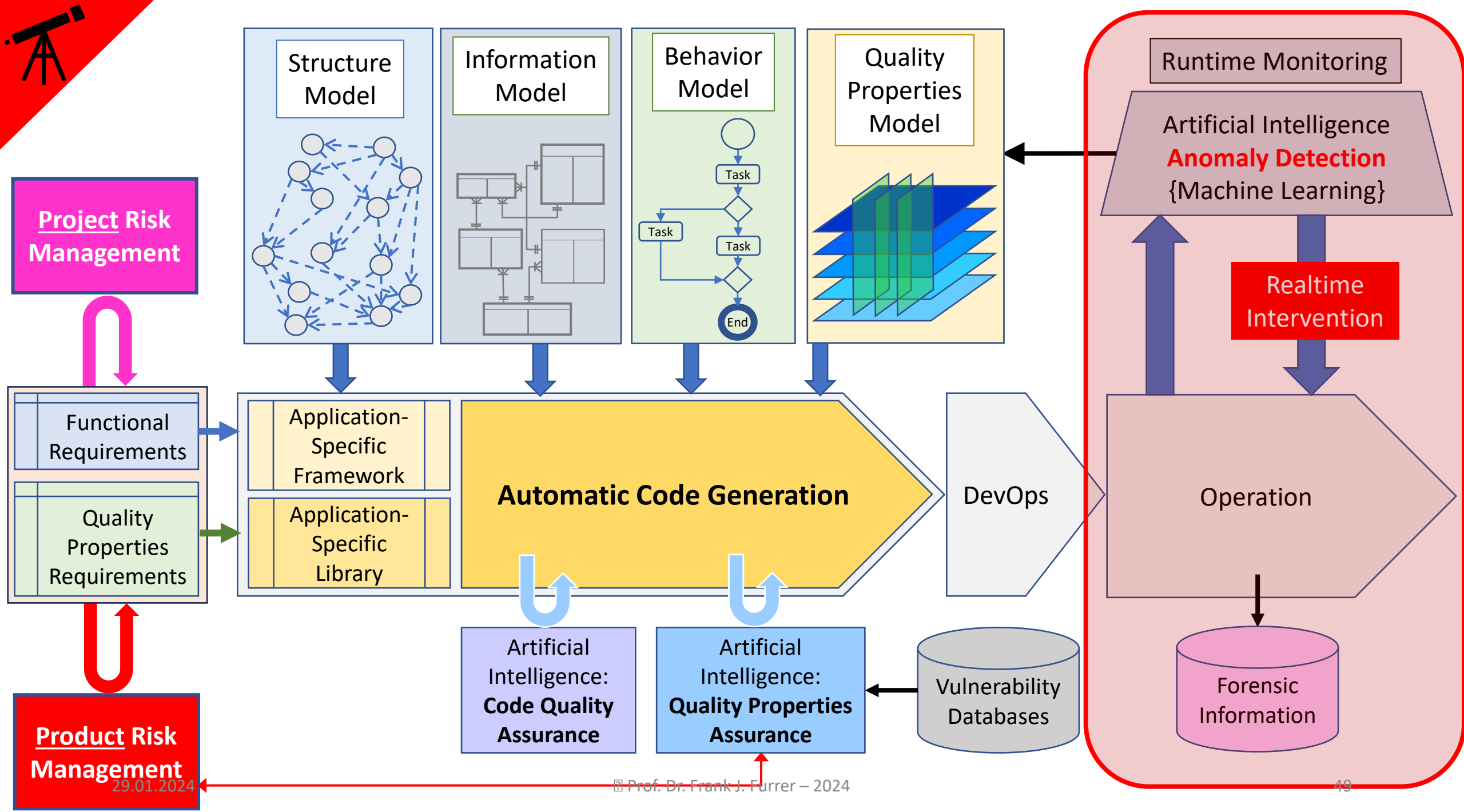
Construction rules to develop, implement, and operate dependable, future-proof systems





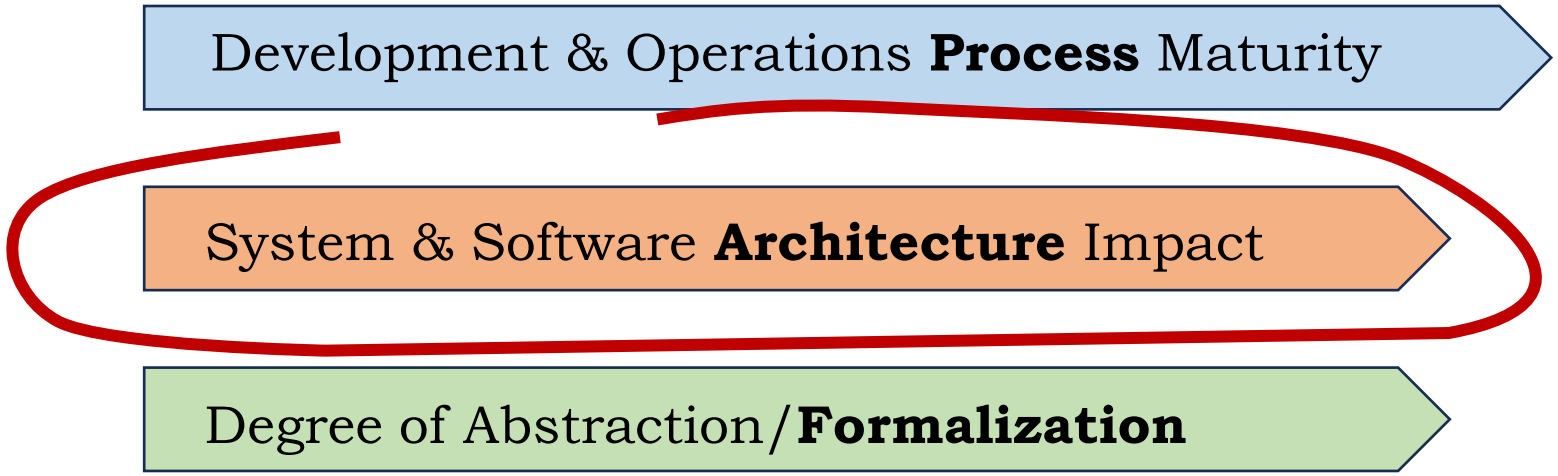



Structure Model
Information Model
Behavior Model
Task
Task
Task
End
Quality Properties Model
Runtime Monitoring
Artificial Intelligence
Anomaly Detection
{Machine Learning}
Project Risk Management
Realtime Intervention
Functional Requirements
Quality Properties Requirements
Application-Specific Framework
Application-Specific Library
Automatic Code Generation
DevOps
Operation
Artificial Intelligence: Code Quality Assurance
Artificial Intelligence: Quality Properties Assurance
Vulnerability Databases
Forensic Information
Product Risk Management
Development Time
Run-Time

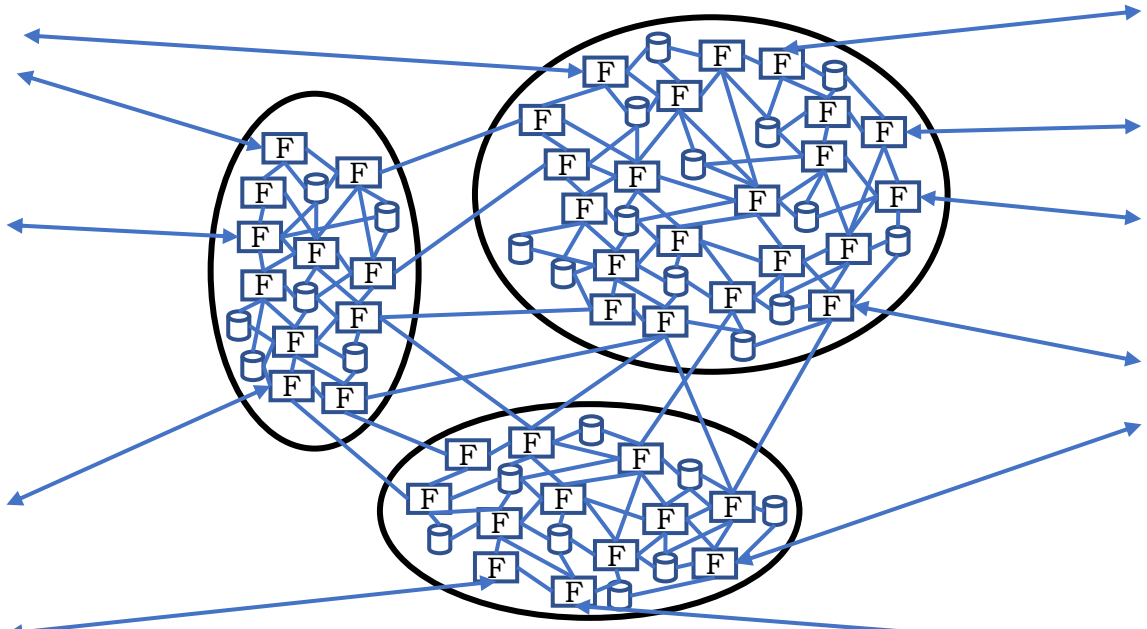

Structure Model
Information Model
Behavior Model
Task
Task
Task
End
Quality Properties Model
Runtime Monitoring
Artificial Intelligence
Anomaly Detection
{Machine Learning}
Realtime Intervention
Project Risk Management
Functional Requirements
Quality Properties Requirements
Application-Specific Framework
Application-Specific Library
Automatic Code Generation
DevOps
Operation
Artificial Intelligence: Code Quality Assurance
Artificial Intelligence: Quality Properties Assurance
Vulnerability Databases
Forensic Information
Product Risk Management

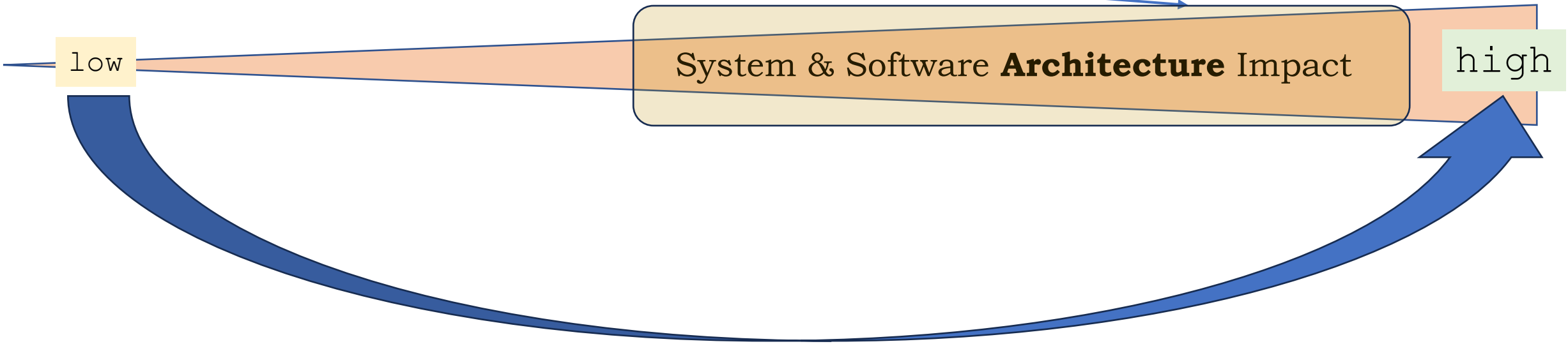
Prof. Dr. Frank J. Furrer

Software Development:
From Black Art to Engineering Science



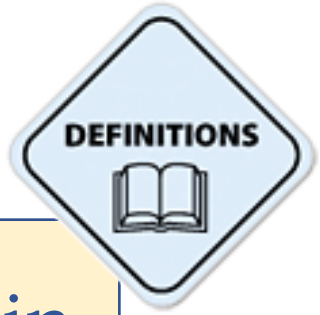
2





https://quotefancy.com/quote/1740727/Brian-Foote-If-you-think-good-architecture-is-expensive-try-bad-architecture

Architecture Fundamentals: **Definition**



Structure

Behaviour

Elements

Principles

"The fundamental *organization* of a system embodied in its *parts*, their *relationships* to each other and to the environment, and the *principles* guiding its design and evolution"

[adapted from IEEE00]

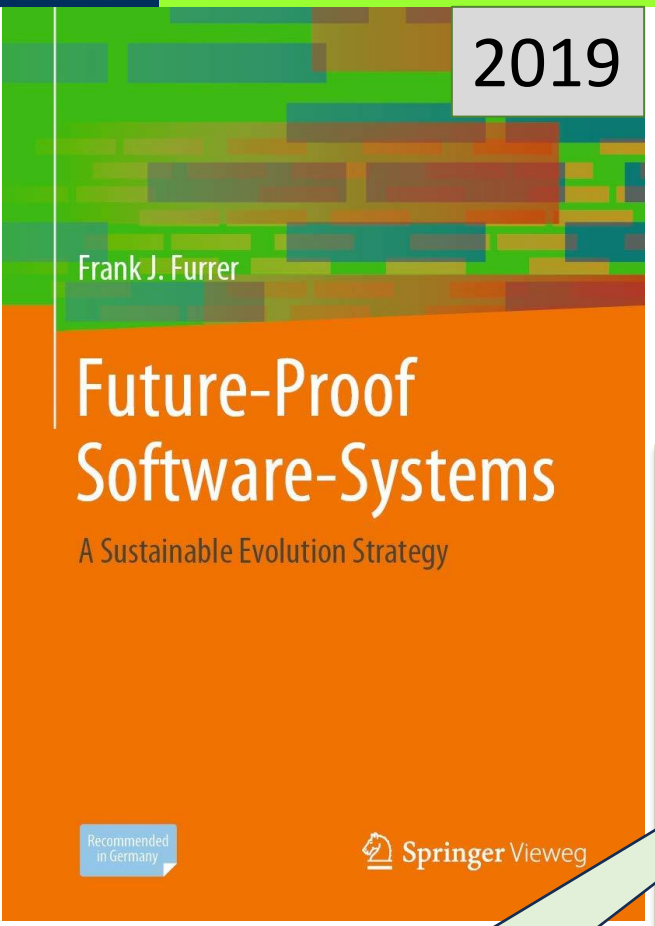
Architecture Fundamentals: **Good Architecture**

?

http://www.0111.com/architecture-exhibitions/?gal=24

http://www.asisbiz.com/index.html

What is good Architecture ?



What is good Architecture ?

Central Question of **successful** Software Engineering

A sustainable architecture which enables and ensures the **system's** quality properties, such as:

- *Business value*
- *Changeability*
- *Scalability*
- *Safety*
- *Security*
- *etc.*

over the full lifetime of the system

https://www.educatinguk.com/what-is-internal-quality-assurance/

Stakeholder [User] View

Stakeholder [User] View

Metrics

Architecture Properties

A sustainable architecture which enables and ensures the **system's** quality properties, such as:

- *Business value*
- *Changeability*
- *Scalability*
- *Safety*
- *Security*
- *etc.*

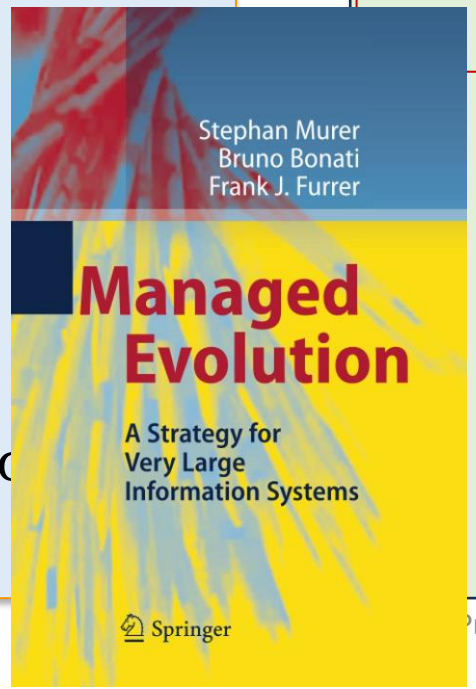
over the full lifetime of the system

Number of Function Points

$$\frac{\#FP^2}{TtM * DevC}$$

Time-to-Market [days]

Development Cost [CHF]



Proven Qualities of **good Architecture**:

- ✓ *Partitioning*
- ✓ *Encapsulation*
- ✓ *Information Hiding*
- ✓ *Interfaces*
- ✓ *Coherence & Cohesion*
- ✓ *Separation of Concerns*
- ✓ *etc.*

Functionality
- **Business Processes**
- **Administration**
- **…**

System Quality Properties
- **Safety**
- **Security**
- **…**

MOST IMPORTANT

A good, well-maintained

System-Architecture

forms the

Indispensable Foundation

not only for the

Functionality,

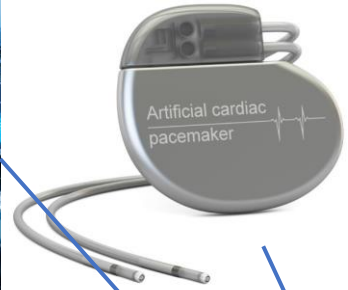
but also for the

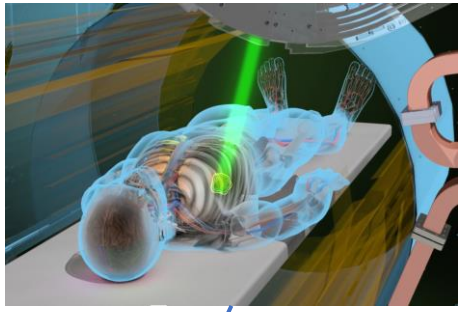
Quality Properties

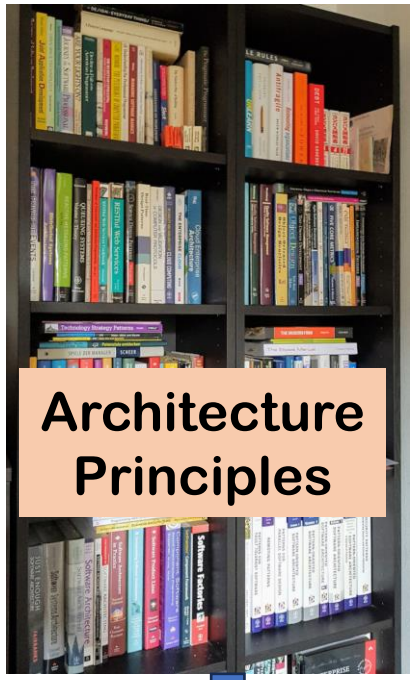
of the System

https://www.dreamstime.com/photos-images/most-important.html

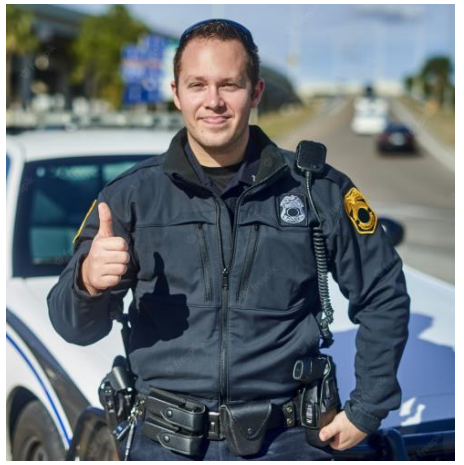
How do we build, maintain, and evolve Good Architecture?







Architecture Principles



Architecture Governance

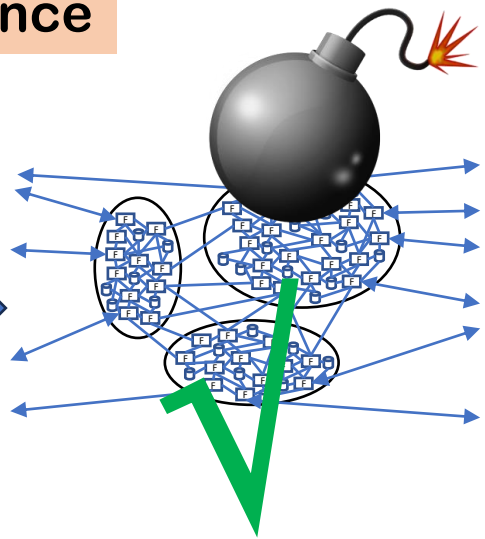
A sustainable architecture which enables and ensures the **system's** quality properties, such as:

- *Business value*
- *Changeability*
- *Scalability*
- *Safety*
- *Security*
- *etc.*

over the full lifetime of the system

Select, specify, and prioritize the **Quality Properties**

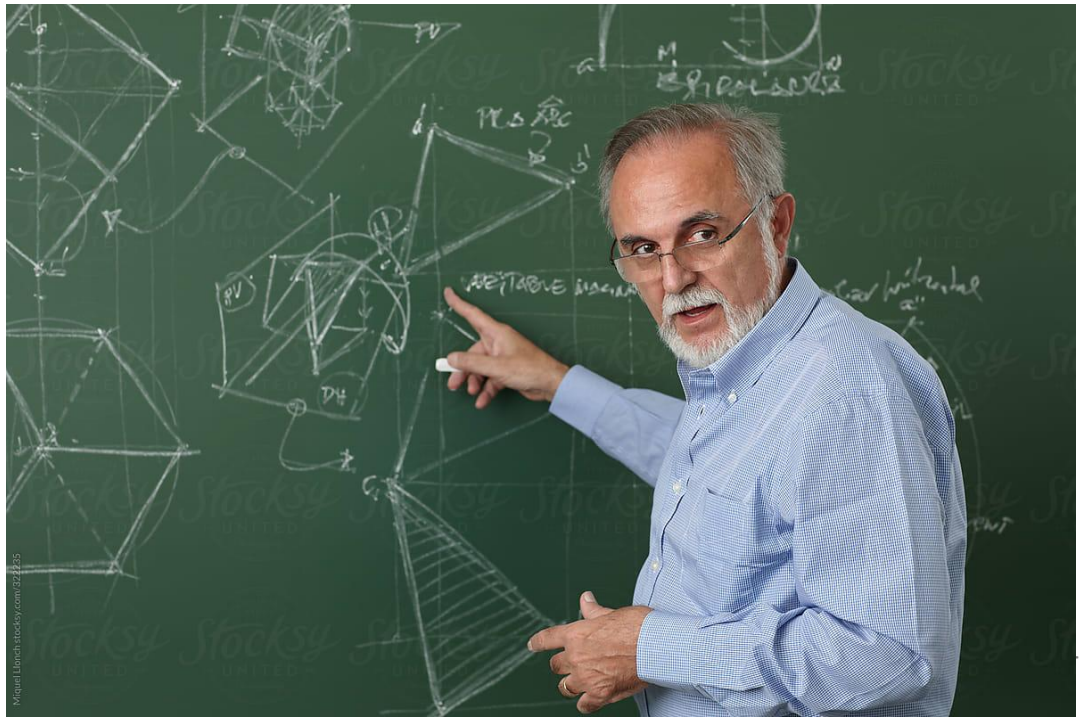
Development/Evolution

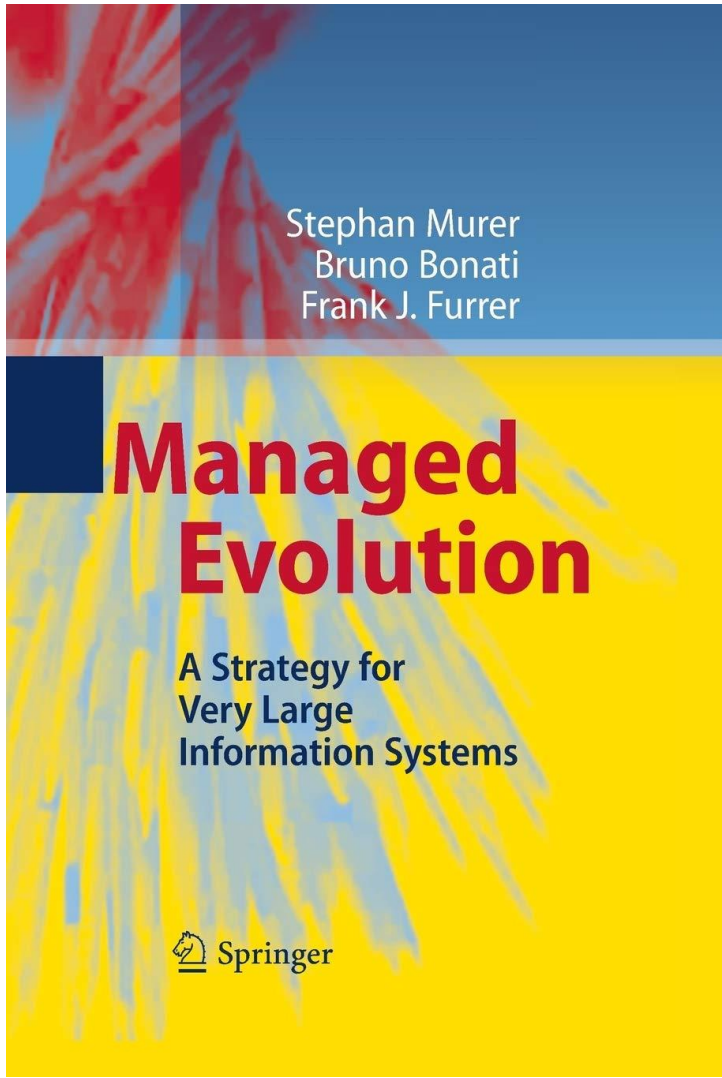


https://creazilla.com/nodes/64368-bomb-cipart

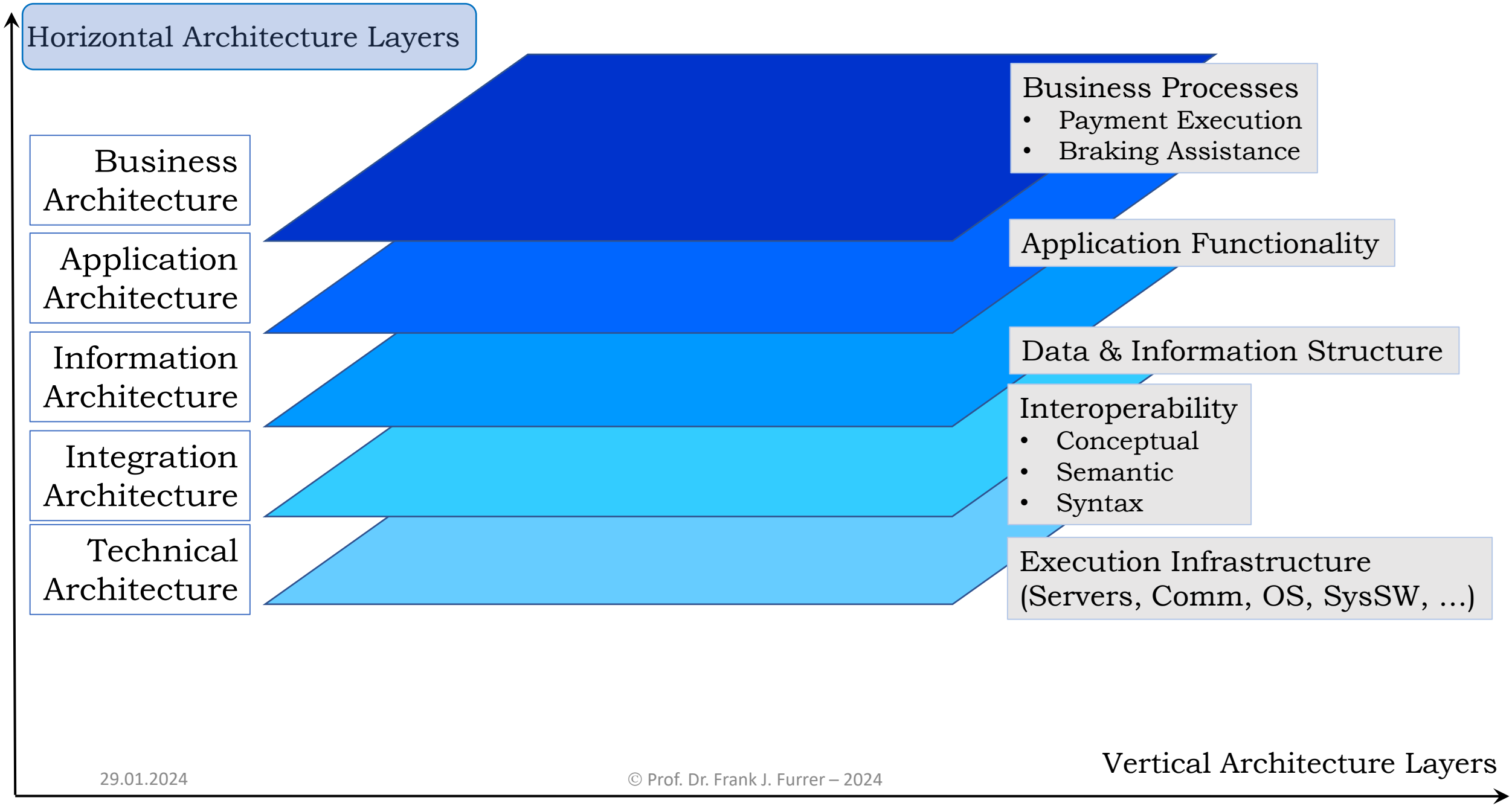
When thinking and talking about System Architecture

… we need an **Architecture Framework** to organize our Thoughts





https://www.stocksy.com/322235/senior-professor-teaching-in-front-of-the-blackboard



The strongest <u>responsibility</u> for all stakeholders is to build and operate **safe** and **secure** software

https://www.johnfoy.com

Software should never **harm** or **hurt** people, animals, property or the environment

https://www.ndr.de/ratgeber

Therefore:

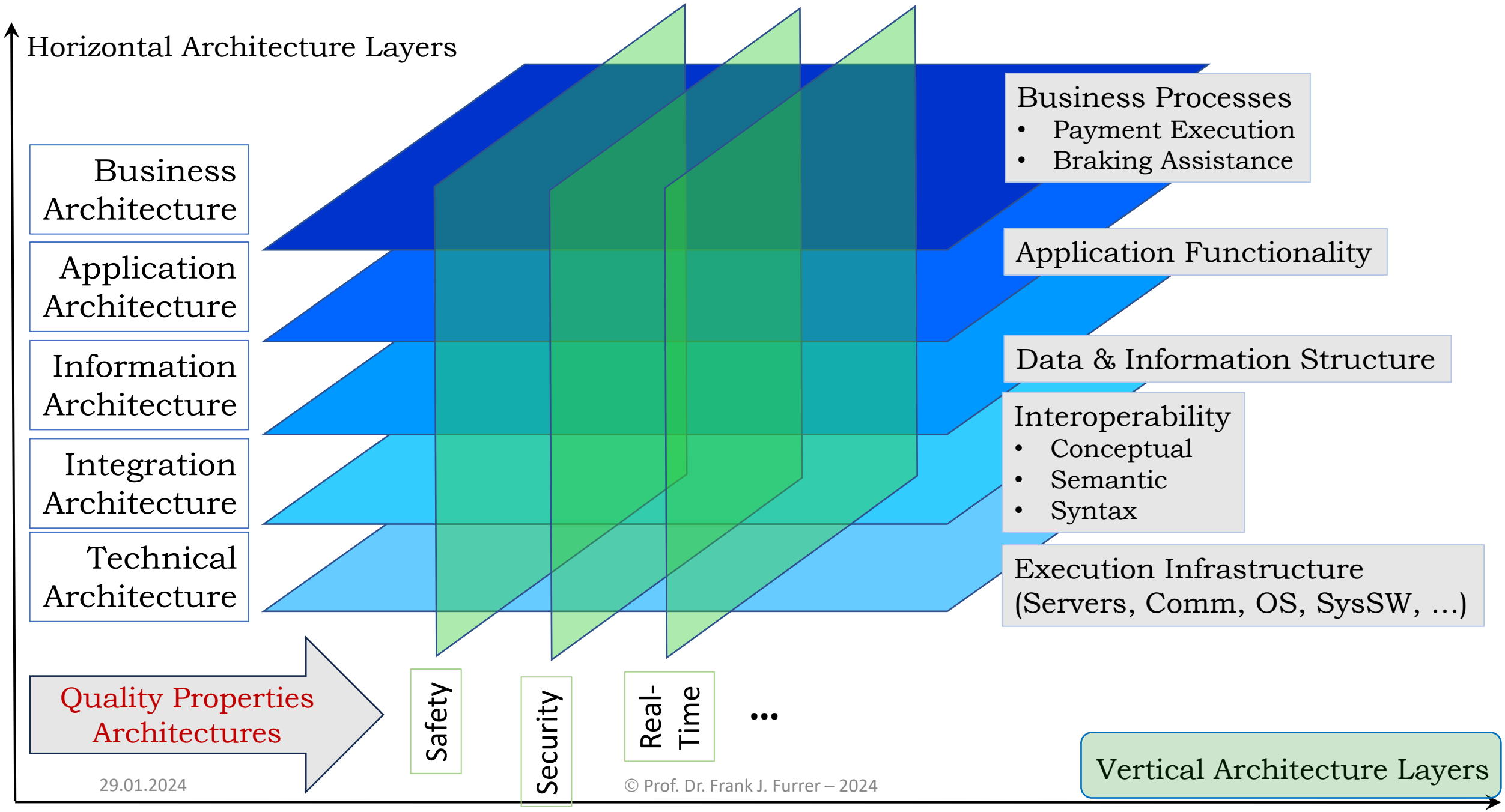
We need <u>specific, strong architectures</u> for the **quality properties** of the software:

https://www.staylegal.net/risk-assessment-guide/

- ✓ Safety architecture
- ✓ Security archiecture
- ✓ …

https://medium.com/quick-code

https://stock.adobe.com/fr/search?k=responsibility+icon



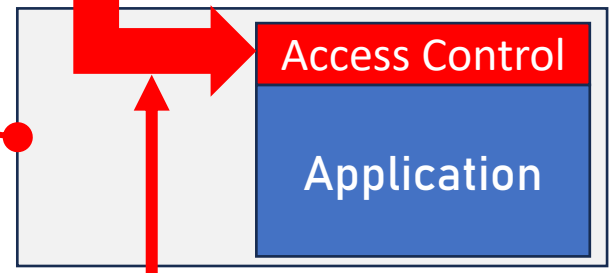
Example

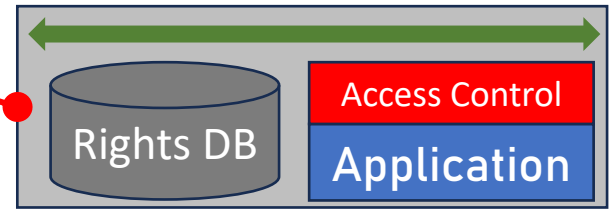
Horizontal Layers: **Functionality**

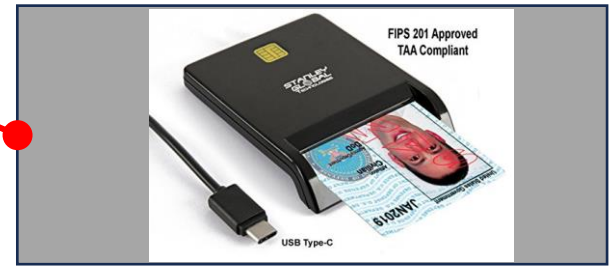
- Business Architecture
- Application Architecture
- Information Architecture
- Integration Architecture
- Technical Architecture

Security

Vertical Layers: **Quality Properties**







Framework for:

- ✓ Separation of Concerns
- ✓ Conceptual Integrity

Security: Authentication & Authorization

Hidden Vulnerabilities

Residual Probability of Hidden Vulnerabilities in the deployed System

(1) It is **impossible** to find all ***design faults*** in a large hardware/software system that may contain millions of lines of software code.

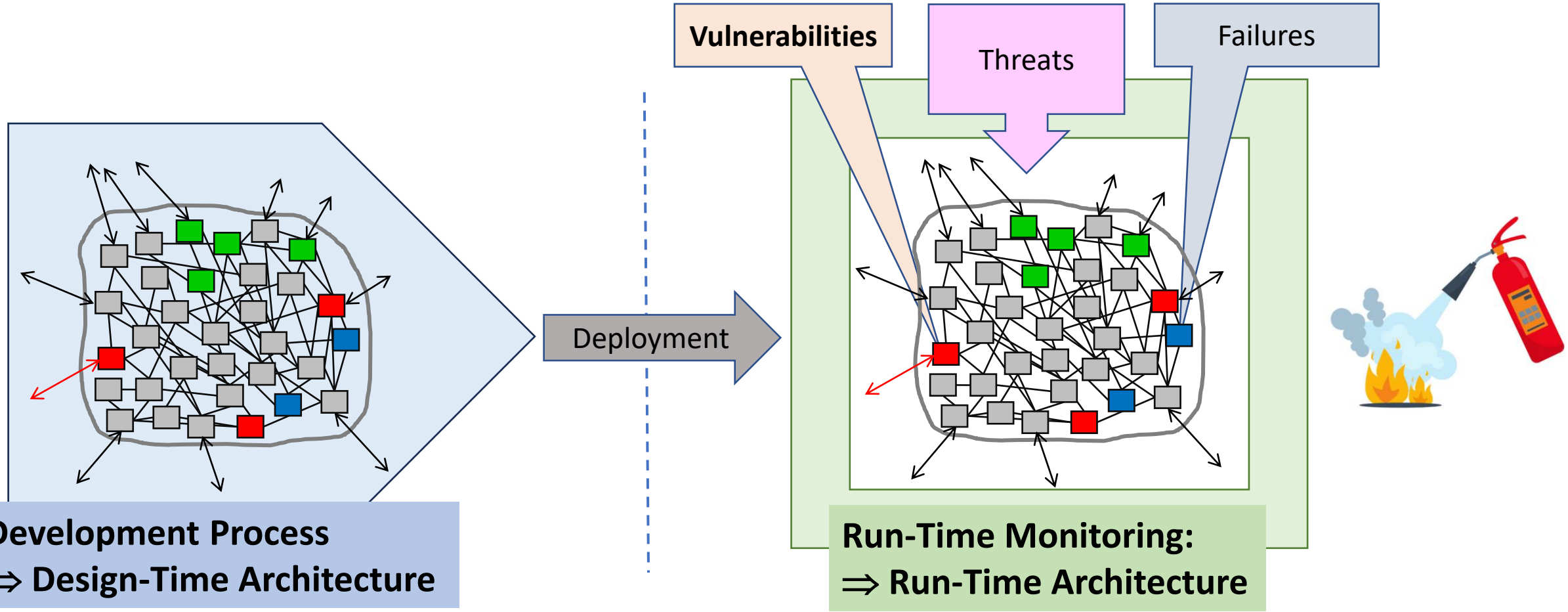
(2) It is **impossible** to find all ***implementation faults*** in a large hardware/software system that may contain millions of lines of software code.

(3) It is **impossible** to foresee all ***operational conditions*** of the system in its real environment (especially off-nominal conditions).

(4) It is **impossible** to mitigate all consequences of ***execution platform failures*** (Hardware failures, network outages, support systems non-availability [such as GPS or car-to-car comm, 5&6G]).

(5) It is **impossible** to fully identify emergent properties or ***emergent behaviour*** while assembling the system (especially negative emergence in large systems-of-systems)

Hermann Kopetz, 2023



Design-Time Architecture:

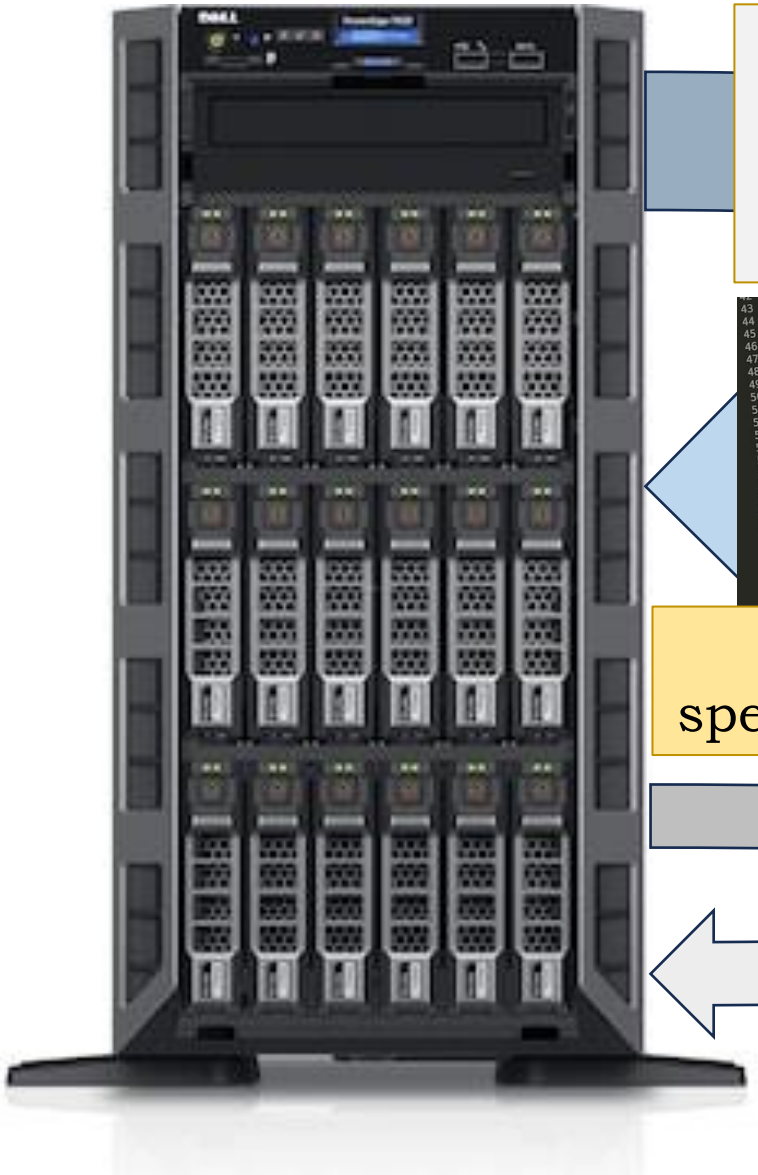
Detect, identify, analyze, and mitigate as many vulnerabilities as possible

Run-Time Architecture:

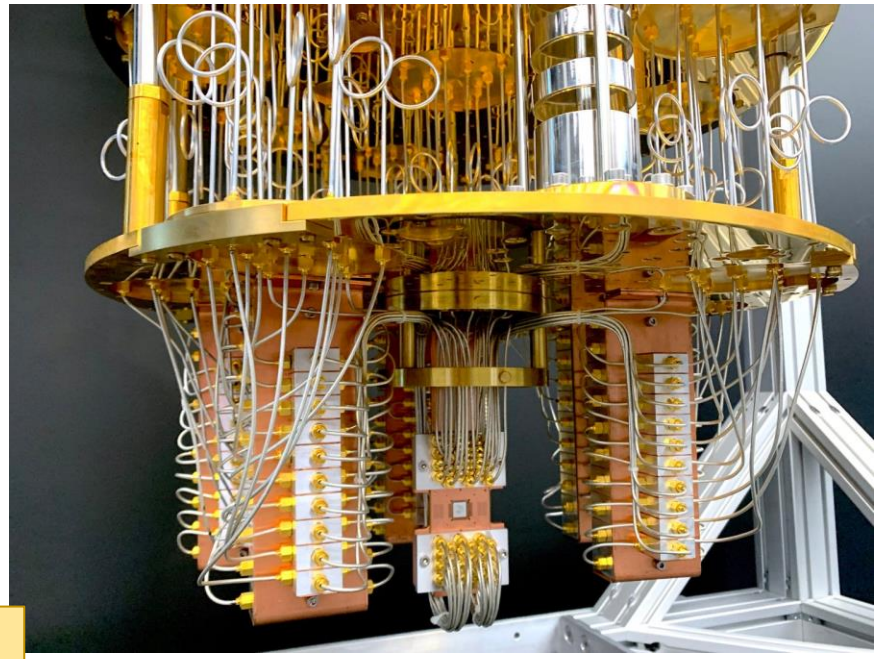
Prevent the consequences of undetected, unexpected or emerging vulnerabilities

https://www.istockphoto.com

Does **Quantum Computing** change the Architecture Principles?



NO
The Quantum Computer becomes a **Coprocessor** to a conventional Computer



However, using a specialized Programming Language

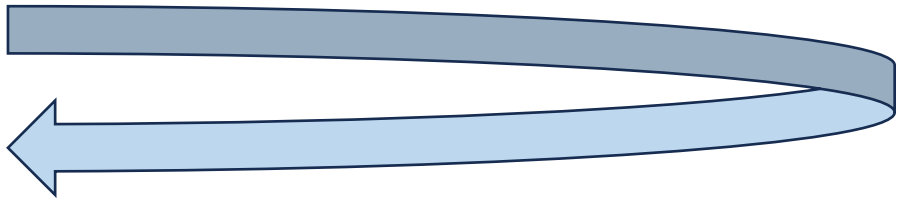
… just like a **graphic processor** chip

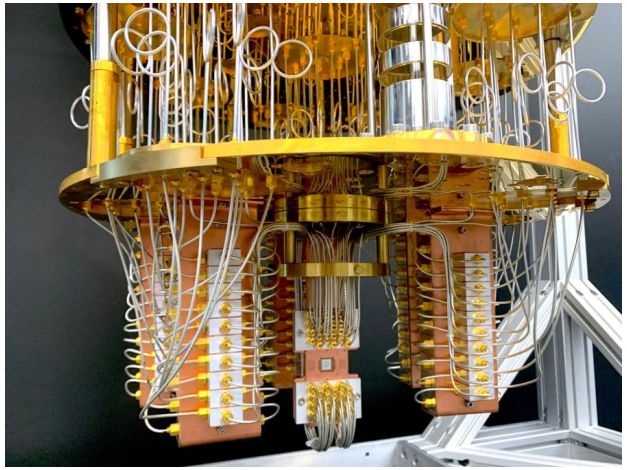
https://www.popsci.com/technology/in-photos-journey-to-the-center-of-a-quantum-computer/

https://www.notebookcheck.net/Microsoft-creates-a-new-coding-language-for-quantum-computers.251734.0.html

https://www.computer.org/publications/tech-news/chasing-pixels/intels-gpu-history

The Quantum Computer becomes a **Coprocessor** to a conventional Computer





Algorithms

- Control Flow
- Data Management
- User Interface
- Exception Handling

However:
Programming the **Quantum Computer** needs new Paradigms and Methods

https://www.freeimages.com

A.1.1 QO for the Deutsch Algorithm

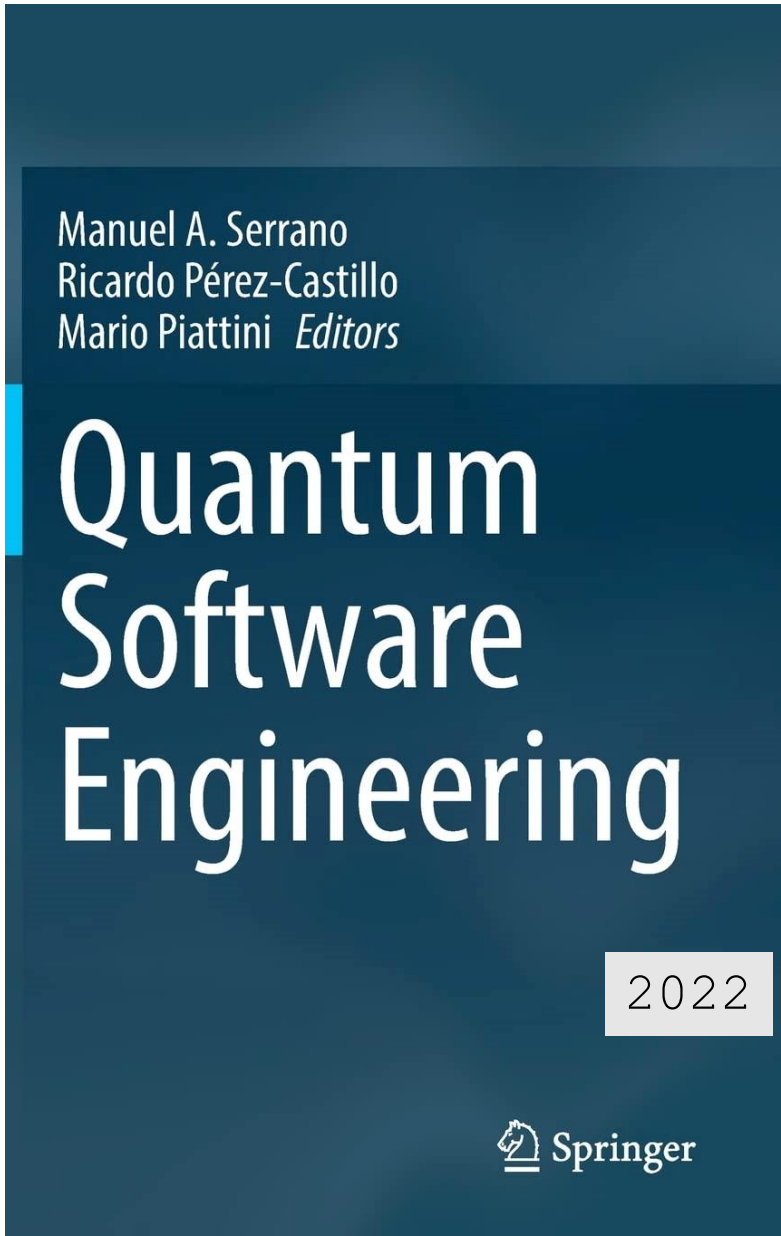
Example

```
— return a qubit in a specified state
qb :: [Char] -> QIO ( Qbit )
qb qstate
  | qstate = = "|0>" = mkQ( False )
  | qstate = = "|1>" = mkQ( True )
  | qstate = = "|+>" || qstate = = "H|0>" = do
            qBit <- qb( "|0>" )
            applyU( uhad( qBit ) )
            return( qBit )
  | qstate = = "|->" || qstate = = "H|1>" = do
            qBit <- qb( "|1>" )
            applyU( uhad( qBit ) )
            return( qBit )
  | otherwise = error "qb: wrong argument"

— apply the C-Not (N) gate to a qubit
qN :: (Bool -> Bool) -> Qbit -> Qbit -> QIO ()
qN f qx qy = applyU( cond (qx) (\ a → if f(a) then unot(qy) else mempty ))

— apply the Hadamard (H) gate to a qubit
qH :: Qbit -> QIO ()
qH qbit = applyU( uhad( qbit ) )

— measure a qubit
mq :: Qbit -> QIO ( Bool )
mq qbit = measQ( qbit )
```

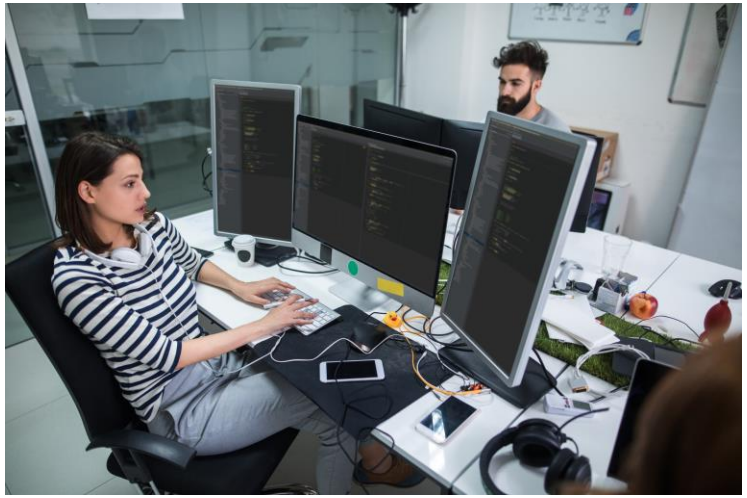


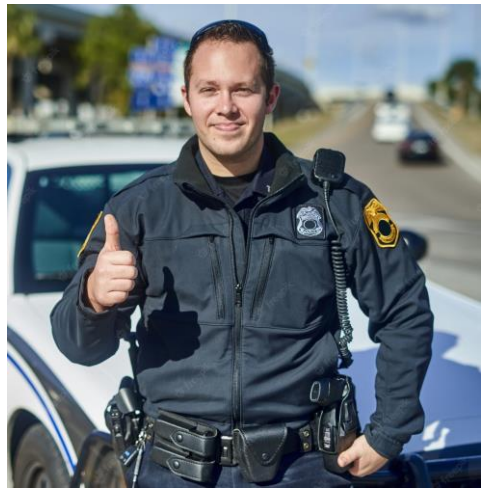
Architecture: Vision

https://www.istockphoto.com/de/vektor/

System-wide, strong, competent Architecture **Development** and **Maintenance**

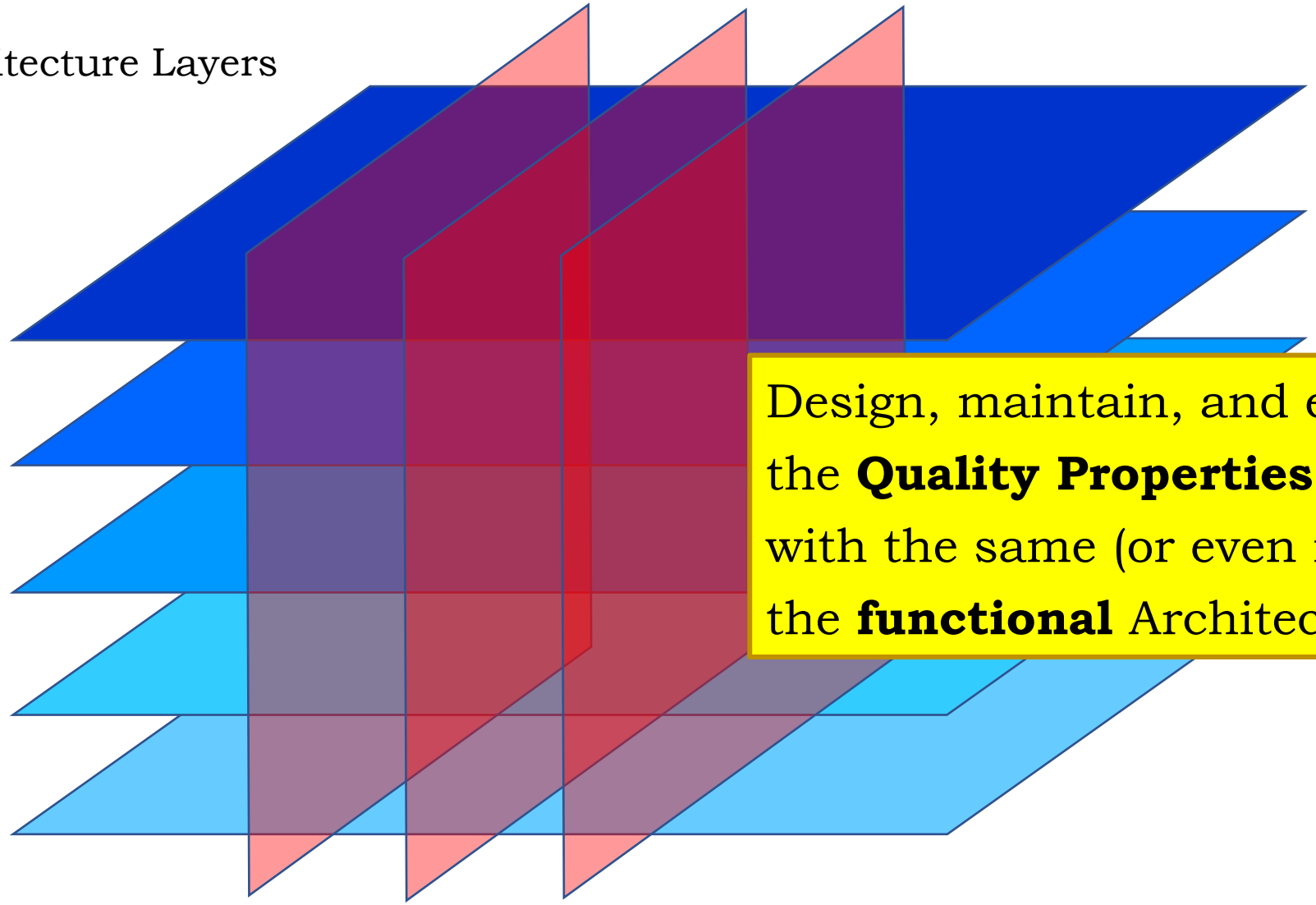
Keep the System Foundation intact at all times





System-wide, consistent, complete Architecture **Governance** & **Enforcement**

orizontal Architecture Layers



- Business Architecture
- Application Architecture
- Information Architecture
- Integration Architecture
- Technical Architecture

Design, maintain, and evolve the **Quality Properties Architectures** with the same (or even more) **care** as the **functional** Architecture

Safety

Security

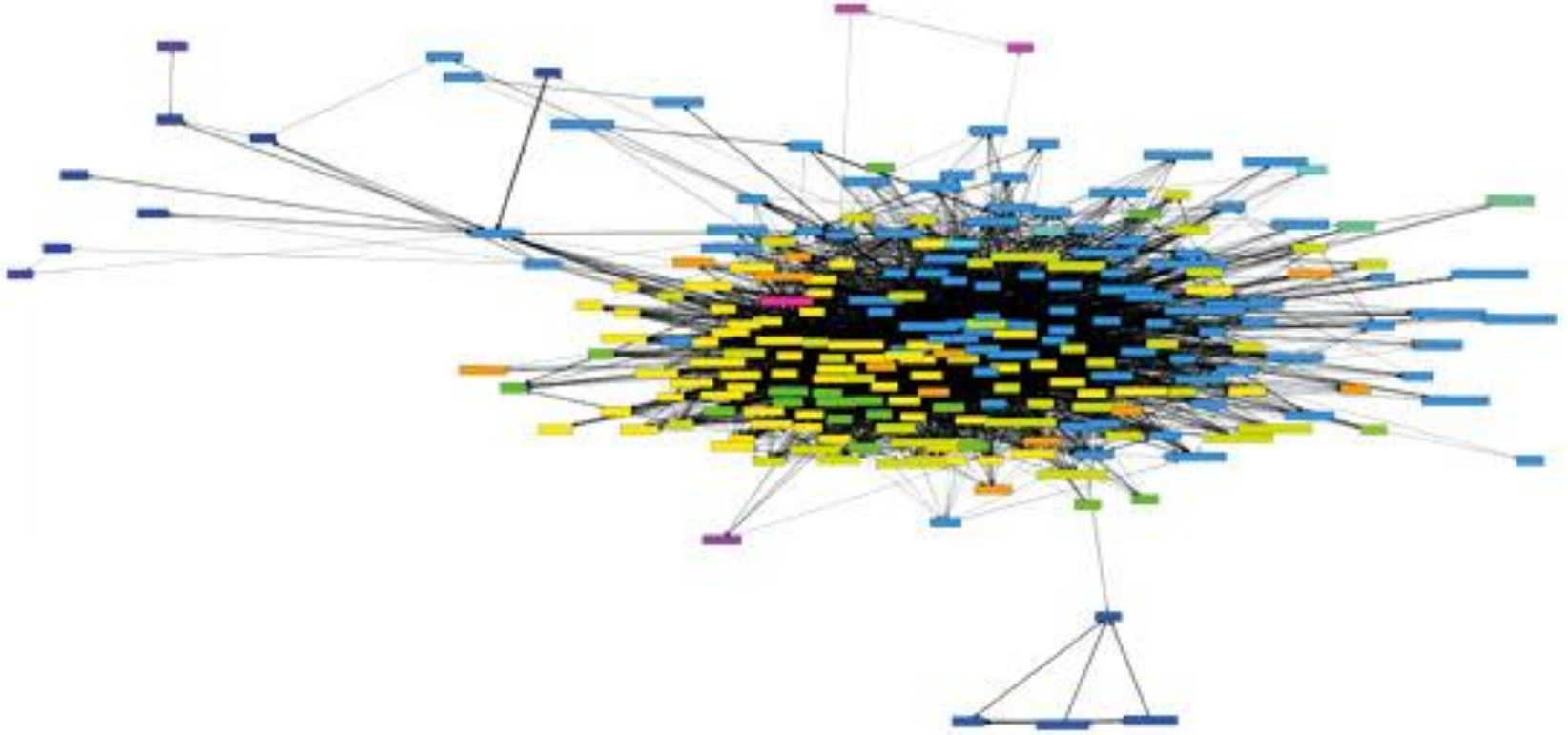
Real-Time

...

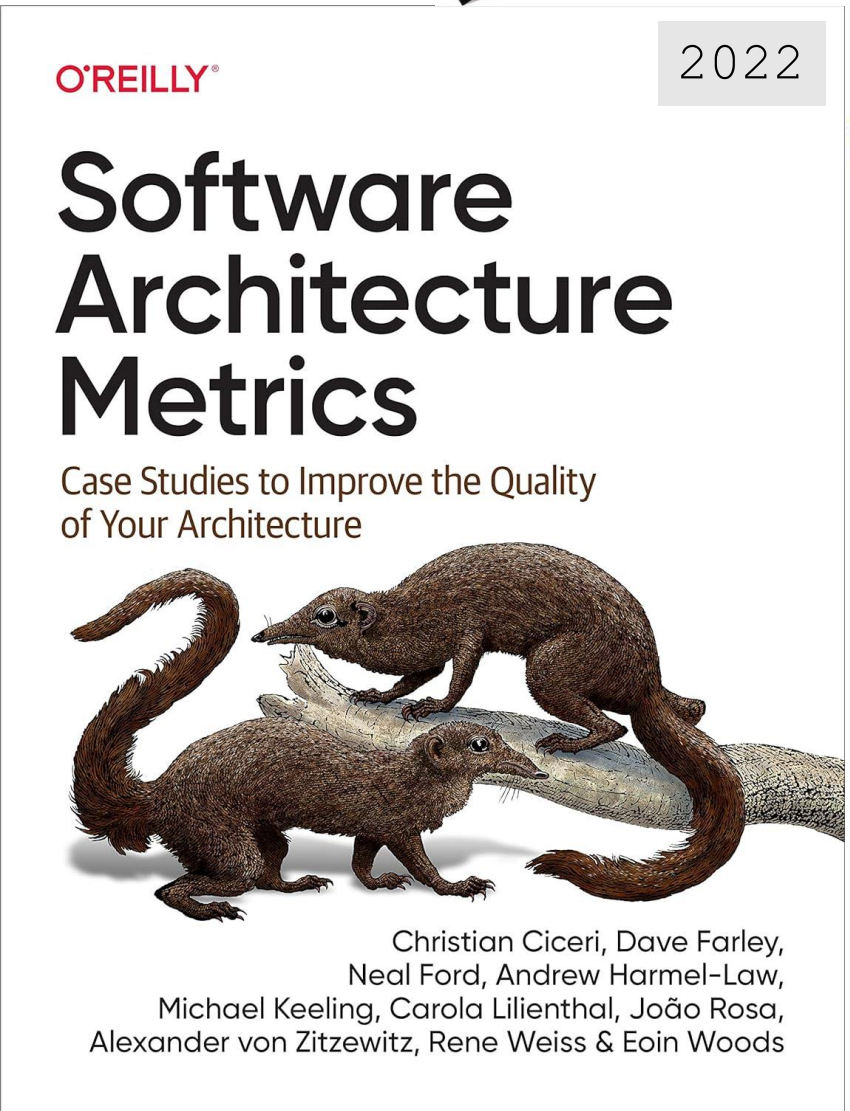
Vertical Architecture Layers

https://www.istockphoto.com

Use **Architecture Metrics** to:



https://jax.de/blog/modularisierung-und-kognitive-psychologie/



- Manage **Complexity** {Essential/Accidental Complexity}
- Control **Technical Debt**
- Prevent **Architecture Erosion**
- Provably ensure **Quality Properties**

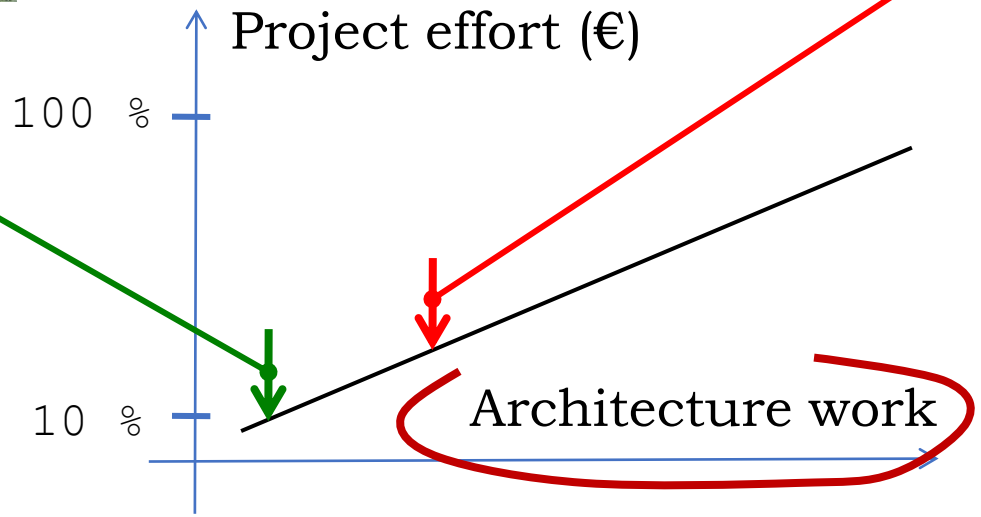
Risk-Based Architecture

Low Risk

http://www.dimensionsinfo.com/dimensions-of-a-dog-house

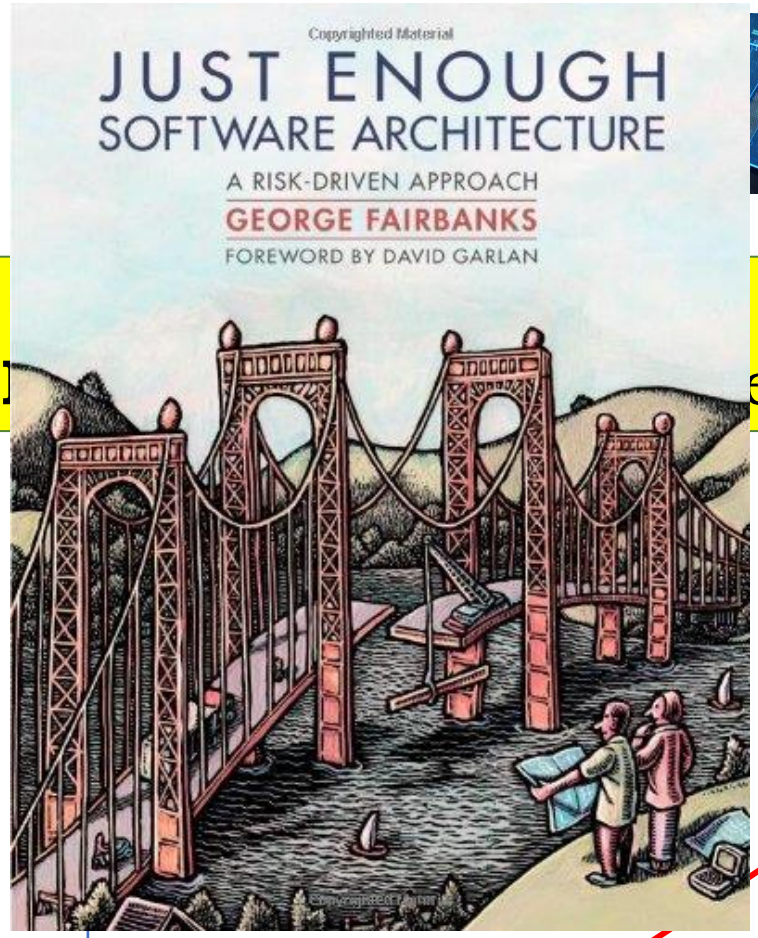
High Risk

http://www.skyscrapernews.org



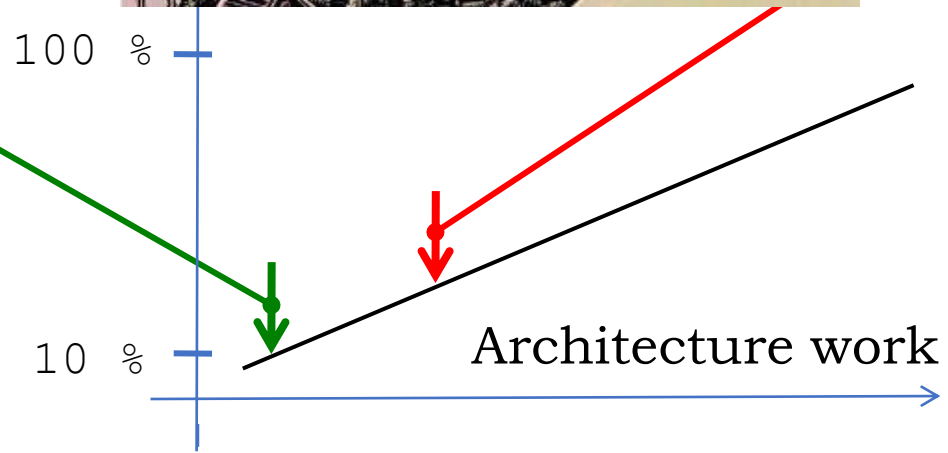
Low Risk

http://www.dimensionsinfo.com/dimensions-of-a-dog-house



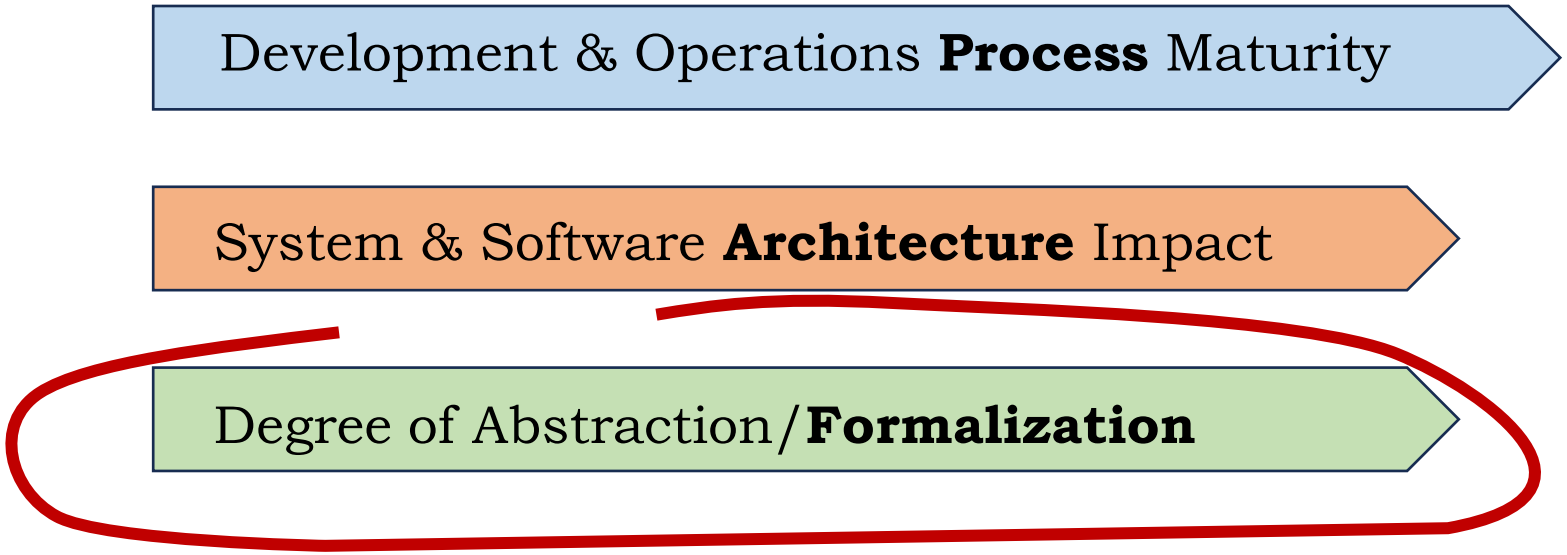
High Risk

http://www.skyscrapernews.org

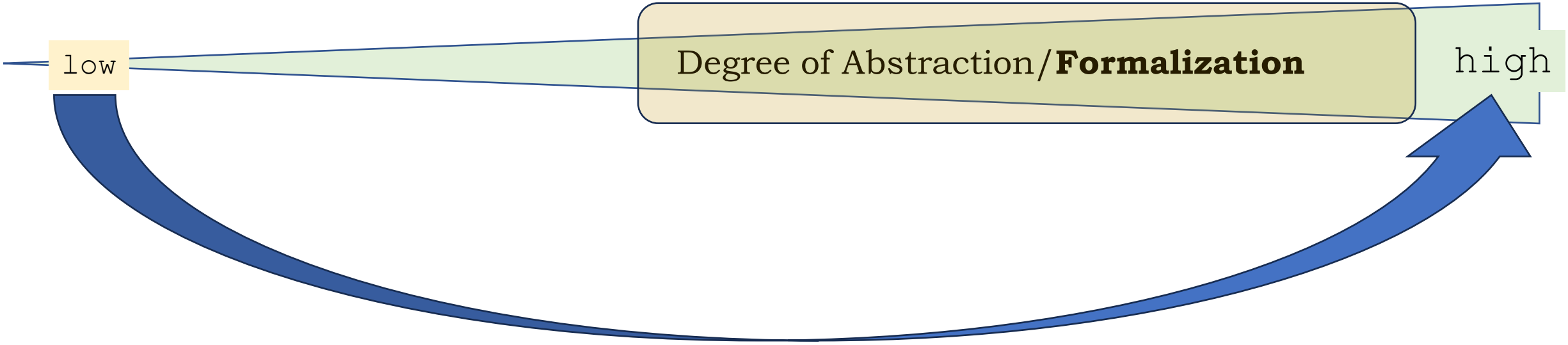


Prof. Dr. Frank J. Furrer

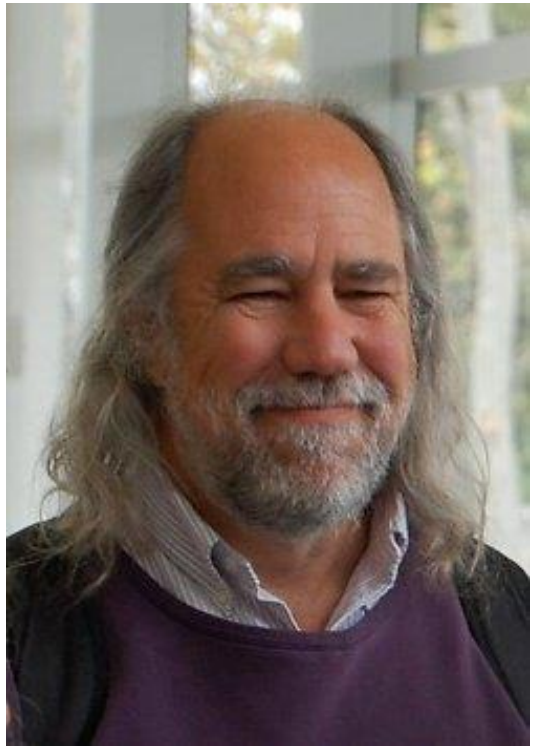
Software Development:
From Black Art to Engineering Science



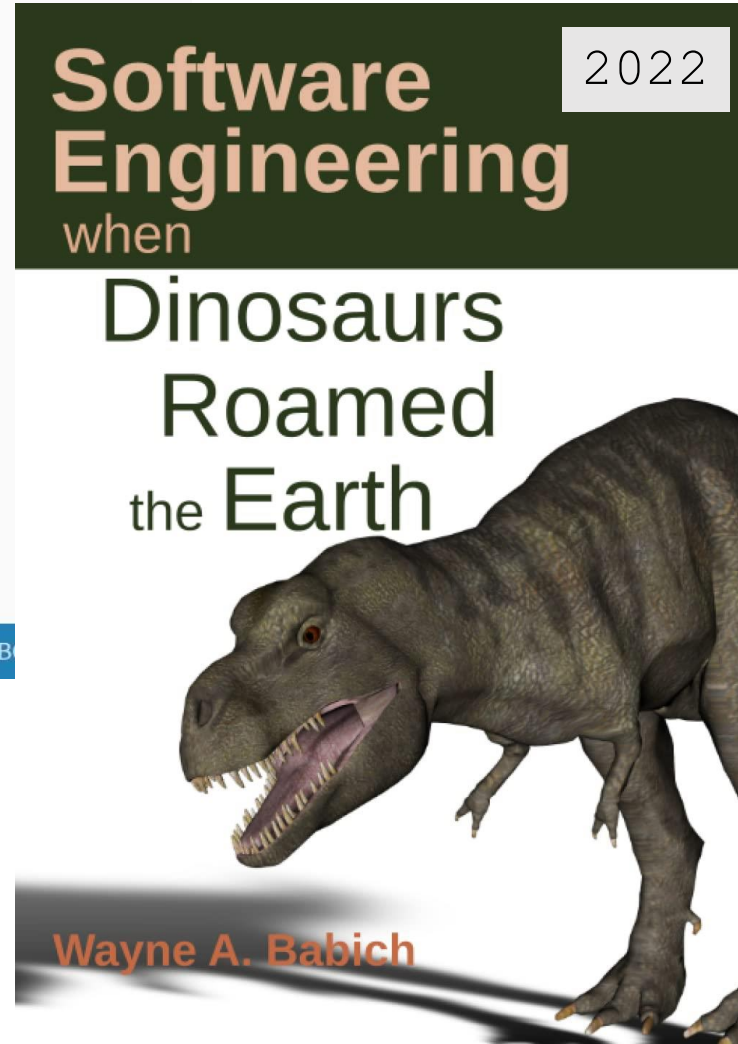
3



https://medium.com/swlh/programming-is-abstracting-part-1-7e7b2fb09854



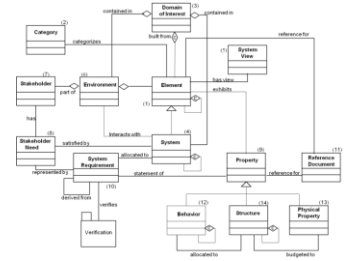
> THE ENTIRE HISTORY OF SOFTWARE ENGINEERING IS THAT OF THE RISE IN LEVELS OF ABSTRACTION.
>
> - GRADY BOOCH -



https://libquotes.com/grady-booch/quote/lbt7w1u

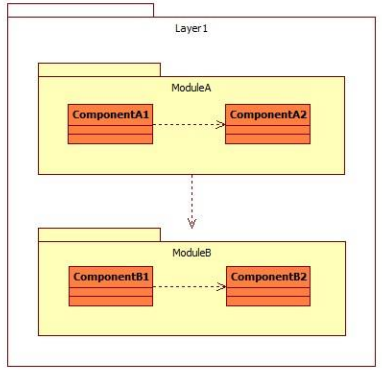
Level of Abstraction

high



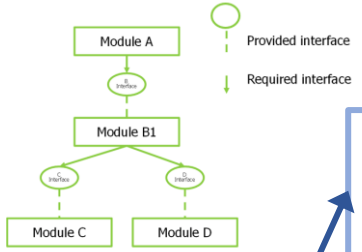
Model [MBSE]

Service



Component

Interface [Contract]



Module

```
def fahr_to_celsius(temp):
    return ((temp - 32) * (5/9))
```

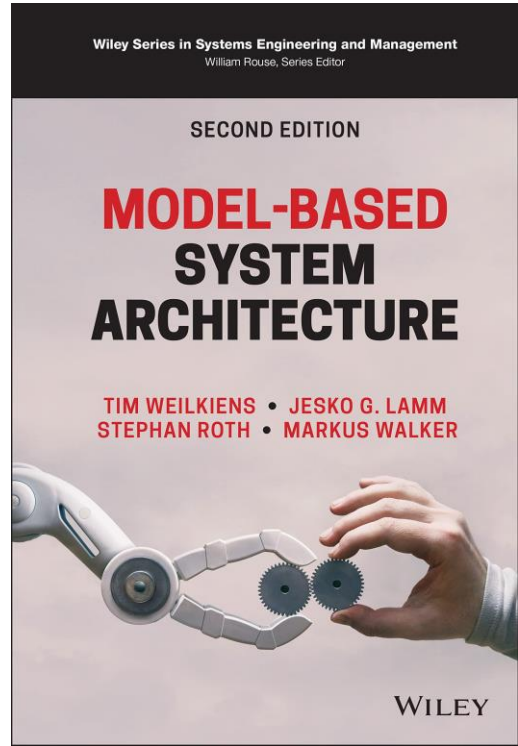
def keyword · name · parameter · return statement · return value

Subroutine

```
begin                0000
PAUSE                F000
MOV Rsw Rsrc1        0079
MOV Rsw R7seg        0078
PAUSE                F000
MOV Rsw Rsrc2        007A
MOV Rsw R7seg        0078
ADD                  1100
PAUSE                F000
MOV Rdest R7seg      00B8
end
```

Program

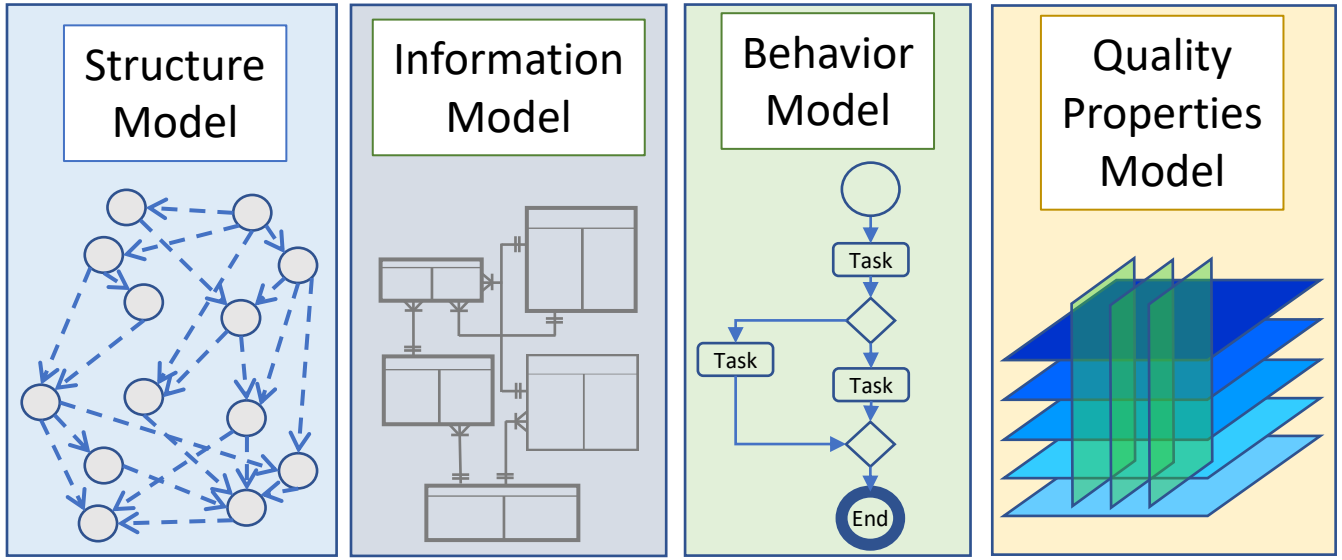
low

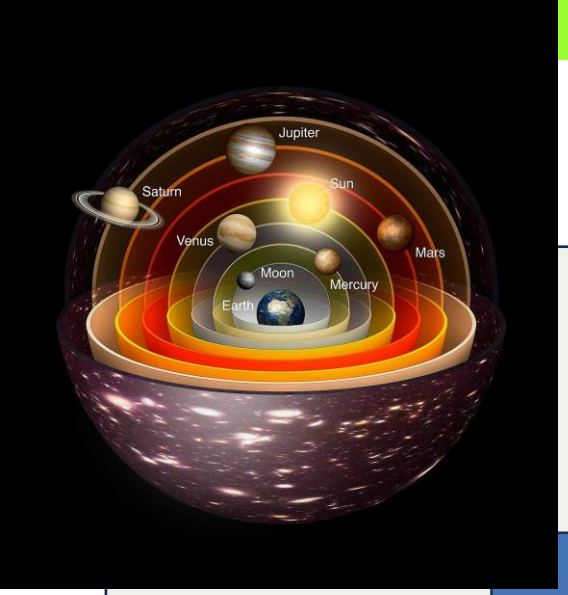


Model-based systems engineering (MBSE) is a formalized **methodology** that is used to support the requirements, design, analysis, verification, and validation associated with the **development of complex systems**

https://insights.sei.cmu.edu/blog/introduction-model-based-systems-engineering-mbse/

Context Model («World Model»)



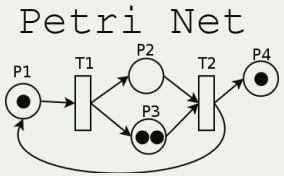


Overarching, formal, comprehensive Modeling Method

ARCHITECTURE DESCRIPTION LANGUAGE (ADL)

ERD

Petri Net



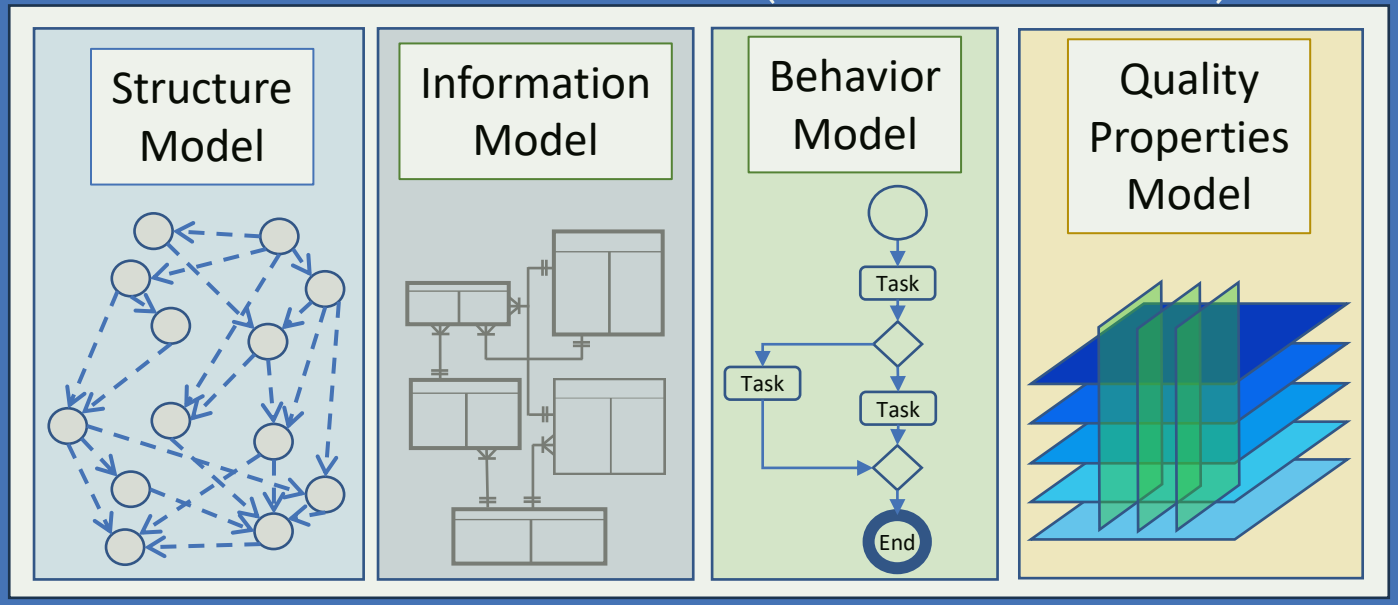
ONTOLOGY

BPMN™

Schemas

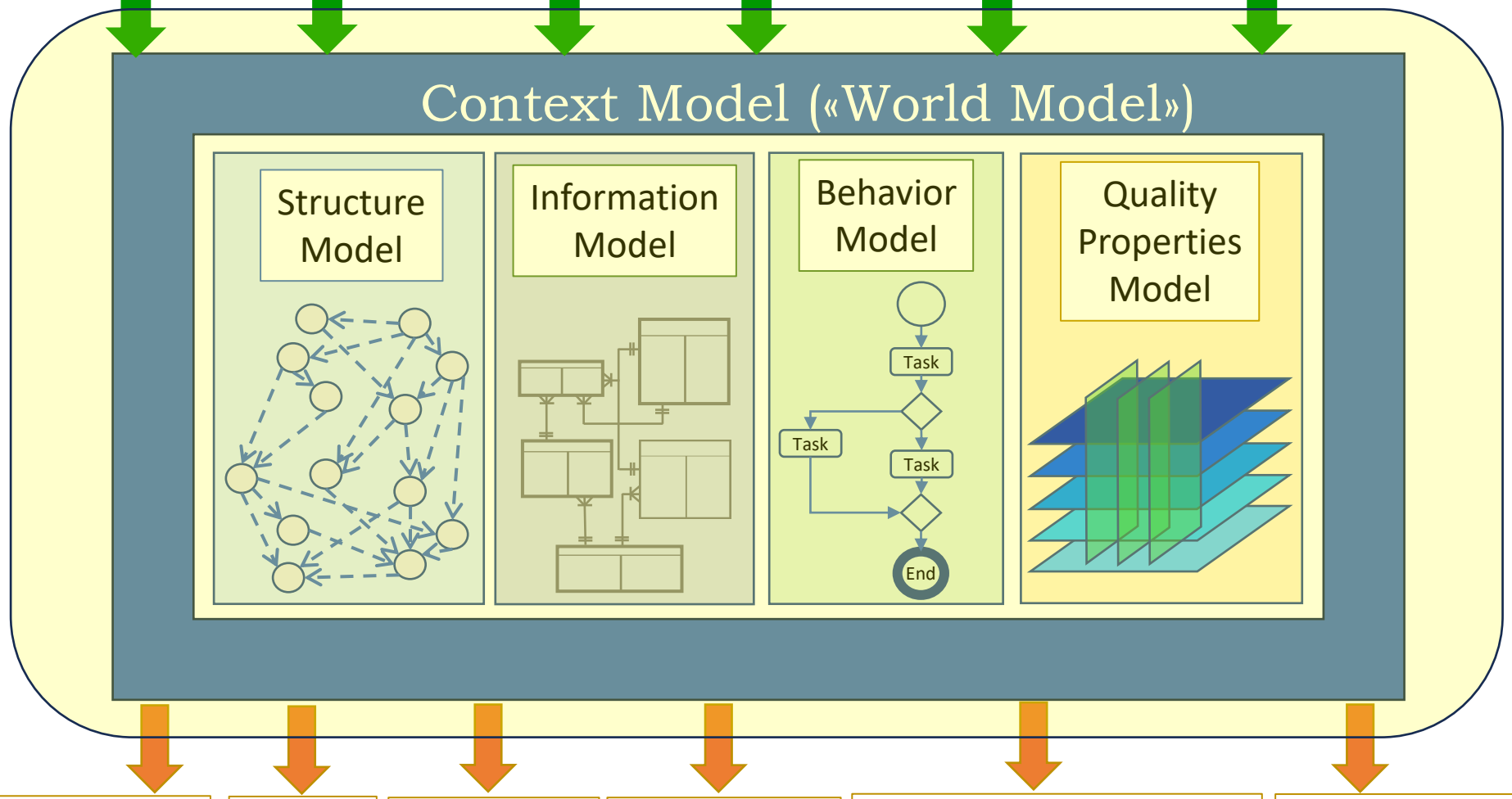
... etc.

Context Model («World Model»)



Model Boundary {UoD} | Model Purpose & Audience | Domain Model | Domain Ontology | Modeling Philosophy | Formal Mathematical Foundation

Overarching, formal, comprehensive Modeling Method



Model Architecture | Meta-Model | Syntax & Notation | Semantics | Modeling Principles & Rules | Model Verification & Validation

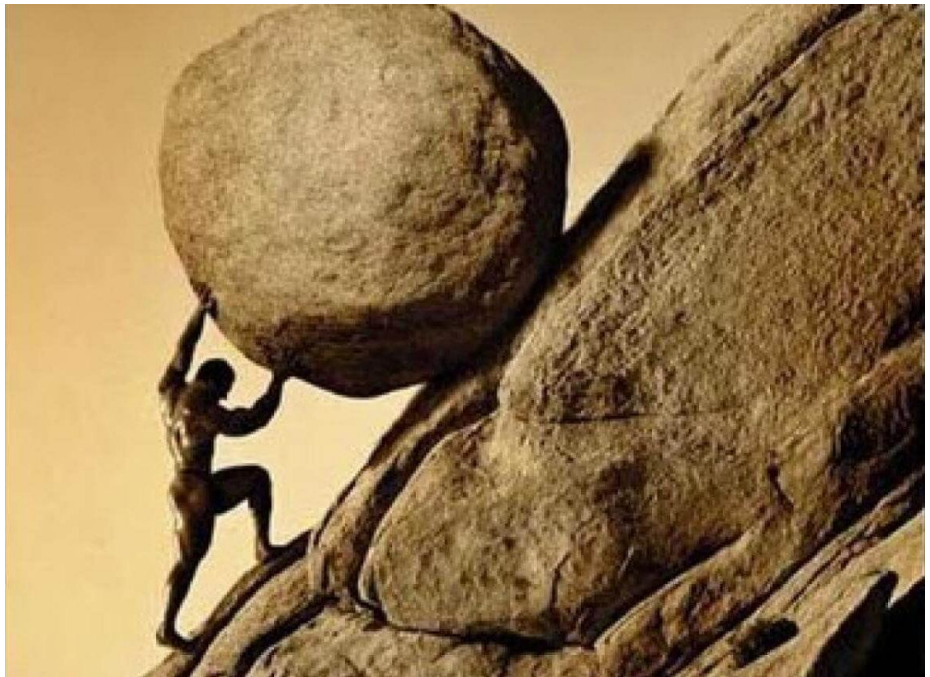
https://www.amazon.com/Flat-Earth-Dome-Display-Model/dp/B07RBKKFN7

Formal methods *make us confront the hard problems early.*

$\Rightarrow$ *the difficulties cannot be escaped, only deferred.*

Superficial methods put off the hard parts until coding and testing – but then they appear with a vengeance.

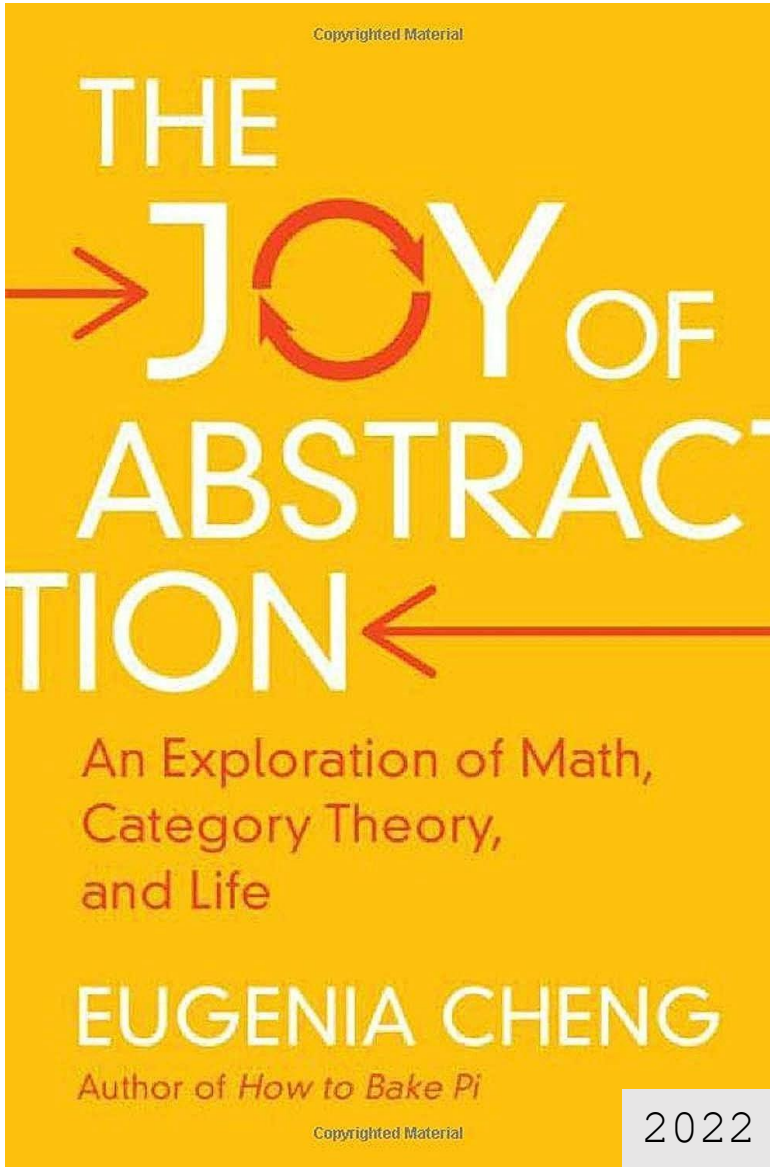
Jonathan Jacky, 1997 (ISBN 978-0-521-55976-8)



https://www.davelabowitz.com/blog/sisyphus-and-leadership/

Abstraction: Vision

https://www.istockphoto.com/de/vektor/



The three Devils of Systems Engineering

Change

Complexity

Uncertainty

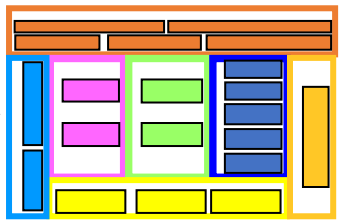
Real World

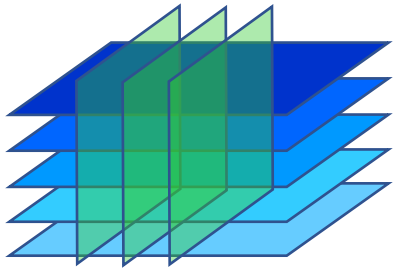
Domain Model {UoD}

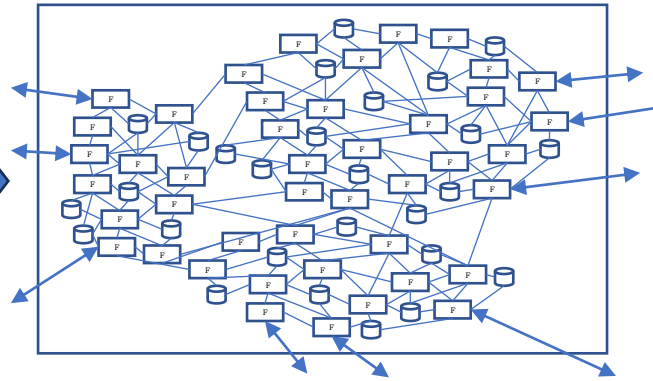
Architecture

Implementation [Runtime System]

System Evolution



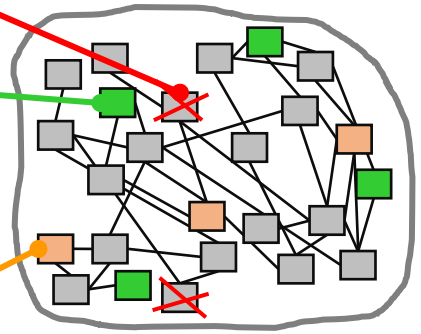




Delete

Create

Modify



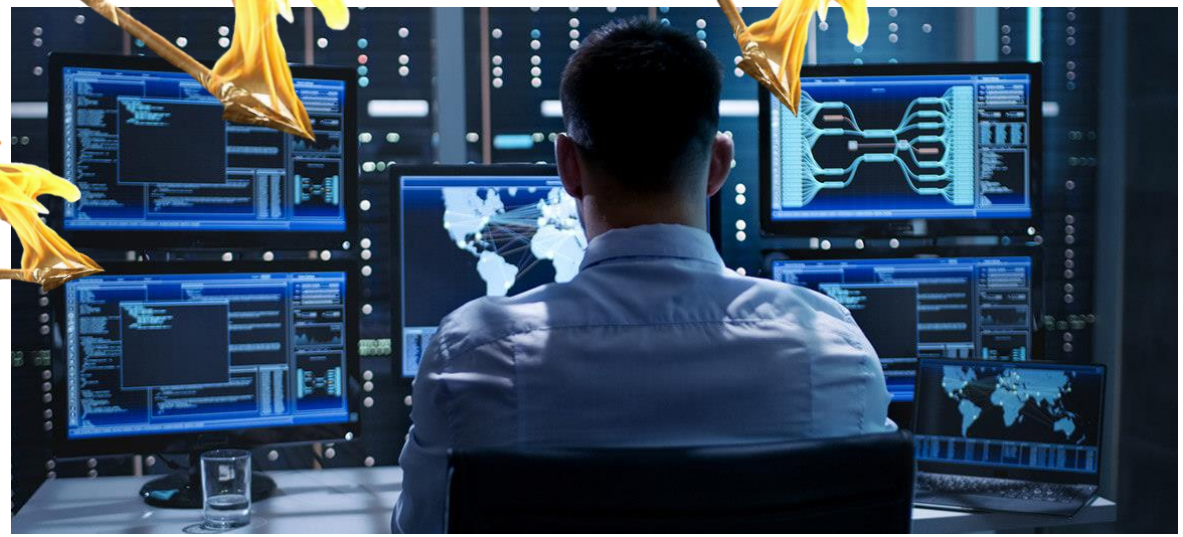
Vision: Full, life-time conceptual integrity of all concepts, terminology, and definitions

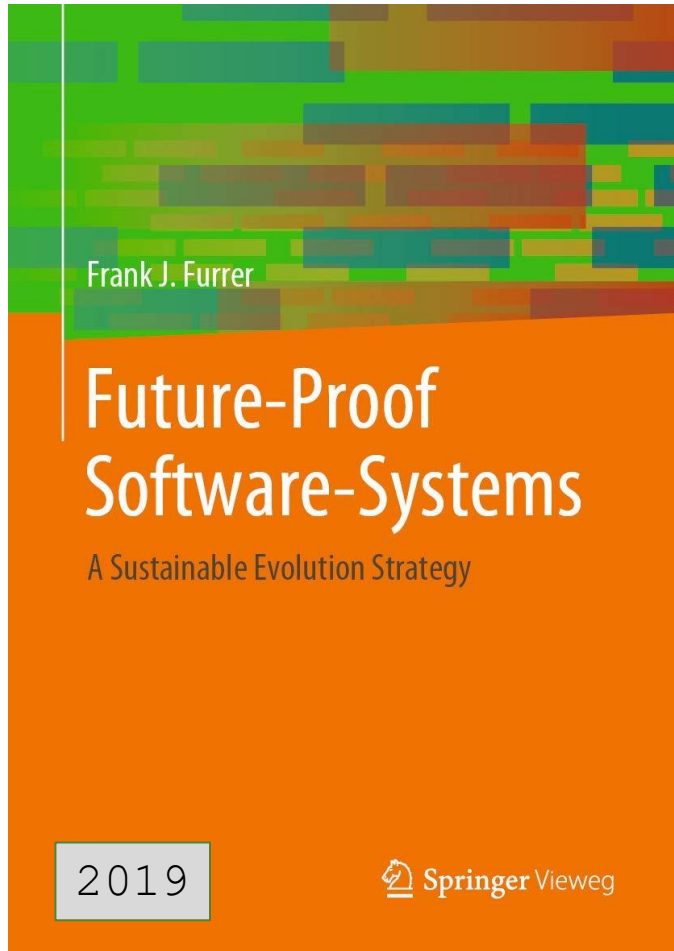
The three Devils of Systems Engineering

Relentless Change

Brutal Complexity

Frightening Uncertainty

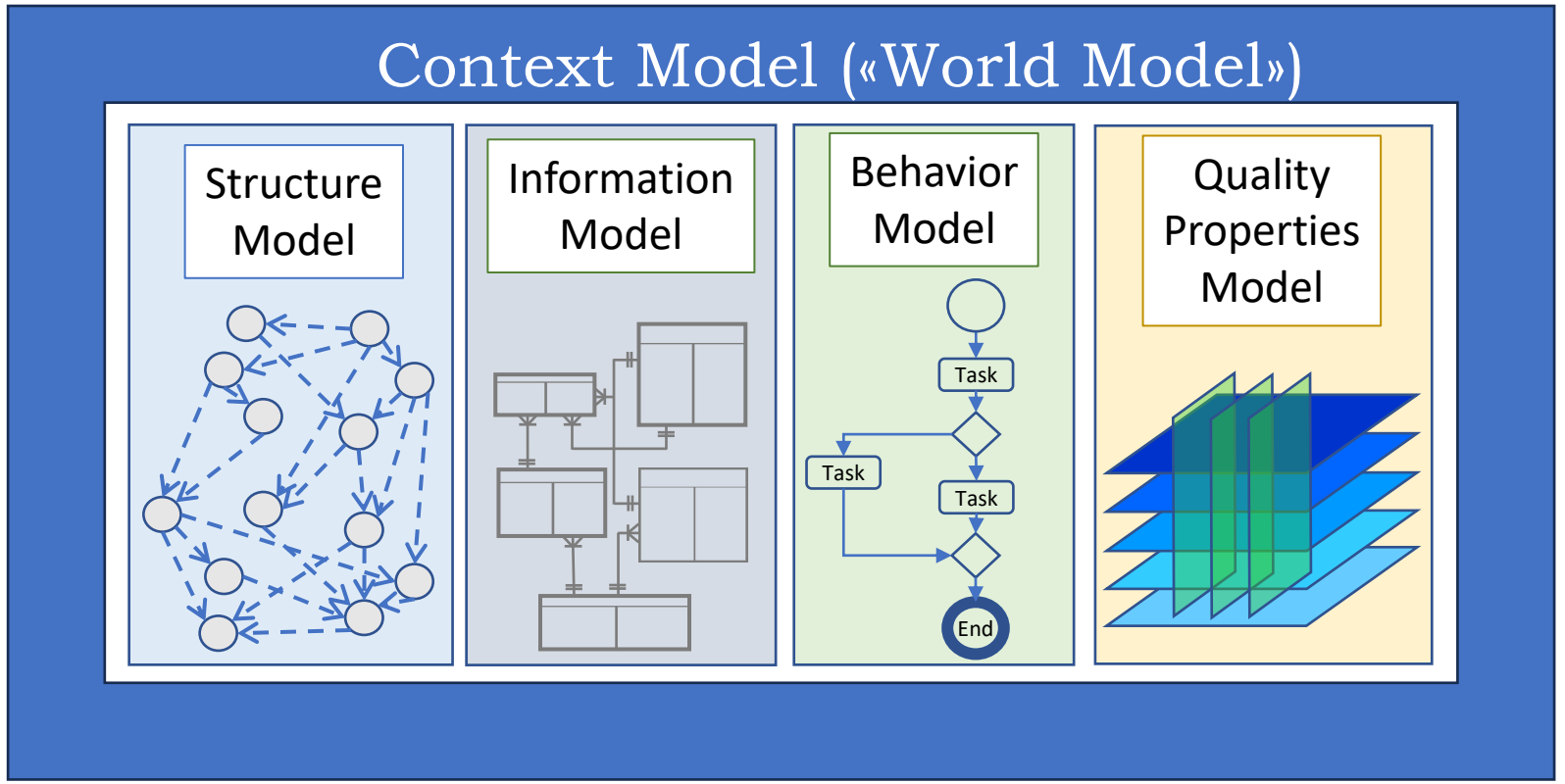




https://science.howstuffworks.com

Vision A:

Overarching, formal, comprehensive Modeling Method & Language



https://penntoday.upenn.edu

Vision B:

A small number of formal, consistent, complete, and comprehensive Modeling Methods & Languages

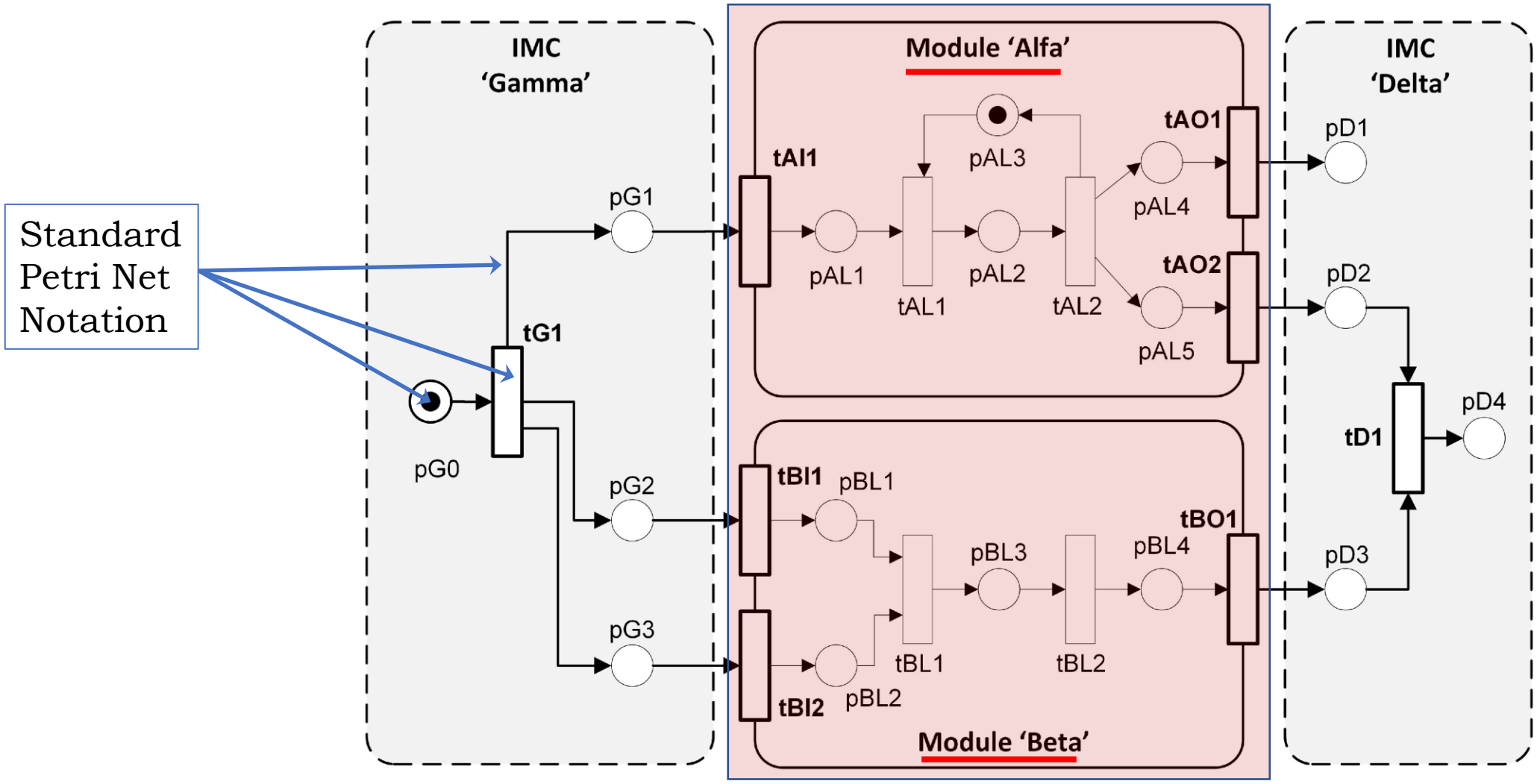
Prof. Dr. Frank J. Furrer

Software Development:
From Black Art to Engineering Science

Example: Formal System

Example: Modular Petri Nets

1 Petri Module



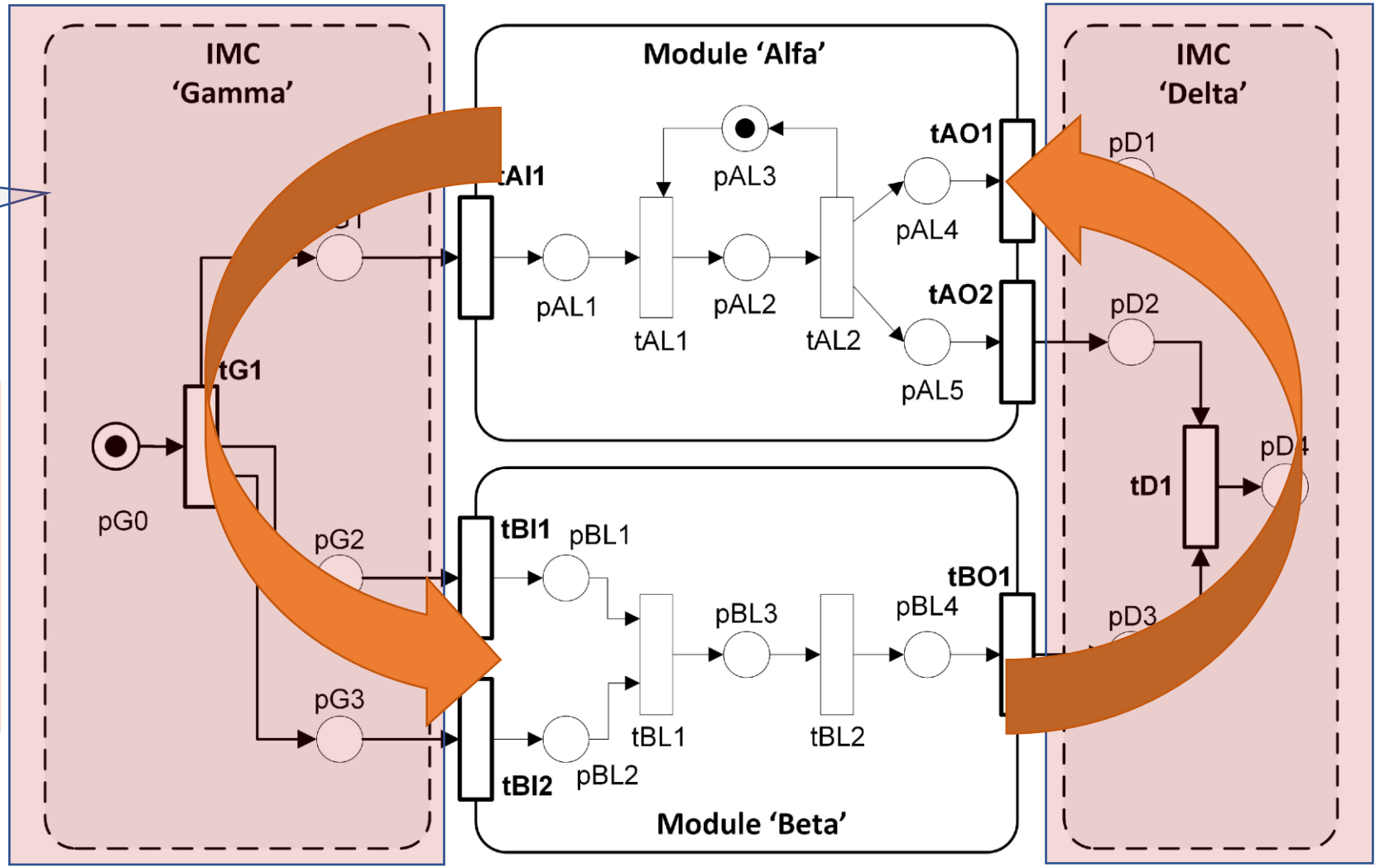
Example: Modular Petri Nets

2 Inter-Modular Connector [Composition]

Inter-Modular Connector [IMC]

Connecting/Composing Petri Modules:

- Functionality
- Control Flow
- Timing
- Mesh Structure
- Same Formalism



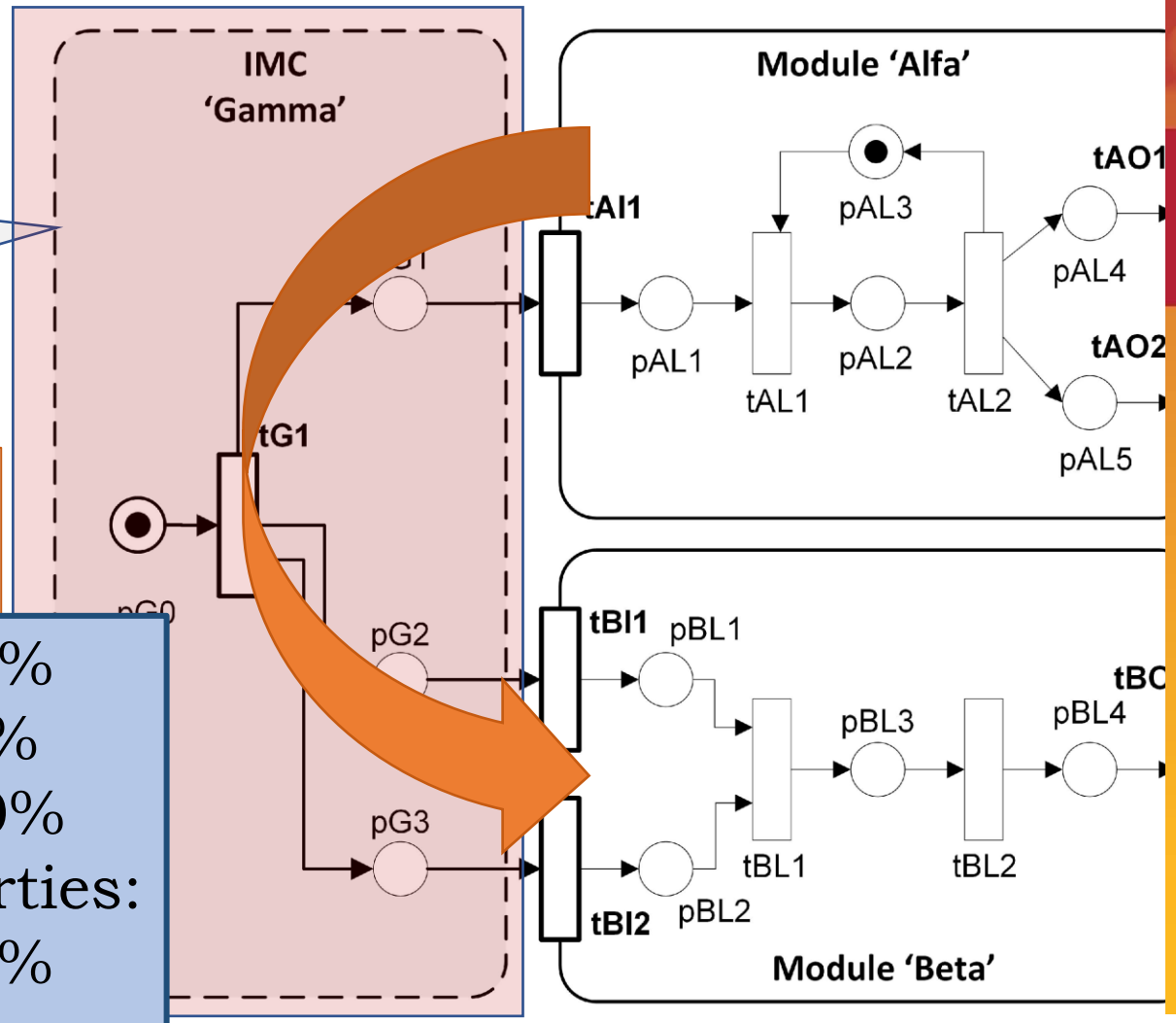
Example: Modular Petri Nets

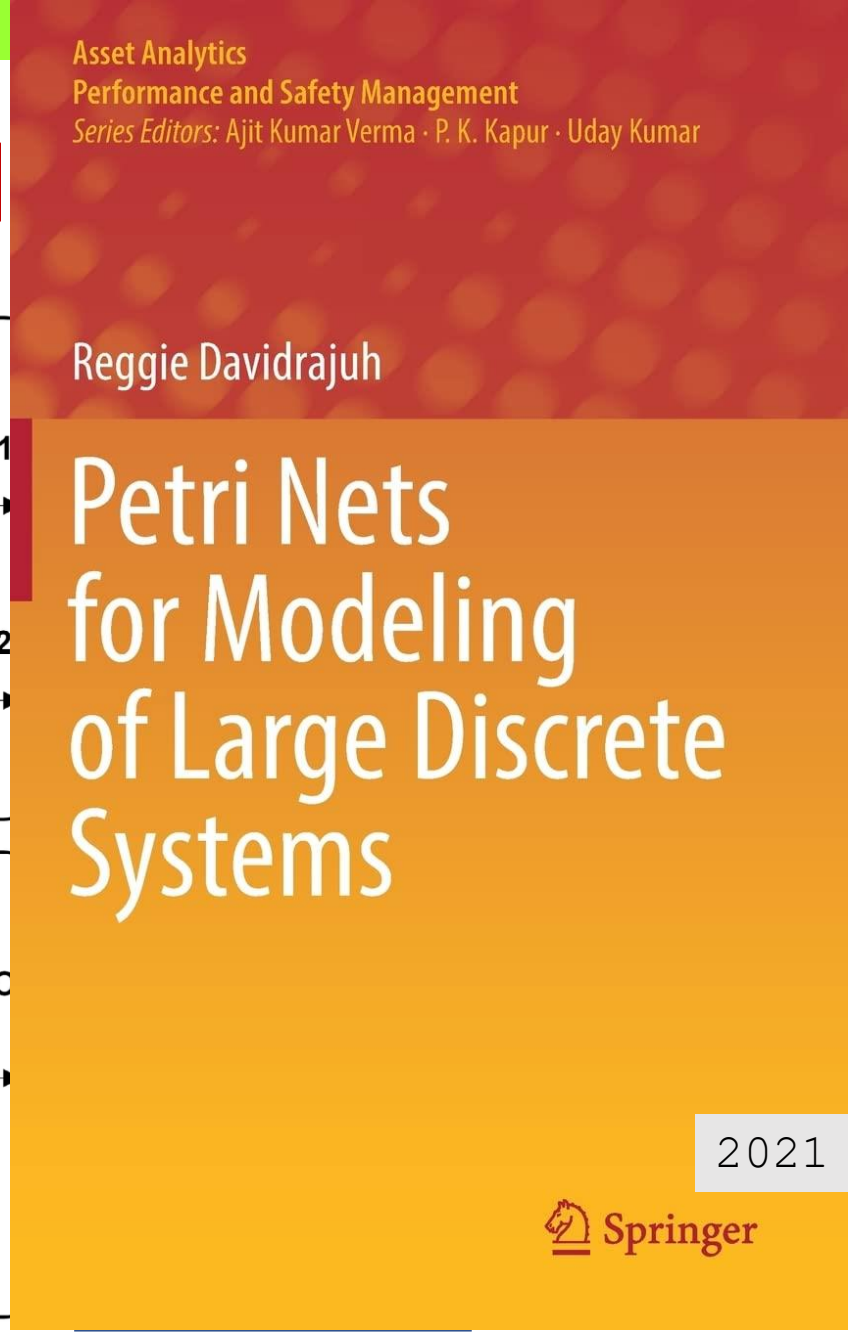
2 Inter-Modular Connector [Composition]

Inter-Modular Connector [IMC]

Connecting/Composing Petri Modules:

- Functionality
- Behavior: 100%
- Structure: 50%
- Information: 0%
- Quality Properties:
  - Timing: 100%
  - Rest: 0%





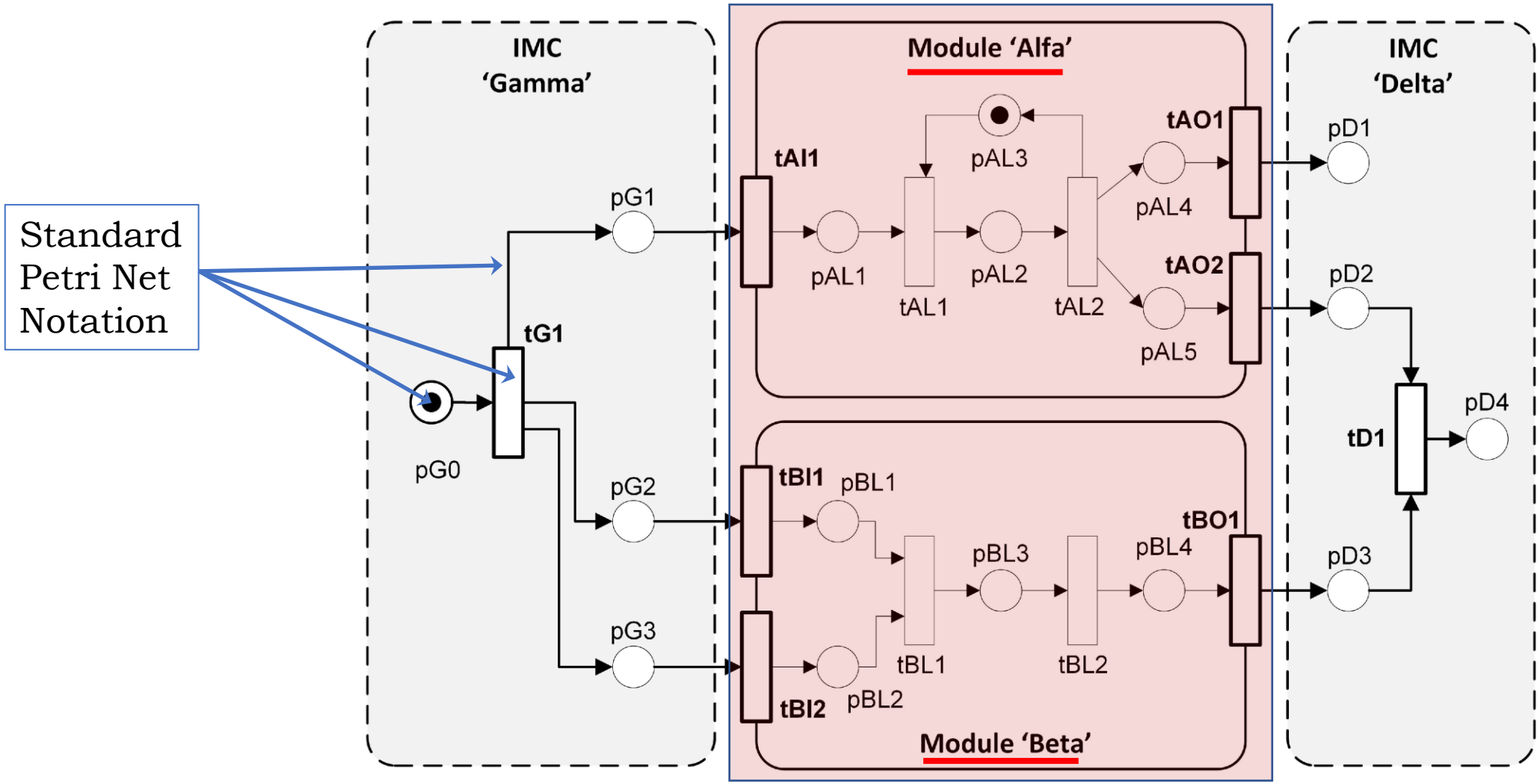
Prof. Dr. Frank J. Furrer

Software Development:
From Black Art to Engineering Science

Example: Formal System

Example: Modular Petri Nets

1 Petri Module



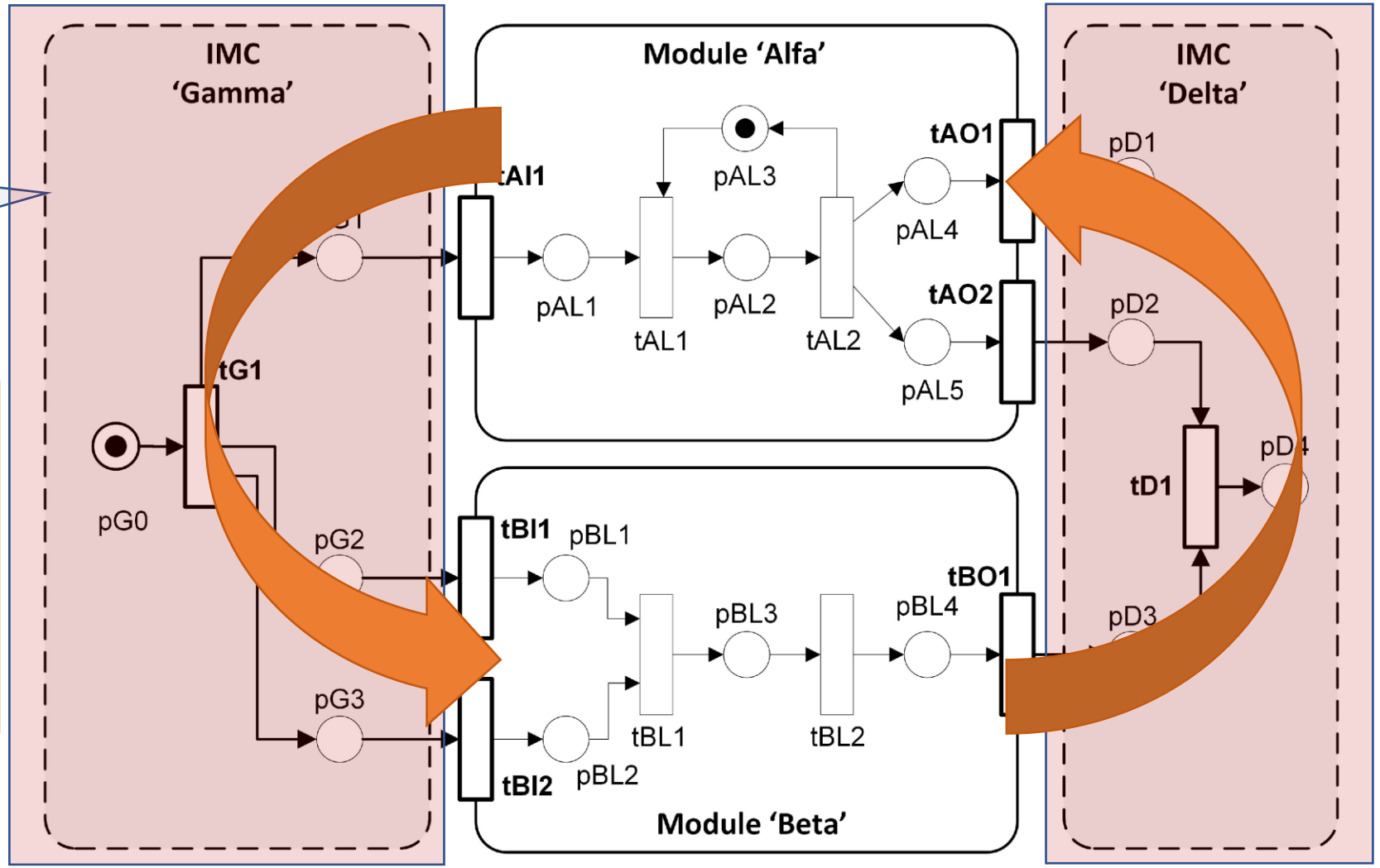
Example: Modular Petri Nets

2 Inter-Modular Connector [Composition]

Inter-Modular Connector [IMC]

Connecting/Composing Petri Modules:

- Functionality
- Control Flow
- Timing
- Mesh Structure
- Same Formalism



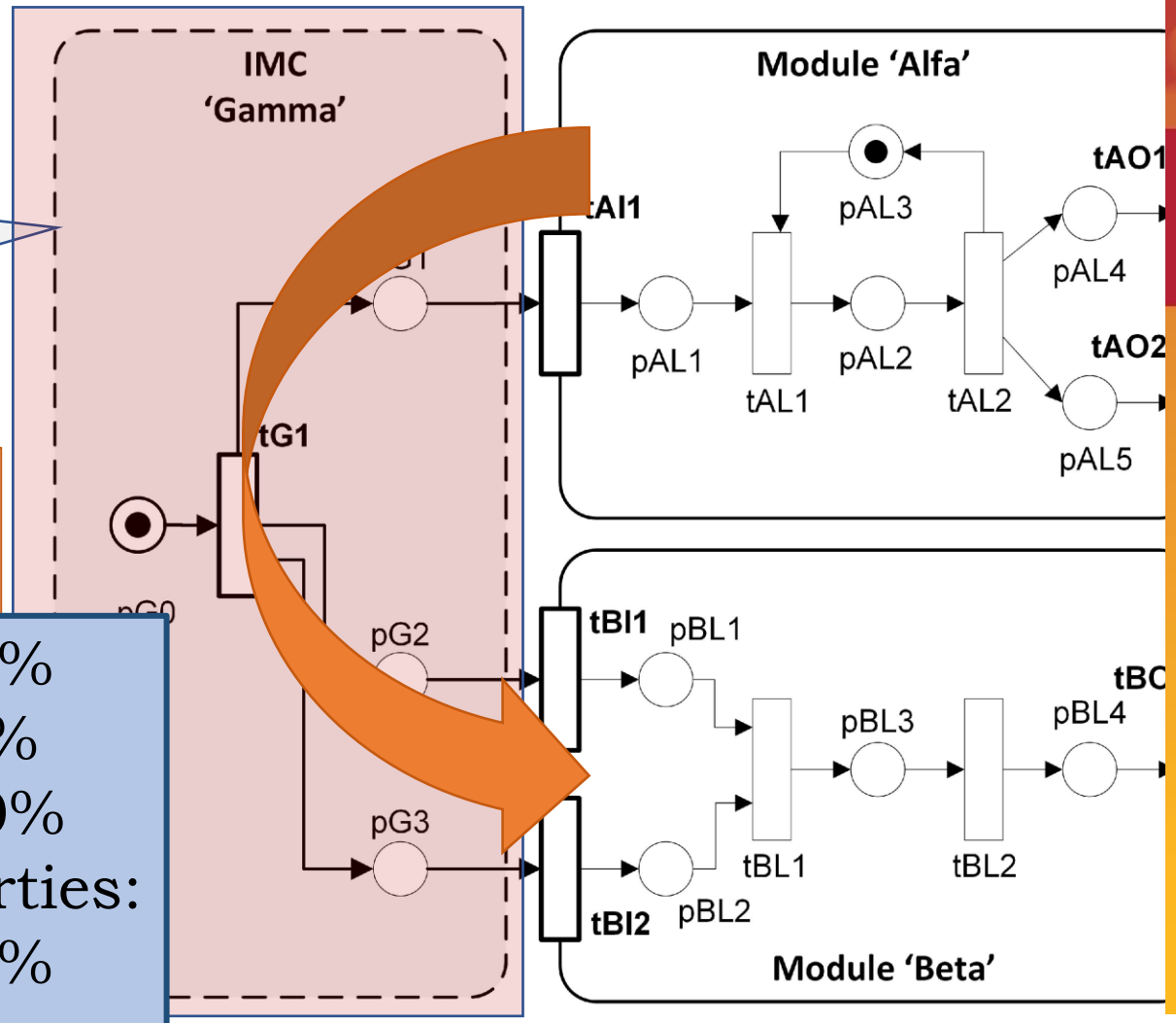
Example: Modular Petri Nets

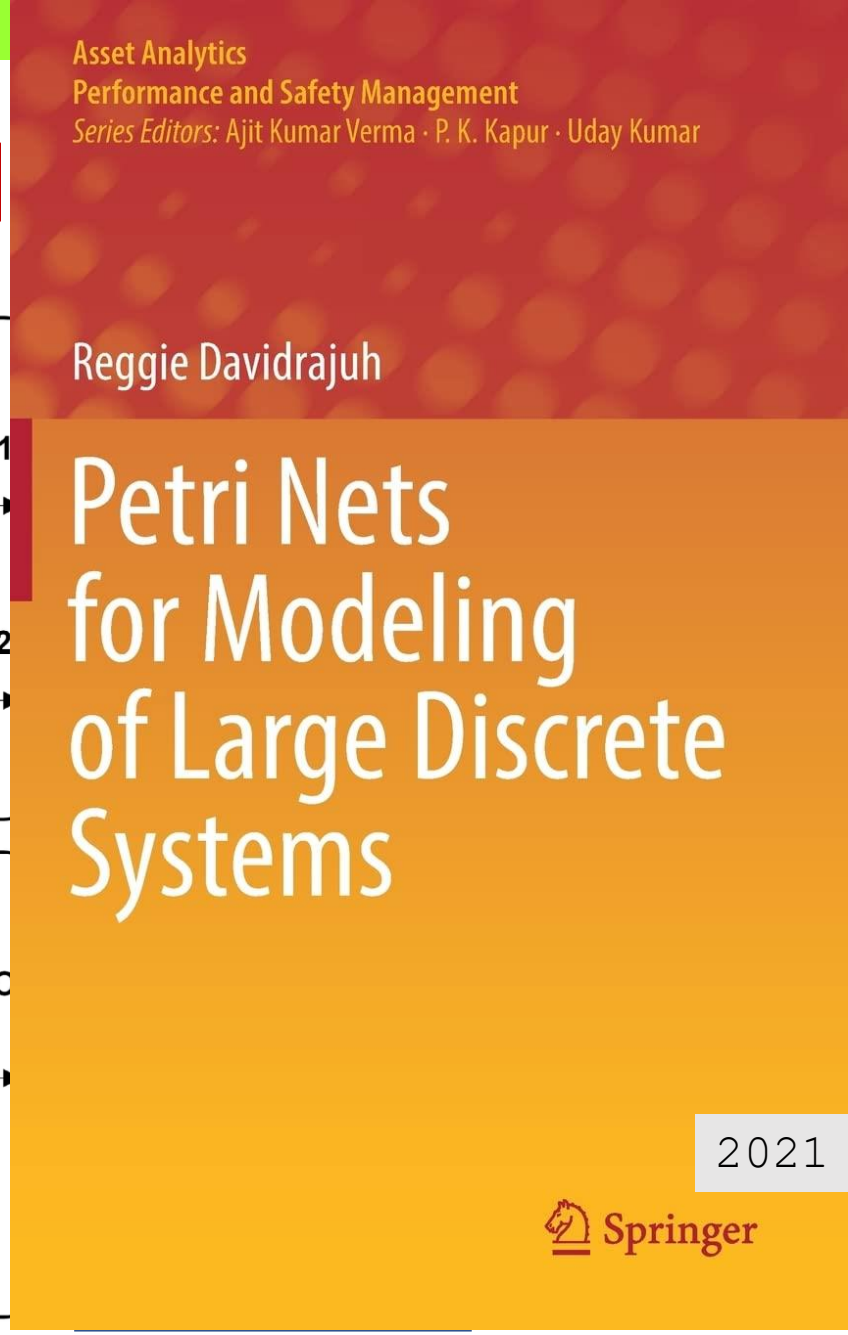
2 Inter-Modular Connector [Composition]

Inter-Modular Connector [IMC]

Connecting/Composing Petri Modules:

- Functionality
- Behavior: 100%
- Structure: 50%
- Information: 0%
- Quality Properties:
  - Timing: 100%
  - Rest: 0%

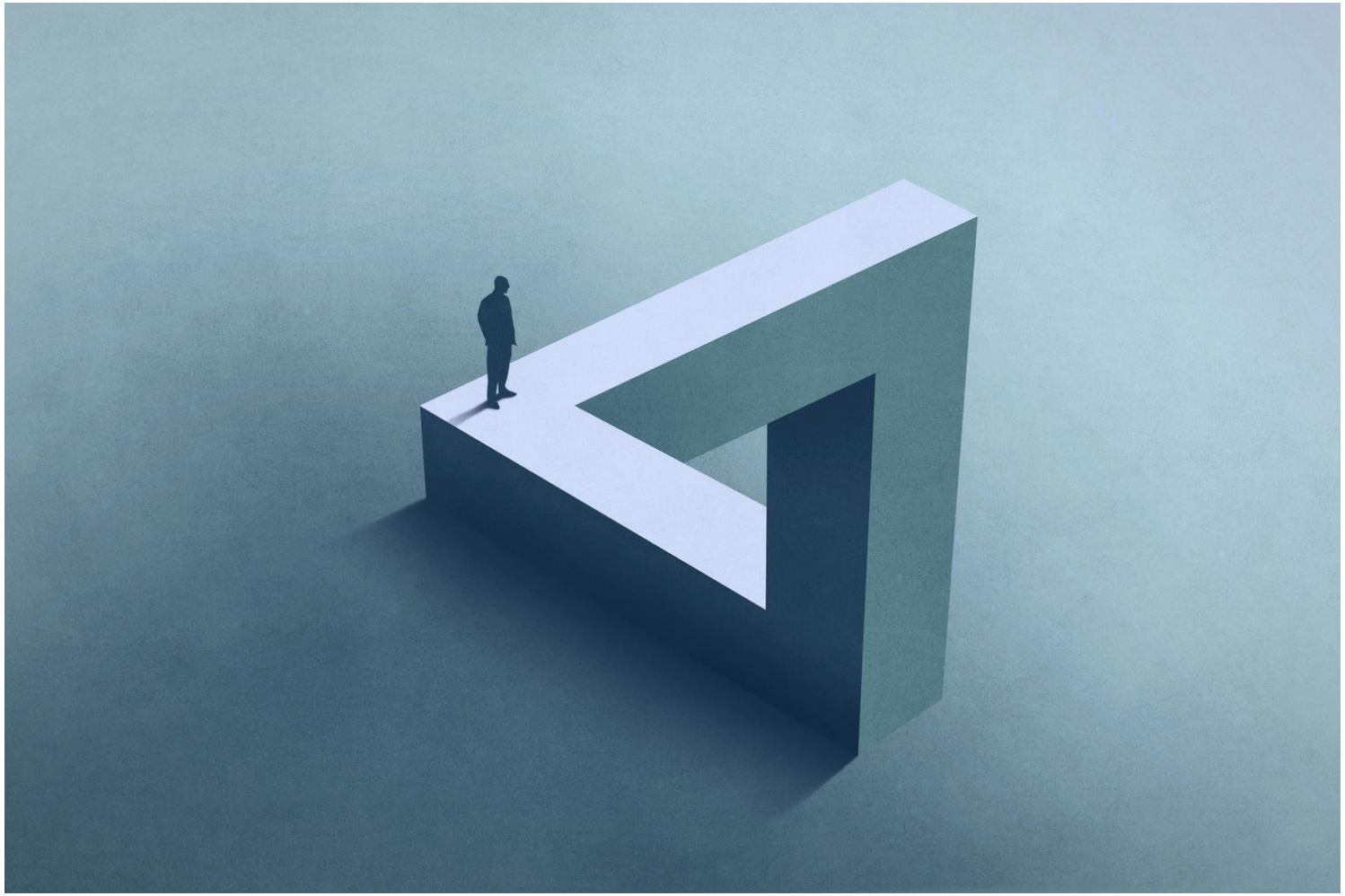




Prof. Dr. Frank J. Furrer

Software Development:
From Black Art to Engineering Science

Today's State ?

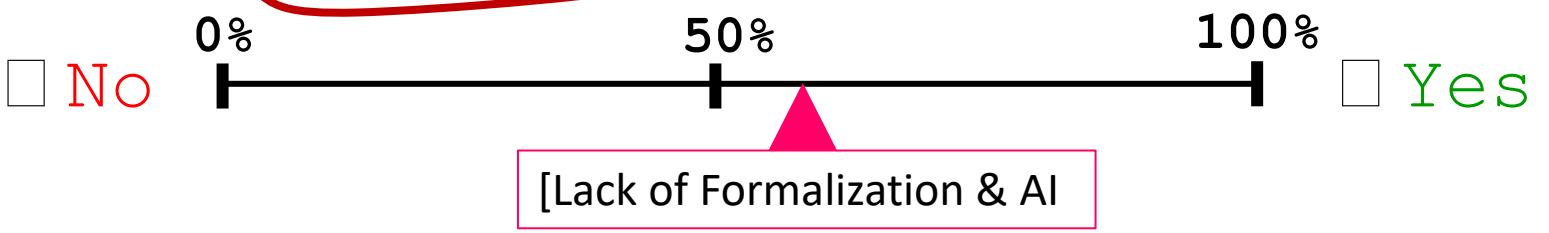


https://www.corporatecomplianceinsights.com/sustainability-reporting-where-do-we-stand/

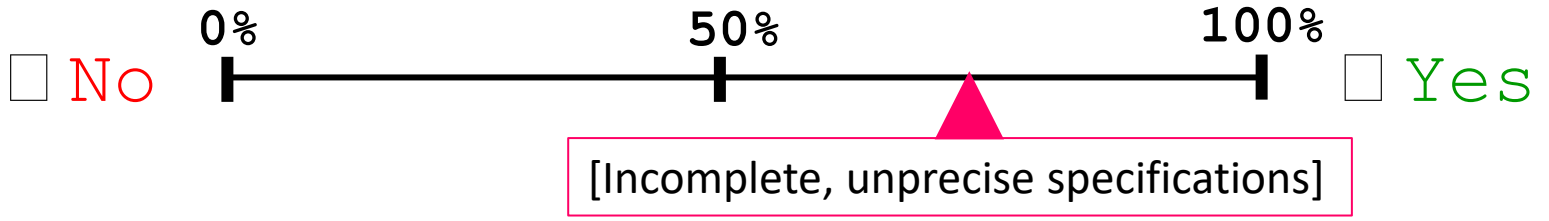
Where do we stand today?

Today's 5 Answers:

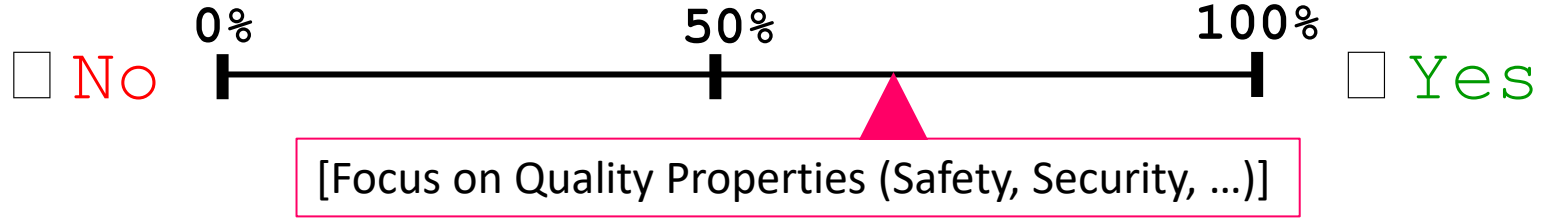
1) Is today's software development **already** an engineering science?



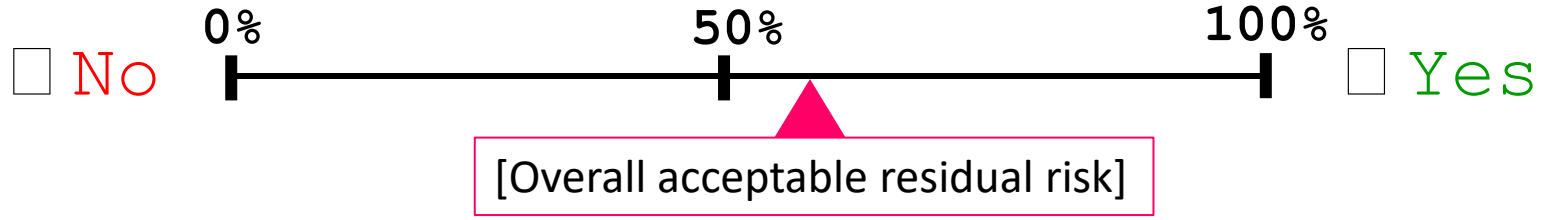
2) Does today's software development still have a component of **black art**?



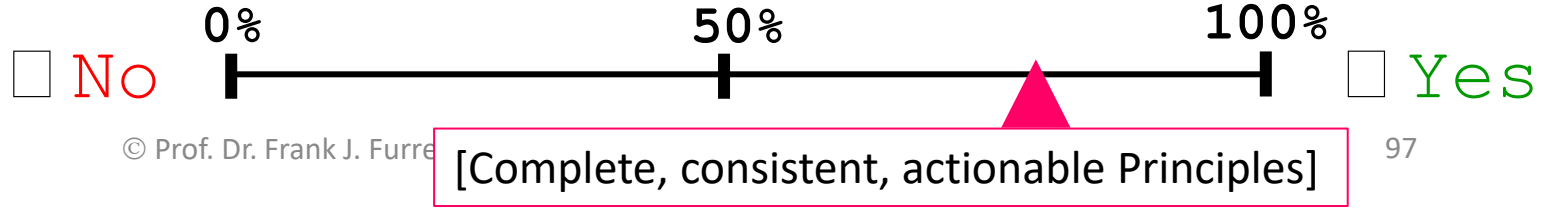
3) Is the **progress** for software development towards true engineering science sufficient?



4) Have we **reached** the state of engineering science?



5) Are the **mechanisms** to achieve the state of engineering science in place?



Topics

A pleasurable Promenade from 1848 to 2024

Principle-based Software Engineering

https://www.istockphoto.com/de/grafiken/software-development

https://www.freepik.com/free-photos-vectors/systems-engineering

Software Knowledge

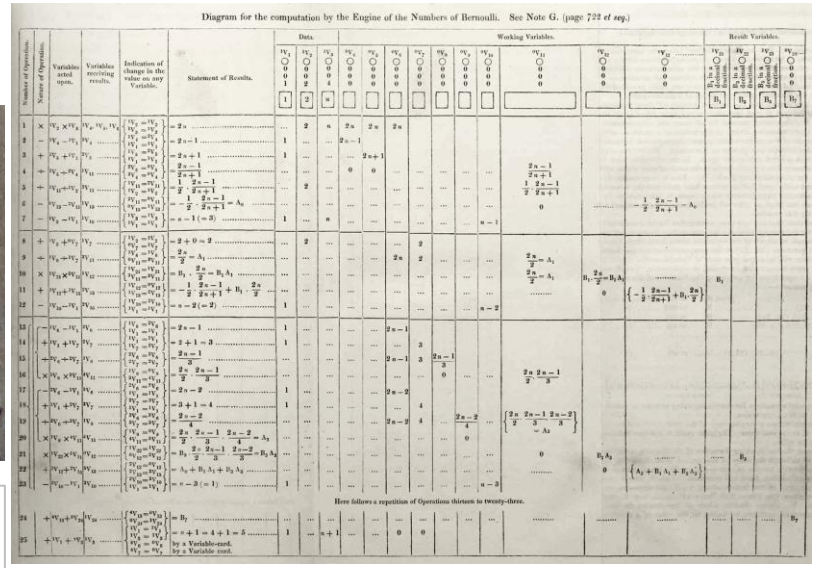
Continuous, accelerated generation of software-knowledge

Time

1848

2024

Ada Lovelace
1st Program



… < 100 Lines of Code

176 years

https://www.dreamstime.com

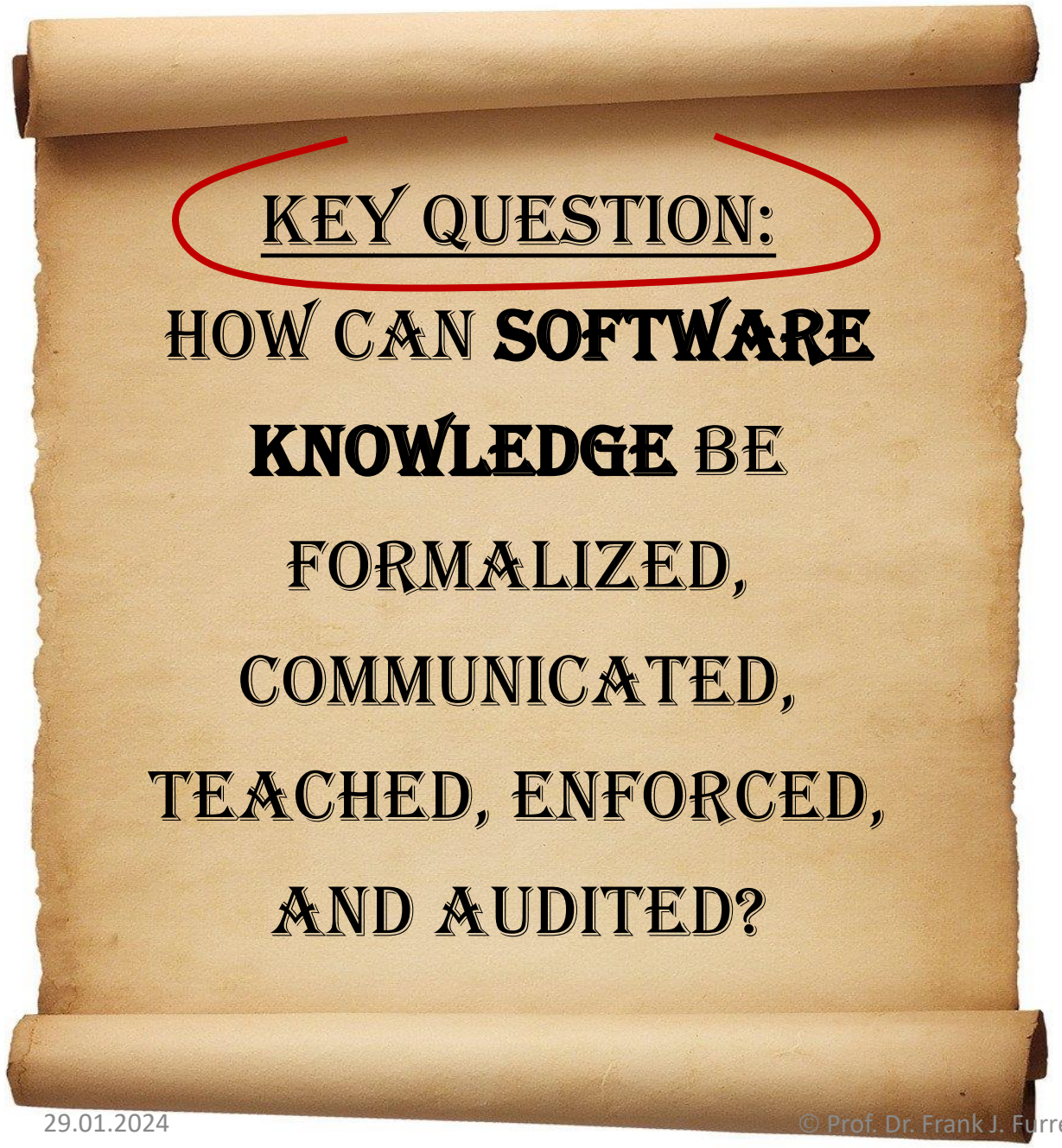
Global Software Market:
2022: 589 Billion (Milliarden)US$
2030: 1'789 Billion US$
+ Embedded Software
https://www.precedenceresearch.com

https://timenote.info/de/Ada-Lovelace https://projectlovelace.net/problems/ada-lovelaces-note-g

Prof. Dr. Frank J. Furrer

Software Development:
From Black Art to Engineering Science

System Principles



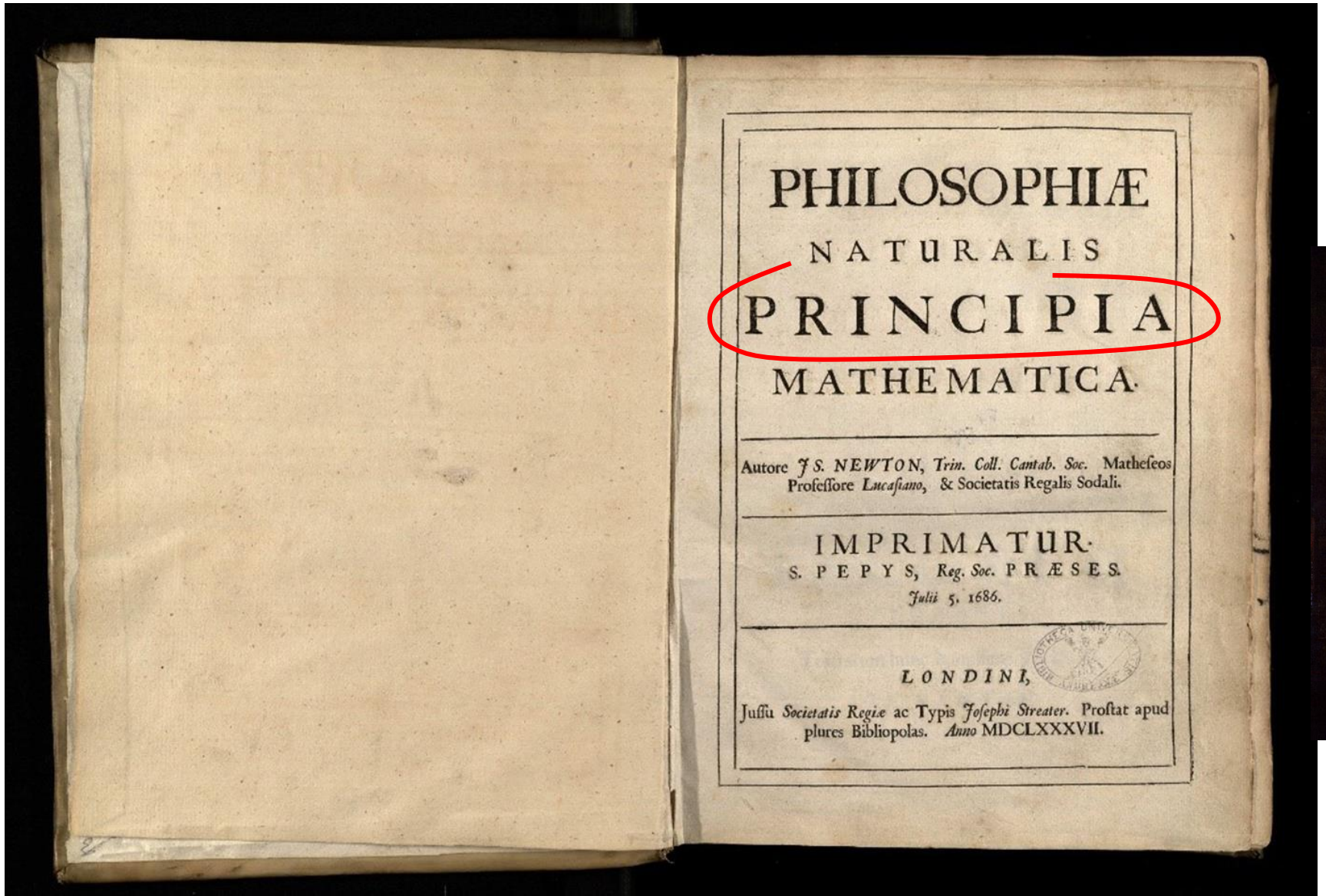
By formulating **PRINCIPLES**

Principles are the **knowledge treasure** of software development, operation, and evolution

https://www.freeimages.com/de/photo/parchment-1425941

https://www.numis.ch/ideen-fuer-geschenke/schatztruhe.php

Every mature science and dependable engineering discipline has a **proven set of principles**

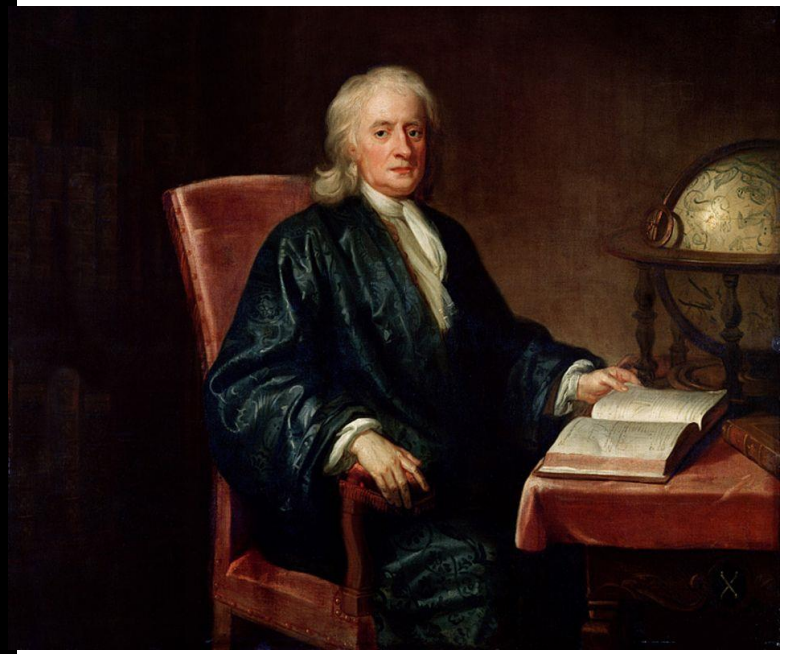


PHILOSOPHIÆ
NATURALIS
PRINCIPIA
MATHEMATICA

Autore *J S. NEWTON, Trin. Coll. Cantab. Soc.* Matheseos Professore *Lucasiano*, & Societatis Regalis Sodali.

IMPRIMATUR.
S. PEPYS, *Reg. Soc.* PRÆSES.
Julii 5. 1686.

LONDINI,
Jussu *Societatis Regiæ* ac Typis *Josephi Streater*. Prostat apud plures Bibliopolas. *Anno* MDCLXXXVII.



Isaac Newton
1687

https://special-collections.wp.st-andrews.ac.uk/2017/07/05/celebrating-330-years-of-isaac-newtons-principia/

https://de.postermywall.com

https://www.geo.de/geolino/mensch/isaac-newton-steckbrief-biografie-physik

1972

Principles

Programming Techniques
R. Morris Editor

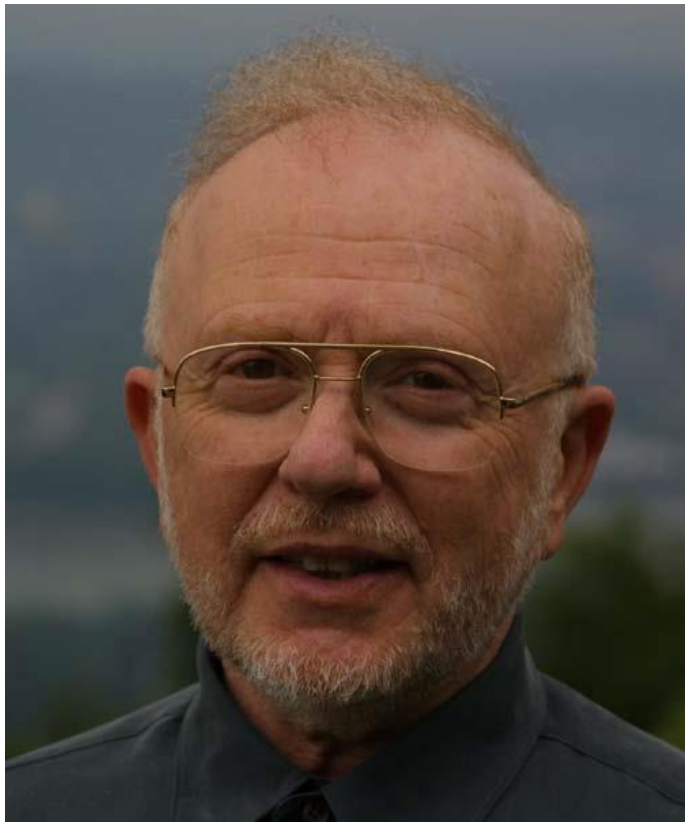
On the Criteria To Be Used in Decomposing Systems into Modules

D.L. Parnas
Carnegie-Mellon University

This paper discusses modularization as a mechanism for improving the flexibility and comprehensibility of a system while allowing the shortening of its development time. The effectiveness of a "modularization" is dependent upon the criteria used in dividing the system into modules. A system design problem is presented and

Introduction

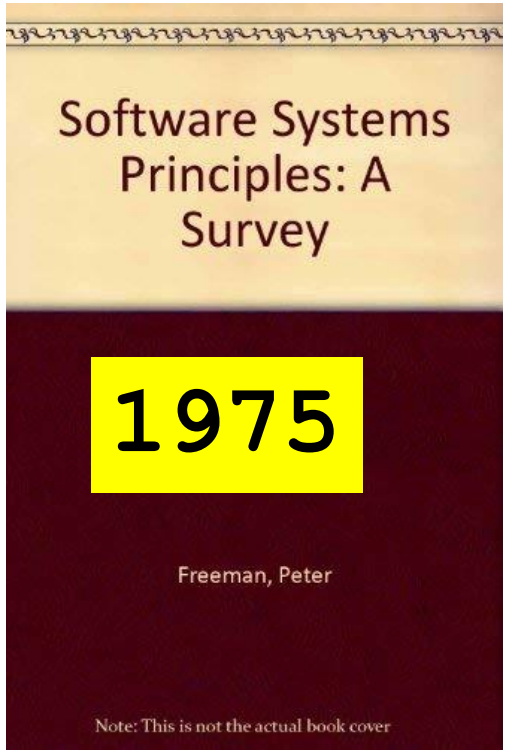
A lucid statement of the philosophy of modular programming can be found in a 1970 textbook on the design of system programs by Gouthier and Pont [1, ¶10.23], which we quote below:[1]



David L. Parnas
* February 10, 1941 in Pittsburgh, USA

http://datapeak.net/computerscientists.htm

Communications of the ACM, Volume 15, Number 12, December 1972
Available at: http://www.cs.umd.edu/class/spring2003/cmsc838p/Design/criteria.pdf



Principles are insights
based on scientific rules, laws, and proven know-how,
that are generally accepted by domain experts.

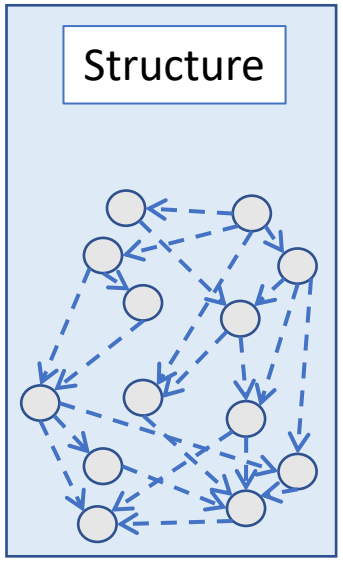
They are **fundamental truths** that form the foundation for the development, operation, and evolution of engineering artifacts

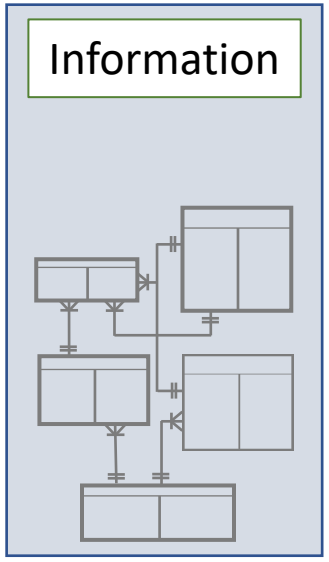
Today:

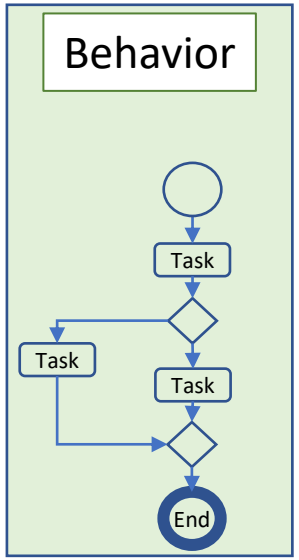
DEFINITIONS

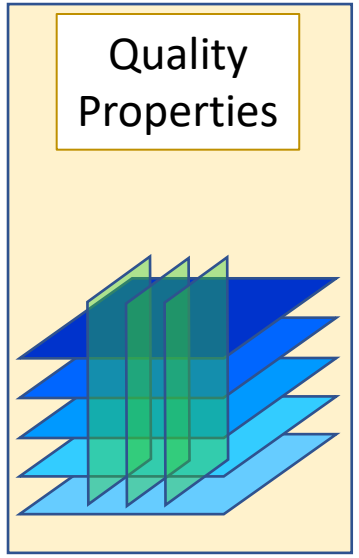
Architecture Principles:

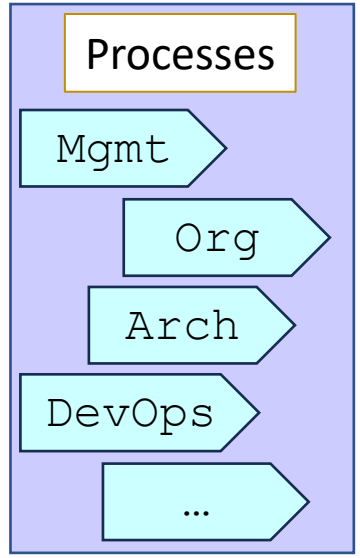
Fundamental insights – formulated as *enforcable rules* – how a good, future-proof software-system should be built [⇐ «Eternal Truths»]

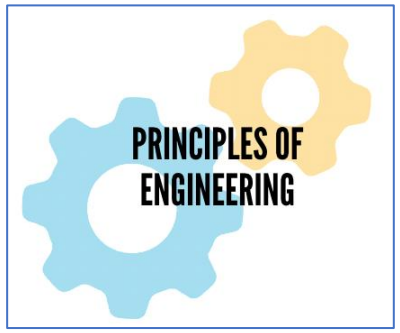


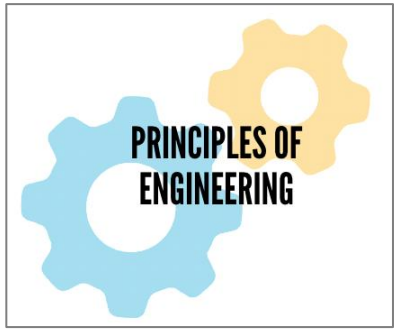


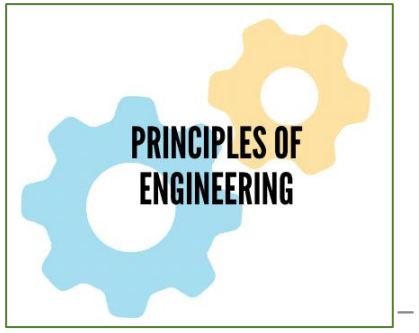


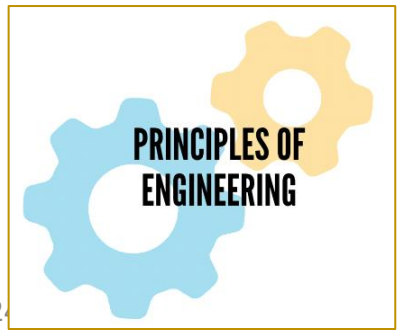


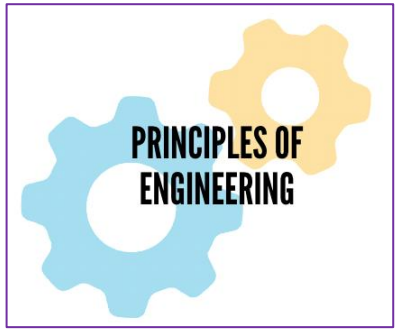






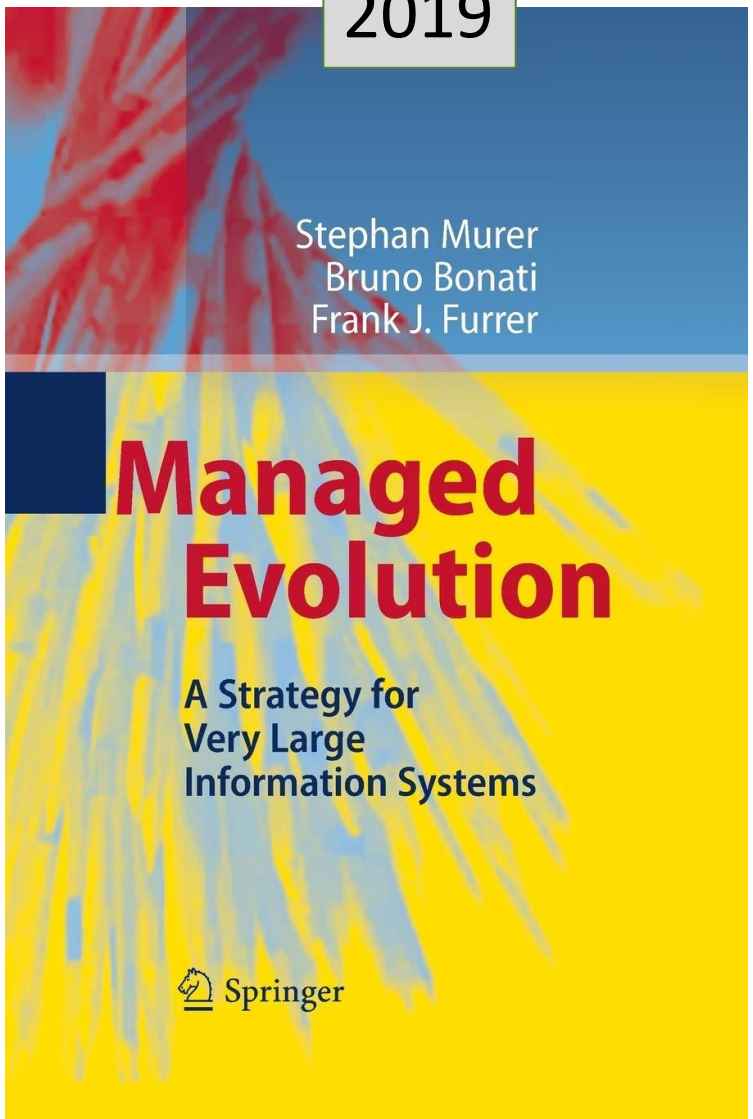




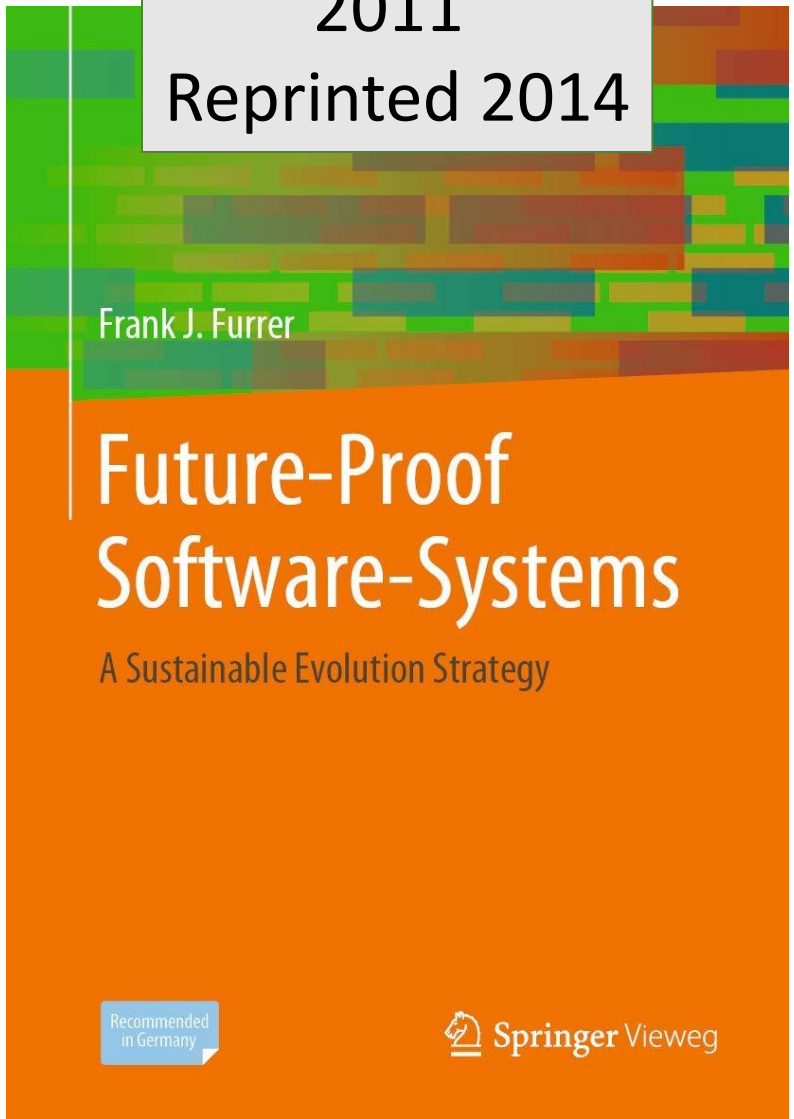


System Architure Principles and their Application:
⇒ My main Work as a System-Architect

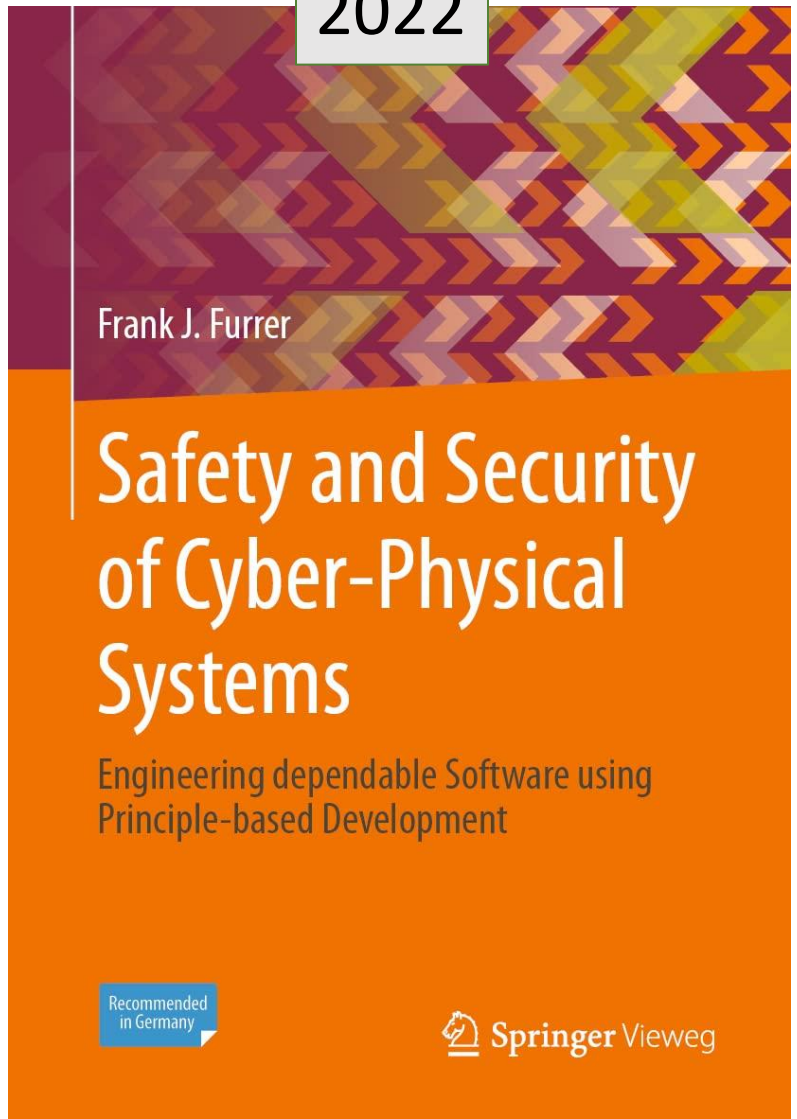
2019



2011
Reprinted 2014



2022



Principle Example 1: **Management**

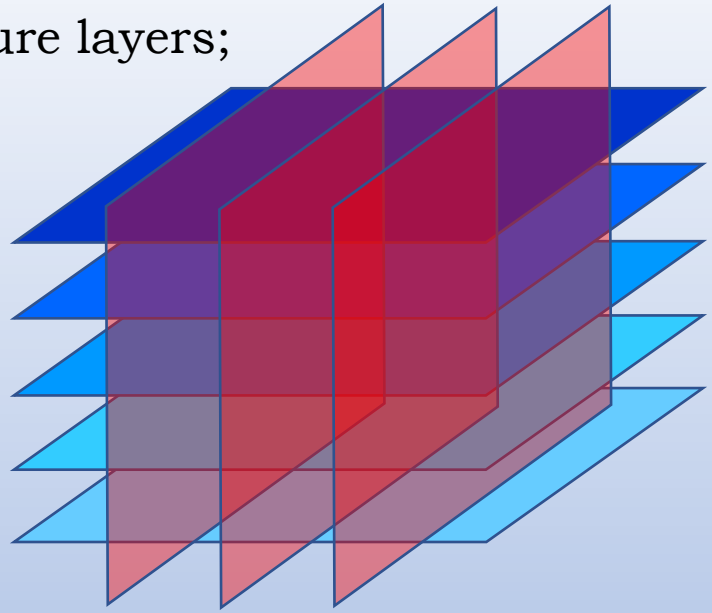
Principle 14-1: IT-System as Investment

1) Foster the point of view that the IT-system is a highly **valuable asset** with a high **investment value** on all levels of the organization;
2) Push the recognition that the IT-system must be maintained and evolved with **great care** and deep responsibility;
3) Fund and execute periodic, well-justified and well-planned **re-engineering, re-architecting or refactoring programs** under the direction of the IT chief-architect.

https://www.freepik.com/free-photos-vectors/gold-bag

Principle Example 2: **Architecture Layer Isolation**

Principle 15-1: Architecture Layer Isolation

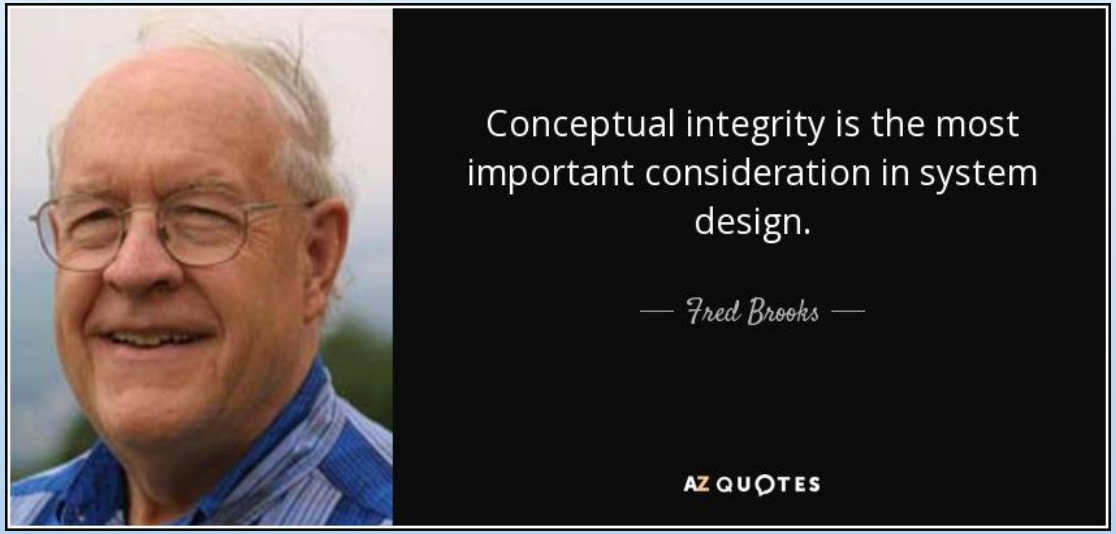


1) **Partition** your system into horizontal and vertical functional architecture layers;
2) Horizontal architecture layers:
   - Business Architecture,
   - Application Architecture,
   - Information Architecture,
   - Integration Architecture,
   - Technical Architecture;
3) Vertical architecture layers:
   - Quality Properties architecture layers [Safety, Security, …];
4) Assign the complete functionality and data/information of the IT-system to the **proper architecture layer**;
5) Always use industry-standardized, **technology-independent**, and product-independent mechanisms for transfer of data and control, and for access to functionality and information, between layers;
6) **Never** implement functionality from vertical layers in the horizontal layers (especially no technical functionality in the applications).

Principle Example 3: **Conceptual Integrity**

Principle 15-4: Conceptual Integrity

1) All stakeholders must have the **same, precise**, well documented, and unambiguous understanding of all concepts – both physical and virtual – used in the IT-system;
2) All stakeholders must share a common, well-defined **terminology**;
3) All stakeholders must base their work on an adequate, sound set of valid models, specifically recommended:
   - A domain model
   - A business object model
   - A structural model;
4) Use an **architecture-centric** development process which encourages and enforces conceptual integrity in all projects;
1) Install **strong governance** for architectural decisions;
2) When using concepts from other organizations, their meaning must be clearly understood – in some cases a **transformation** may be necessary;
3) Use adequate methods and mechanisms to assure **temporal integrity** in the distributed systems;
4) Never allow **hidden assumptions** in any of the development artifacts.



Principle Example 4: **Software Integrity**

https://clipground.com/safe-box-clipart.html

Principle 12-6: Software Integrity

Code signing: Protect the integrity of all software and firmware artifacts by a digitally signed hash (applied by the creator/developer of the soft-/firmware using his private key);

Check the integrity and intactness of all software and firmware artifacts **during boot/start-up** (using the public key of the developer);

Protect the integrity of all configuration and information artifacts by a **digitally signed hash** (applied by the developer of the soft-/firmware using his private key);

Check the integrity of all software and firmware artifacts during boot/start-up (using the public key of the developer);

Check the **consistency** (= match of the version numbers of all software, firmware, configuration files, etc.) of all artifacts during boot/start-up;

Implement correct error-handling in case of detected integrity faults or configuration mismatches. In such a case, never start up the system!

At all times, keep all related software artifacts, such as source code, models, configuration files, and documentation synchronized. Use round-trip engineering (RTE) development tools that support RTE;

If the risk analysis shows a probability for malfeasant control flow alteration, then use control flow integrity techniques;

Define and consistently apply an unequivocal and comprehensive artifact numbering/versioning scheme and use it consistently for all work products and for thorough archiving. Use an industry-standard version control system;

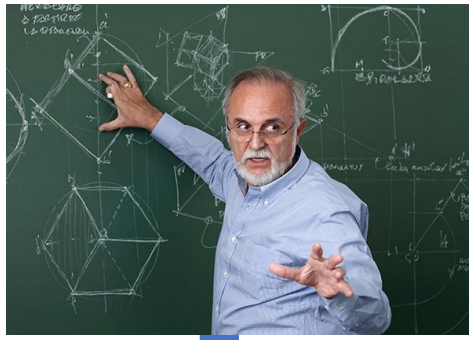
https://www.bairesdev.com/blog/how-to-manage-a-software-development-team/
https://slate.com/human-interest
https://www.dreamstime.com/illustration/police-officer.html

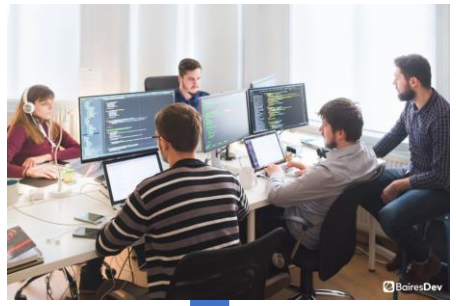
System Research

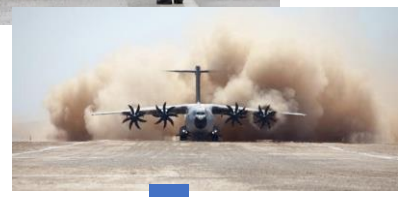
NIST

Standards

Experience







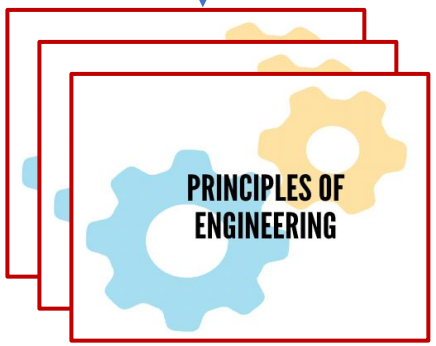
Principle Development

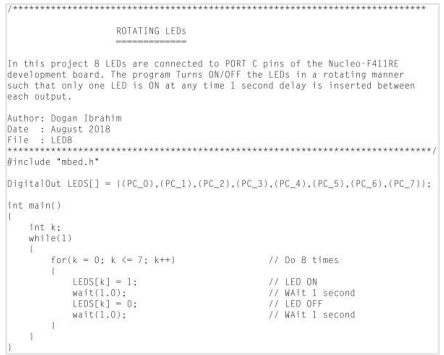
Principle Teaching

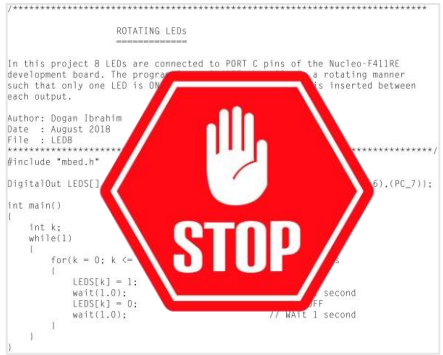
Principle Application

Principle Enforcement

Principle Maintenance







https://www.ucf.uni-freiburg.de

http://www.lovethispic.com

Thank you for your Attention

Questions?

Discussion?

https://www.istockphoto.com/de/vektor/fighting-gm460138041-31958576

My final words:

THANK YOU !

- ✓ To the TUD
- ✓ To the Faculty
- ✓ To my Students

It was one of the best times in my life

Lecture downloadable (pdf file) from:
https://st.inf.tu-dresden.de/teaching/Prof_FJ_Furrer/